
*Shealyn Cossette, Claire Fleckney,*

*Josh Hickman, Nathaniel Meyer,*

*Timothy Streibel, Bryan Towpich*

# Milestone Repository

*For*

## *Project "Open Road"*

*Project "Open Road" seeks to create a DBMS application that increases ease-of-access to services provided by Precision Riding Academy. It will also provide staff and students with the tools to create a community of like-minded individuals that support each other, with the goal of increasing the number of skilled motorcycle riders on the road.*

### Client Information:

*Bruce Streibel, President*

*Precision Riding Academy*

*1505 2 Avenue South, Lethbridge AB*

*(403) 394-6228*

*ridingacademy@ppslethbridge.com*

# Table of Contents

# Table of Contents Cont'd

# Table of Contents Cont'd

# Table of Contents Cont'd

# Table of Contents Cont'd

# 1. Milestone One

# 1.1. Team Introduction

This section will provide a basic introduction to the S.A.D. team known as "SIT SOFTware". For a more detailed description of team guidelines and methods see *Appendix A* (Team Charter).

## 1.1.1. Team Members

The team that was formed to complete the project named "Open Road" is comprised of the following six member:

| Name | Contact Information |
|---|---|
| Timothy Streibel | timothy.streibel@lethbridgecollege.ca<br>Cell: (403) 394-5787 |
| Josh Hickman | josh.hickman@lethbridgecollege.ca<br>Cell: (403) 894-6955 |
| Shealyn Cossette | shealyn.cossette@lethbridgecollege.ca<br>Cell: (403) 635-9778 |
| Bryan Towpich | bryan.towpich@lethbridgecollege.ca<br>Cell: (403) 360-7344 |
| Nathaniel Meyer | nathaniel.meyer@lethbridgecollege.ca<br>Cell: (403) 894-8386 |
| Claire Fleckney | claire.e.poulsen@gmail.com<br>Cell: (403) 929-1237 |

## 1.1.2. Team Branding

The team's name is SIT SOFTware, which is an abbreviation of "Students in Technology", and apt description of the team members.


*Figure 1.1.2 - SIT SOFTware logo*

The slogan is "Sit Back, Relax", playing on the team name.

The logo for the team is a comfortable looking chair, above the name and slogan, all in silver, on a black or transparent background. (See figure 1.1.2)

### 1.1.3. Team Member Roles and Responsibilities

The SIT SOFTware team has assigned the roles and responsibilities normally found in a software development team as follows. Each team member has been assigned a primary role and a more flexible secondary role.

#### Project Manager – Nathaniel Meyer

Nathaniel Meyer as the Project Manager is responsible for managing all aspects of the project. Nathaniel's secondary role is as the UI/UX specialist, designing all inputs and outputs for the system.

#### Infrastructure Analyst – Josh Hickman

Josh Hickman as the Infrastructure Analyst will focus on the technical issues involved with the Information System. Josh's secondary role is as the Lead Front-End Developer, ensuring that all front-end coding done by the team works properly with the back-end and database, and follows industry standards and defined conventions.

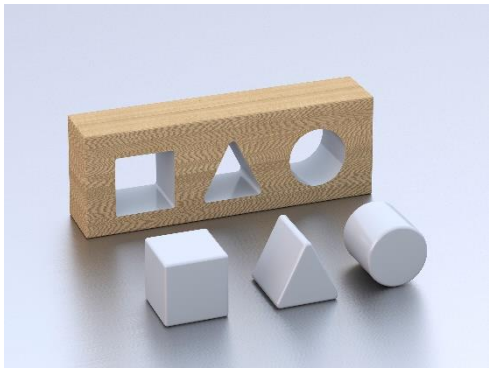
#### Change Mgmt. Analyst – Shealyn Cossette

Shealyn Cossette as the Change Management Analyst will ensure that proper version history is kept, as well as branch management. Shealyn's secondary role is as the HR Lead and Communication Specialist, making her the main point of contact between the team and the client, as well as managing disputes between team members.

## System Analyst –Bryan Towpich

Bryan Towpich is our System Analyst and will oversee the developing of ideas and suggestions for how the IS can fulfill the client's business needs. He will also ensure that all IS industry standards are maintained during development of the system. Bryan's secondary role is as the Database Specialist, ensuring that the database used by the IS contains all relevant information while maintaining proper structure and system integrity.

## Business Analyst – Claire Fleckney



Claire Fleckney is Business Analyst, focusing on the business issues regarding the IS, and eliciting requirements from the client. Claire's secondary role is as the Documentation Specialist, ensuring that all aspects of the project are properly documented and submitted to both the client and instructor.

## 1.2. Client Introduction

The following introduces the company for whom SIT SOFTware seeks to develop an information system solution.

### 1.2.1. Sponsoring Company

## Precision Riding Academy

Located at:

Precision Powersport Ltd.

1505 2nd Avenue South, Lethbridge AB T1J 0E8

Primary Contact:

Bruce Streibel

Office phone: (403) 394-6228

Email: ridingacademy@ppslethbridge.com

### 1.2.2. Company Background

Precision Riding Academy was established at the beginning of the year 2022. Its primary mission is to offer Motorcycle Rider safety Training to all those that want to acquire an Alberta Class 6 operators' license. In support of this mission, they offer the available courses at a price point of $299 per registrant. Comparable courses offered by other businesses see costs starting at ~$600 or more.

The courses run by Precision Riding Academy include 3 hours of theory learning and up to 14 hours of practical skills exercises. The theory portion of the course helps students to learn the different facets of owning and operating a motorcycle. The material covered includes the differences between motorcycle types, appropriate gear and why wearing gear is important, dealing with various environmental conditions, rules of the road, and potential situations and how to deal with them. The practical skills portion of the course is focused on helping riders become comfortable operating a motorcycle at low speeds, learn where the controls are and how to operate them, and various techniques required to handle a motorcycle safely.

### 1.2.3. Methods of Communication

Our primary method of communication will be by email through our Client Communication Specialist, Shealyn Cossette. After initiating contact with our client, we intend to schedule regular in-person meetings to facilitate the exchange of information and ideas. Each meeting with the client will be accompanied by an agenda detailing the contents of the meeting and distributed no later than 48hrs before the scheduled meeting time. Should the client's schedule not allow for an in-person meeting, a video conference meeting over an appropriate provider (MS Teams, Zoom, etc.) is an acceptable substitute.

## 1.3. Project Introduction

What follows is an introduction and outline of the information system being developed by SIT SOFTware, as requested by the Precision Riding Academy.

### 1.3.1. Project Request

The essential features of the system are that it should be able to store and retrieve user information and a small amount of course material in electronic format. The system also needs to be able to support business processes such as student registration, event scheduling, online payments, and staff management. It will also need to include some email utilities that will allow for more efficient targeted and mass messaging by instructors to students and/or the delivery of automated messages.

Some "*wish listed*" requests that we would like to include in some capacity, but only given there is enough time, include a social space for students and instructors, an interactive calendar, basic accounting support, and a mobile app for supporting instructors during a course. One way in which we could increase the likelihood of being able to include some of these wish listed items, would be to seek simple plugins that have few compatibility requirements and integrating them into the core system.

### 1.3.2. Possible Solutions

Local Application

- Taxing on physical computer systems.
- Challenging to access off site.
- System uptime is the organizations responsibility.

Pre-Built Software

- Higher purchase and maintenance costs.
- Redundant features in our context.
- Faster time to implementation and delivery.

Web Based Application

- Easily accessible regardless of physical location.
- Uptime guaranteed by a third party.
- Highly customizable, aiding in familiarity and feature specificity.

### 1.3.3. Known Constraints

Security:

Part of the user information we are dealing with will include sensitive data, such as Driver's License numbers. Therefore, it will be necessary that we implement strong data encryption and file storage protection.

Online Payments:

We cannot directly store or retrieve financial information related to interacting directly with banks or credit unions. As an alternative, we plan on including third party payment solutions like Interac e-transfer, Google Pay, and Apple Pay.

Network Access:

Wi-fi access is limited within the building, the business should be compelled to contact their service provider and correct this issue to ensure that the system is accessible throughout the building.

Available Hardware:

The computers/laptops currently available for instructor use are very basic, a factor relevant to the speed at which locally installed software will be able to operate.

## 1.4. Feasibility Analysis

To ensure that the selected project will bring about a positive result, the SIT SOFTware team has performed a feasibility analysis that justifies its potential.

### 1.4.1. Technical Feasibility

#### Familiarity with Application

Due to the lack of a pre-existing information system within the company, our primary familiarity concern is ensuring that the information system and interface we create are closely related to the physical documents and forms currently being used. The system we design should also remain compatible with pre-existing physical forms, so that in the case of a system outage, or an in-person signup, these forms can be used and later re-entered into the system. The staff at Precision Riding Academy also have some experience using web-based application, meaning there should be no difficulty in learning how to use the proposed information.

#### Familiarity with Technology

While the company will be installing a new computer terminal that will be used to access the system, it will run the same operating system and software as the laptop currently used by instructors to access and deliver course materials.

#### Project Size

While the scope of the project seems extensive, when taken individually the required features are not particularly robust but still provide a broad range of significant utility and optimization to the businesses processes. We plan to develop the systems components in a modular fashion, focusing on achieving core feature goals first, with special care being taken to ensure that these modules remain compatible with components that might be added even beyond the scope of this project. We aim to use effective plugins for features when applicable to reduce workload and compensate for scope creep.

## Compatibility

As there is no pre-existing application in place, our only options are to either build a system from the ground up or to purchase, customize, and install a prebuilt software package. Project requirements in our case are a broad spectrum of services, which a prebuilt system is unlikely to meet without the accompaniment of a plethora of other features that could pollute the system. It would therefore be more beneficial to create a system from scratch, allowing us to have greater control over the way in which the different features will integrate with each other.

## Decision Matrix

*Below is a comparison of key attributes between the potential solution and the current system. Each attribute is rated on a scale of how well it is supported by that system.*

| Good<br>Average<br>Poor | Business Goal Alignment | Performance | Scalability | Cost | Potential Risk | Duration | Legal Fit | Security |
|---|---|---|---|---|---|---|---|---|
| Proposed Solution | | | | | | | | |
| Current Processes | | | | | | | | |

*Table 1.4.1. – 1*

## Summary

In general, indicated technical requirements fit primarily into the category of achievable within the indicated timeframe. After reviewing the system request, a few requirements were labeled as wish-list items that we would like to tackle, but only given enough time once the core requirements have been accomplished. We found our clients requirements to be very practical requests, focusing on areas such as process optimizations and user accessibility, that are well within our teams' ability to deliver.

### 1.4.2. Economic Feasibility

The Precision Riding Academy is a "not-for-profit" organization, who's primary focus is helping new and experienced motorcycle riders develop potentially life-saving skills and habits. They offer their services at a low price point with the intent to make it accessible to as many individuals as they can while ensuring their operational costs are covered. Their primary investor, Precision Powersports Ltd., is also not looking to receive a tangible return for their involvement.

### Development Costs

SIT SOFTware is very excited to be able to help the Precision Riding Academy realize their goals through some of the benefits associated with "hiring" a student development team. While projected costs for hiring a developer could easily go into the tens of thousands of dollars range, SIT SOFTware will be performing the services pro-bono, while using the experience as a learning opportunity. As they are currently in the midst of setting up, many of the other costs associated with development, like acquiring needed technology and services, have been recognized and planned for.

## Operational Costs

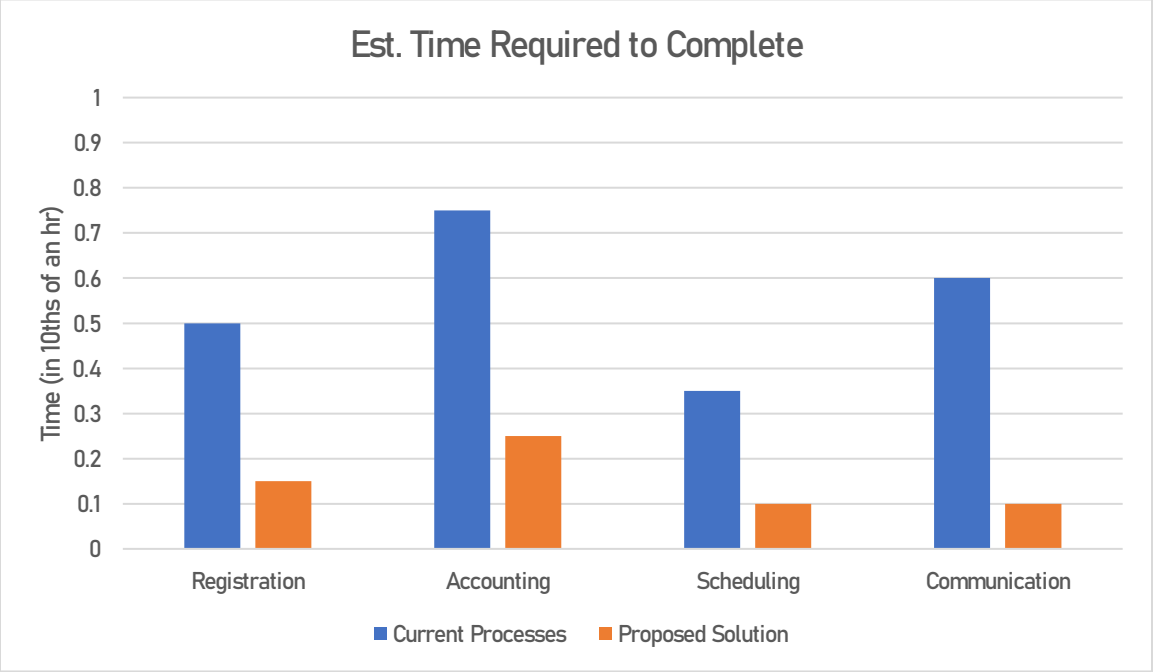



*Figure 1.4.2. - 1*

As can be seen in the above graph (*Figure 1.4.2. - 1*), Precision Riding Academy's current business processes are inefficient for handling the target amount of throughput that is their goal. As the number of scheduled courses, registered students, and instructors increase, more and more time will need to be allotted to complete processes related to administration. An appropriate information system that can automate many of these business processes will enable staff to redirect their efforts into tasks and projects intended for growth rather than just support. While there might not be a visual change in operational costs, stakeholders can rest assured that time is being spent in areas of greater benefit to the business. One of our goals as a development team will be to reduce the operational cost of the information system, thereby creating less resistance towards its development, installation, and use.

## Tangible and Intangible Benefits

Many of the tangible benefits will be apparent in the automation of business processes. The value of such a system, especially for a not-for-profit, is in the systems ability to support growth rather than in reducing costs. Such a system will also provide many intangible benefits such as an increased online presence, better user accessibility to services, and a greater market share. These intangible benefits will provide Precision Riding Academy with more opportunities to grow according to their goals.

## Summary

It would be extremely beneficial for Precision Riding Academy to have an appropriate information system developed for their use. By reducing the amount of time required for individual tasks, Precision Riding Academy will be able to handle the increase in throughput that will stem from the increase in market share. This is of immense value to Precision Riding Academy when considering their current company goals.

### 1.4.3. Organizational Feasibility

The economic feasibility study showed that an information system could provide immense value to Precision Riding Academy. Not only would a web-based system increase their presence and accessibility, but many of their current processes could be improved or automated. As an example, lets look at the student registration process.

## Process Comparison

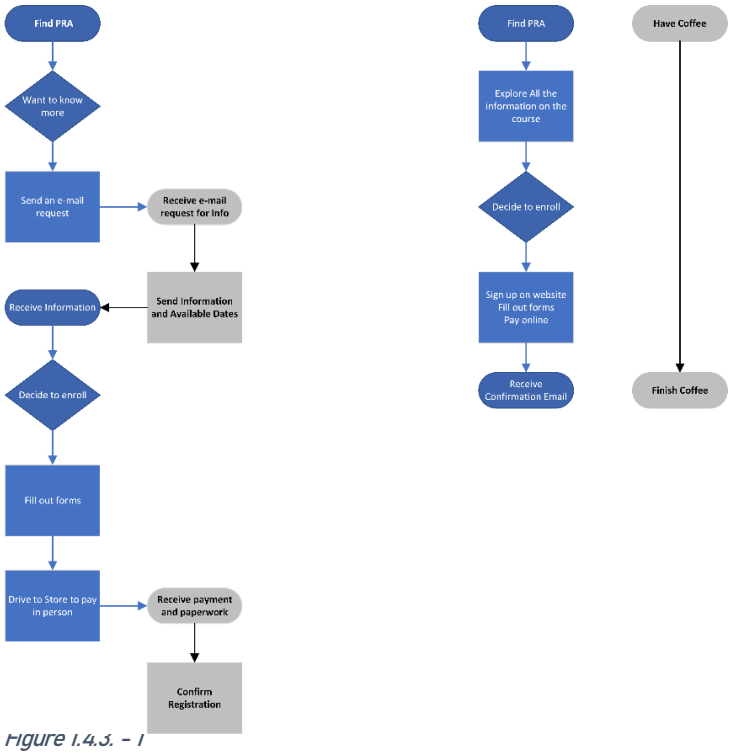



*Figure 1.4.3. - 1*

As can be seen in the flowcharts above (*Figure 1.4.3. - 1)*, the automation and improvement by way of an information system would drastically decrease the time and manpower required to complete the process. Because Precision Riding Academy does not have an information system currently in place to support their processes, we can safely assume that every process could be automated, improved, or redesigned to be more in line with company goals. The other important factor that needs to be considered is that for every step in a process that requires user input, the risk of errors or failing to complete the process increases exponentially. Reducing such risks is of particular importance for a growing business. Fewer of those risks mean more people purchasing services, staff can more easily complete tasks and responsibilities, and reporting the state of the business can be done more accurately.

## Stakeholder Resistance

So far, we've shown that the potential system will benefit many of the different stakeholders involved with Precision Riding Academy. The final point we need to think about now is whether there exists resistance to the potential system, or in other words is there any resistance to change? Are current users hesitant to change because of attachment to the current system? Does management think the hurdle of implementation and training is too high? Thankfully, the answer to these questions is a resounding "no". Recalling that Precision Riding Academy was founded only this past year, SIT SOFTware has the unique opportunity to provide software development services to a business still in its formative period. Being involved with a business at such an early point in its life means that it is less likely for staff to have attached themselves to a system that may only partially fill their needs. Another benefit is that SIT SOFTware can easily involve every staff member of Precision Riding Academy in the development of the information system. This is excellent news for both SIT SOFTware and the Academy as that means the features needed to fully support the Academy can more easily be discerned and accounted for.

## Summary

In summary, it can be said that Precision Riding Academy has chosen the perfect time to have a system developed for them. SIT SOFTware will need to take extra care to ensure that nothing is missed, but in the end, it is likely that they will have fewer than the usual number of problems encountered during the development of specialized software. It will not be difficult however, for SIT SOFTware to design the system to fulfill the business needs of the Academy.

### 1.4.4. Feasibility Analysis Summary

Based on the information provided by the analyses above, we can infer that an appropriate information system would bring numerous benefits to Precision Riding Academy. From a technical standpoint, the project is well within the capabilities of the SIT SOFTware development team. As well, because the Academy is still establishing themselves the system has the potential to further enhance their growth beyond what might have been expected. Economically this means that while operational costs will increase due to the growth, a good support IS will ensure that funds are used towards that expansion rather than just supporting current processes. Considering their goals as an organization, an IS such as the one in the early stages of development aligns perfectly with what Precision Riding Academy wants to accomplish. In conclusion, an information system that is specifically designed to suit their needs will only enhance what the Academy seeks to provide to their consumers.

## 1.5. Conclusion

The primary focus of this Milestone was setting up the project through team selection, client identification, and minimally justifying a chosen project. Our team was officially created, and project roles were selected. This created clear responsibilities within our team. Project selection allowed our team to choose a client. The wide range of clients allowed us to make significant choice as a team. And work together to find practical solutions to the problem. The feasibility analysis allowed our team to see how the project is possible. The feasibility analysis cleared up how much work our team would have to do and the time investments involved.

# 2. Milestone Two

## 2.1. Project Timeline

The following section will outline the projected timeline of project "Open Road". The first portion will provide a general overview of the anticipate completion dates for Milestones 1 through 10 as well as some other important events. The second portion will outline the deliverables required by each Milestone as well as detailed tasks for the completion of deliverables in Milestones 2 and 3.

### 2.1.1. Overview Timeline

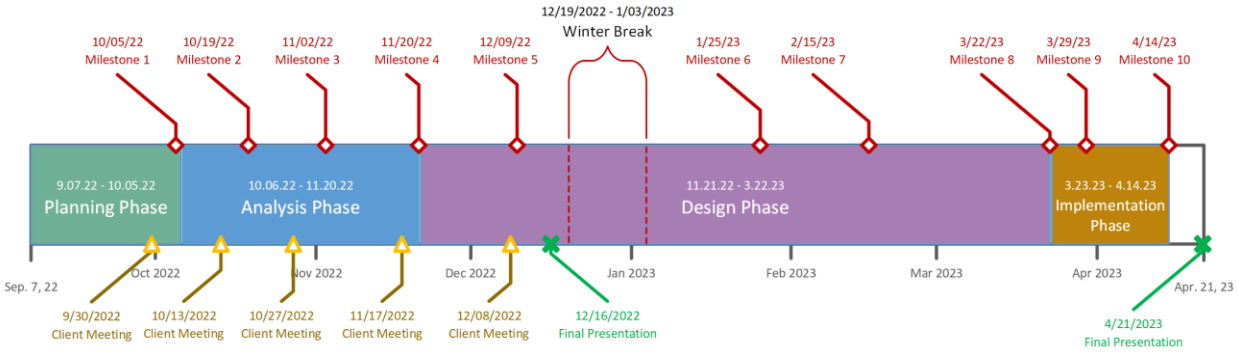



*Figure 2.1.1. - 1*

The above timeline (*figure 2.1.1 – 1)* shows the projected completion dates for each of the project milestones and how they relate to the different phases of the Systems Development Life Cycle. Segmenting each of the SDLC phases in milestones provides many benefits including creating a better platform for understanding the project, a more even distribution of work, and accurate tracking of project progress. It will be key for the SIT SOFTware to use every tool they have to their advantage.

### 2.1.2. Milestones 1 to 5

The following Gantt Chart shows the Milestones 1 to 5 along with their associated deliverables.

## Milestones 1 to 5

| ID | Task Name | Start | Finish | Duration |
|---|---|---|---|---|
| 1 | **Milestone 1** | **9/07/22** | **10/05/22** | **21d** |
| 2 | Team Introduction | 9/07/22 | 9/13/22 | 5d |
| 3 | Client Introduction | 9/15/22 | 9/20/22 | 4d |
| 4 | Project Introduction | 9/21/22 | 9/26/22 | 4d |
| 5 | Feasibility Analysis | 9/27/22 | 10/05/22 | 7d |
| 6 | **Milestone 2** | **10/06/22** | **10/19/22** | **10d** |
| 7 | Project Task Outline | 10/06/22 | 10/13/22 | 6d |
| 8 | System Requirements & Project Scope | 10/06/22 | 10/19/22 | 10d |
| 9 | **Milestone 3** | **10/20/22** | **10/31/22** | **8d** |
| 10 | Current System Model | 10/20/22 | 10/25/22 | 4d |
| 11 | Proposed System Model | 10/26/22 | 10/31/22 | 4d |
| 12 | **Milestone 4** | **11/01/22** | **11/18/22** | **14d** |
| 13 | Logical System Model | 11/01/22 | 11/18/22 | 14d |
| 14 | **Milestone 5** | **11/21/22** | **12/09/22** | **15d** |
| 15 | Feasibility Analysis | 11/21/22 | 11/30/22 | 8d |
| 16 | Summary Recommendation | 12/01/22 | 12/09/22 | 7d |
| 17 | Project Budget | 12/01/22 | 12/09/22 | 7d |
| 18 | Project Timeline | 12/01/22 | 12/09/22 | 7d |

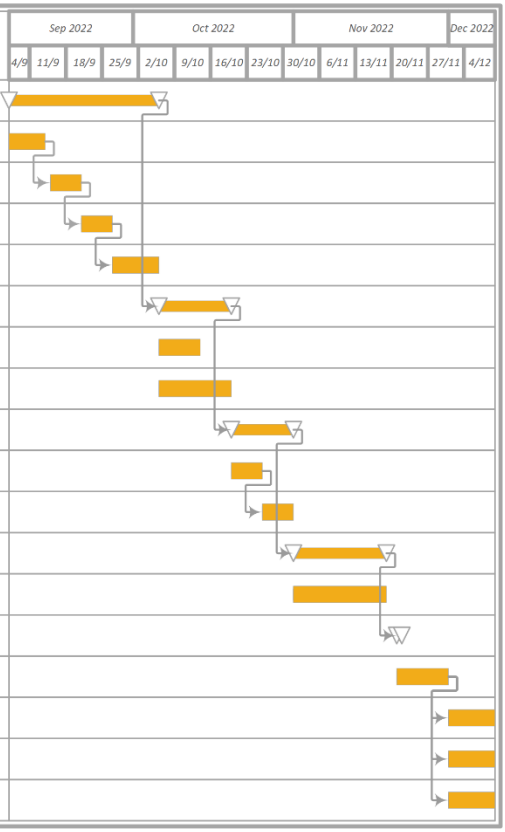
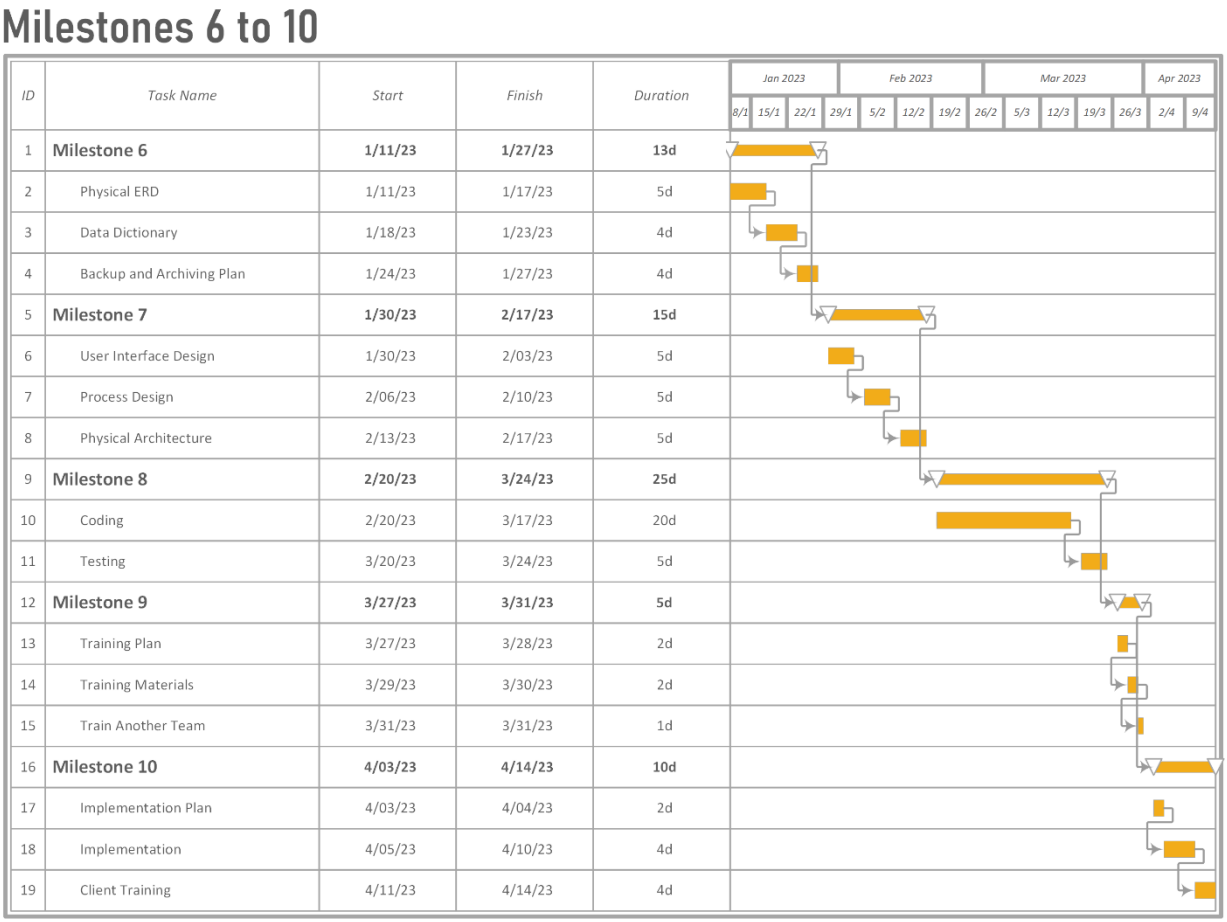



*Figure 2.1.2. - 1*

### 2.1.3. Milestones 6 to 10

The following Gantt Chart shows the Milestones 6 to 10 along with their associated deliverables.




| ID | Task Name | Start | Finish | Duration |
|---|---|---|---|---|
| 1 | **Milestone 6** | **1/11/23** | **1/27/23** | **13d** |
| 2 | Physical ERD | 1/11/23 | 1/17/23 | 5d |
| 3 | Data Dictionary | 1/18/23 | 1/23/23 | 4d |
| 4 | Backup and Archiving Plan | 1/24/23 | 1/27/23 | 4d |
| 5 | **Milestone 7** | **1/30/23** | **2/17/23** | **15d** |
| 6 | User Interface Design | 1/30/23 | 2/03/23 | 5d |
| 7 | Process Design | 2/06/23 | 2/10/23 | 5d |
| 8 | Physical Architecture | 2/13/23 | 2/17/23 | 5d |
| 9 | **Milestone 8** | **2/20/23** | **3/24/23** | **25d** |
| 10 | Coding | 2/20/23 | 3/17/23 | 20d |
| 11 | Testing | 3/20/23 | 3/24/23 | 5d |
| 12 | **Milestone 9** | **3/27/23** | **3/31/23** | **5d** |
| 13 | Training Plan | 3/27/23 | 3/28/23 | 2d |
| 14 | Training Materials | 3/29/23 | 3/30/23 | 2d |
| 15 | Train Another Team | 3/31/23 | 3/31/23 | 1d |
| 16 | **Milestone 10** | **4/03/23** | **4/14/23** | **10d** |
| 17 | Implementation Plan | 4/03/23 | 4/04/23 | 2d |
| 18 | Implementation | 4/05/23 | 4/10/23 | 4d |
| 19 | Client Training | 4/11/23 | 4/14/23 | 4d |

*Figure 2.1.3. - 1*

### 2.1.4. Milestone 2 and 3 Details

The following Gantt Charts show the tasks associated with completing the deliverables for Milestones 2 (*figure 2.1.4. – 1*) and 3(*figure 2.1.4. – 2*), along with the resource assignments for each task.

## Milestone 2 Details

| ID | Task Name | Start | Finish | Duration | Resource Names |
|---|---|---|---|---|---|
| 1 | **Project Timeline** | **10/06/22** | **10/13/22** | **6d** | |
| 2 | List Milestones and Basic Deliverables | 10/06/22 | 10/07/22 | 2d | Tim Streibel |
| 3 | List Tasks for M2 and M3 | 10/10/22 | 10/11/22 | 2d | Tim Streibel |
| 4 | Create Timeline for meetings | 10/12/22 | 10/12/22 | 1d | Tim Streibel |
| 5 | Assign Resources to listed tasks | 10/13/22 | 10/13/22 | 1d | Tim Streibel |
| 6 | **System Requirements & Project Scope** | **10/06/22** | **10/19/22** | **10d** | |
| 7 | Define Problems and Opportunities | 10/06/22 | 10/07/22 | 2d | Josh Hickman |
| 8 | Investigate Physical Site | 10/10/22 | 10/11/22 | 2d | Josh Hickman |
| 9 | Investigate/Define Business and System P&P | 10/10/22 | 10/11/22 | 2d | Claire Fleckney |
| 10 | Investigate/Define system users | 10/12/22 | 10/13/22 | 2d | Shealyn Cossette |
| 11 | Define & list Requirements | 10/14/22 | 10/17/22 | 2d | Bryan Towpich |
| 12 | Define project scope | 10/18/22 | 10/19/22 | 2d | Nathaniel Meyer |

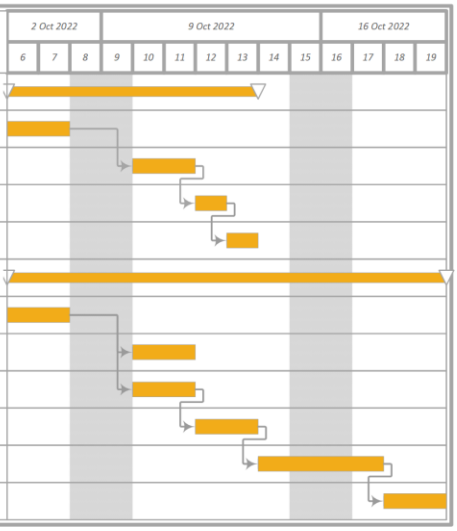



*Figure 2.1.4. - 1*

## Milestone 3 Details

| ID | Task Name | Start | Finish | Duration | Resource Names |
|---|---|---|---|---|---|
| 1 | **Current System Model** | **10/20/22** | **10/31/22** | **8d** | |
| 2 | Compile reference list of current system processes | 10/20/22 | 10/21/22 | 2d | Tim Streibel |
| 3 | Create and define Use Cases for current system processes | 10/24/22 | 10/26/22 | 3d | Shealyn Cossette |
| 4 | Create and define DFDs for current system processes | 10/27/22 | 10/31/22 | 3d | Claire Fleckney |
| 5 | **Proposed System Model** | **10/20/22** | **11/01/22** | **9d** | |
| 6 | Compile reference list of proposed system processes | 10/20/22 | 10/24/22 | 3d | Josh Hickman |
| 7 | Create and define Use cases for proposed system processes | 10/25/22 | 10/27/22 | 3d | Nathaniel Meyer |
| 8 | Create and define DFDs for proposed system processes | 10/28/22 | 11/01/22 | 3d | Bryan Towpich |

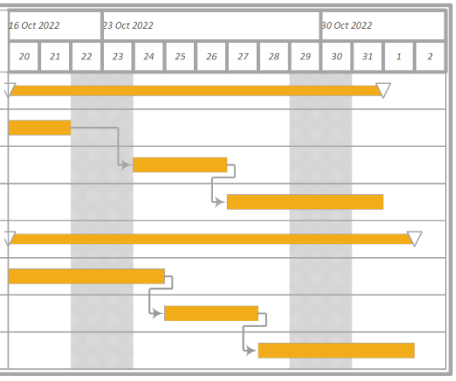



*Figure 2.1.4. - 2*

## 2.2. System Requirements and Scope

### 2.2.1. Problems and Opportunities

The following bulleted points lay out the currently identified business and technical problems prohibiting the optimal execution of regular business operations. Opportunities presented for us as a team of developers implementing a new system are proposed in the paragraph which follows. Our ability to tackle these problems and opportunities is addressed further in the project scope segment of this document.

#### Business Problems

- It currently takes Instructors a long time to communicate with current and prospective students.
- Payment must be done in-person at the store but is required prior to enrollment.
- Minimal branding or advertising presence for the Precision Riding Academy.
- Lack of online social utilities makes it challenging to stay in touch with or build communities around previous or current students.
- No previous student testimonies to indicate the courses level of quality and show that it is a current and well-utilized program within the city.
- Paper records are challenging to physically store and generate useful information from.

#### Technical Problems

- Limited Wi-Fi availability within the building.
- Very basic primary computer (point of access) right now.

## Opportunities

Having database records of relevant student and course data rather than physical forms will allow for the generation of interesting and relevant reports that will be of value to the instructors and leaders of the program as it moves past its inaugural year. Similarly, making course content more readily available by creating online repositories accessible by students and instructors will aid in the scalability of the program moving forward, as well as posing immediate value through increased convenience. Storing these records and files digitally will be more secure than current systems as well as allowing for the generation of useful metrics that can be used to improve outreach and making it easier to deliver the same high-quality course to more people at a time.

While current network infrastructure poses minor limitations, there have been indications of discontent within upper management regarding the current Wi-Fi connectivity and web hosting services, meaning there is room for input on our part as information technology professionals to assist as we see fit. Drawing people towards an online platform presents the opportunity to better promote the course through graphical advertising, articles, and videos as well providing the means to perpetuate a community of likeminded individuals. Moving more of the programs features online means we can implement online payment alongside the current in-store option to ensure that we are maximizing program accessibility. As the program grows, should we organize information within the system well (as is our goal), new instructors will be easier to train and monitor if they are given access to this comprehensive system.

### 2.2.2. System Requirements

The following table (*table 2.2.2. - 1)* lists business, user, functional, non-functional, and system requirements related to both core and wishlist features.

| Requirement | Requirement Type | Description | Core or Wishlist |
|---|---|---|---|
| Online Registration | Business | Allowing users to register online reduces paperwork and storage space requirements | Core |
| Online Payment | Business | Allowing users to pay online will increase the likelihood of successful registration | Core |
| Publish Events | Business | Creating a process for making an event that adds it to the calendar for appropriate users, and sends an email to those that have opted for such emails | Core |
| Course Management Reports | Business | Database functionality that allows reports to be generated on topics such as: student registration, instructor schedules, etc. | Core |
| Accounting Functionality | Business | Being able to keep track of revenue and expenses within the system itself would help with the business | Wishlist |
| Accounting Reports | Business | Database functionality that allows reports to be printed. Examples: Income, Expenses, Financial Statements, etc. | Wishlist |
| Mass Email Functionality | User | Giving the administrator the ability to send out mass emails will reduce their workload | Core |
| Calendar | User | A personalized calendar for each user will help keep everyone informed on classes and other events | Core |
| Social Space | User | Having a space for student comments and pictures will increase company reach | Wishlist |
| Payment Processing through 3rd party | Non-Functional | Using 3rd Party Payment systems to accept payment for the classes to expedite registration | Core |
| Security | Non-Functional | Each login will keep information private for each individual user. The system will make sure each login only gets the correct info following appropriate security protocols | Core |
| Information Storage | Non-Functional | Store all information received from the registration into the system database for the user | Core |
| Email Groups | Non-Functional | Allowing the server to send emails to everyone in a group when the group is written in the address bar | Wishlist |
| Registration Records | Functional | The system will take, and store all needed information from the registration and store it into a database | Core |
| Class scheduling | Functional | Scheduling all the classes and controlling who can register and when for the class (ensuring maximum of 5 students per class) | Core |
| User Account Login | Functional | Separate logins for each user (students and staff) with access to the correct area for each user | Core |

| | | | |
|---|---|---|---|
| User level View | Functional | Ability for Administrators to change views to see what other user levels can see | Wishlist |
| Webserver Hosting | System | The system will be hosted on an appropriate web server to facilitate online access for users | Core |
| Backup onsite Data Storage For all Paperwork | System | All-important data will have an onsite backup | Core |
| Stable Internet | System | A stable internet connection is required to ensure data can be accessed as needed | Core |

*Table 2.2.2. - 1*

### 2.2.3. Project Scope

## Scope Declaration

| Project "Open Road" | | |
|---|---|---|
| **What we Can do** | **What we Might do** | **What we Can't do** |
| Course/Event Creation & Publishing | Interactive Calendar | Direct Payment Online (Debit/Credit) |
| Variable Volume Email Communication | Integrated Accounting | Data Entry |
| Online Registration & Payments | Integrated Social Space | Creating/Managing Social Accounts |
| Course Document Cloud Storage | Mobile Support App for Instructors | |
| Course Management Reporting | | |
| Multi-Level user accounts | | |
| FAQ + Other Common Website Page Templates | | |
| PPS Advertisement | | |
| Web Hosting Recommendations | | |

*Table 2.2.3. - 1*

## Summary of Project Scope

Our focus with this project will be to provide a wider range of functionality to Precision Riding Academy and its users. The core features that will for sure be implemented are centered around course management (i.e., creating/publishing courses and events, course/event registration, variable volume communication, cloud document storage, etc.). Some of the additional features that the Academy is looking for are to support additional business processes like social media (expanding online influence) and accounting. Adding these features directly into the proposed IS will reduce the likelihood that the Academy will need to purchase additional software to support these processes. Our planned method of increasing the possibility of getting to these features during development is to use pre-built plugins that will work alongside the core software.

### 2.2.4. Community of System Users

Types of System Users and Additional Roles

| System User Types | Description |
|---|---|
| Admin | This is a top-level user able to remove, change or add roles and privileges. They are also allowed to make events and post mass messaging when a notice is to be given out. |
| Instructor | This is the second level of account this account can be switched to, added or removal of different roles that would allow people to use them in different ways. |
| Student | This level is the most basic user level. Where people who want to join the site for classes or be part of the community that is grow getting an account is the first step. |
| **Additional Roles** | **Description** |
| Accounting | This role will deal with and financial recording and reporting. |
| Social Media | This role is in control of social media would have access to the social aspect of the website promoting online functions and events that maybe tied to other social Aspects. |
| Front Desk | This is a role for a frontline employee if someone comes to pay in person this will give access to the billing to add new bills and accounts. |
| Developer | This is a level of user that can gain access to the back but not the private information of the clients or business. They can move, remove, and add new functions and new process to the web app. |

*Table 2.2.4. - 1*

System User Community

| System Users | User Type(s) | Description |
|---|---|---|
| Bruce | Admin/Instructor | • Account balance sheet<br>• Event creation<br>• Role assignment<br>• Remove, add, move students and Instructors<br>• Payment lists<br>• Mass email<br>• Approval of images or posts |
| Jacob | Instructor | • In charge of classes<br>• Direct contact with the students<br>• List of class students<br>• List of teaching times<br>• Access to emergency documents ex, incident forms |
| Tim | Instructor/Social Media/Developer | • Ability to help with social media<br>• To promote classes and events.<br>• Can take over-flow classes if he needed to.<br>• List of class students<br>• List of teaching times<br>• Approval of images or posts<br>• Make changes to the website<br>• Add Features<br>• Fix Bugs and Glitches<br>• Access to emergency documents ex, incident forms |
| Trish | Accounting | • Balance sheets<br>• Bills and receipts<br>• Approval of costs<br>• The management of classes and the instructors |
| Student body | Student | • To sign up to classes and an account<br>• To also look up class dates and events<br>• Participation in the community<br>• Access to various PRA support documents |

*Table 2.2.4. - 2*

### 2.2.5. Business and System Policies & Procedures

## Current Procedures

Currently, the Department Head of the Precision Riding Academy manages all registration and scheduling. Students fill out a form on the Precision Powersports website, requesting more information and the registration package. Once the student has received the registration package, they must fill it out and bring it to the Academy, and either pay for the course in person or via e-transfer. The completed registration package is kept in a file cabinet in the Riding Academy classroom. Payment status is also kept with the registration package.

Currently there is no online schedule for students to check. If they have any inquiries regarding the date and time of their course, they must call in to the Academy. Additionally, students can request a certain weekend for their class, but the Department Head is scheduling on a class-by-class basis.

When communicating with groups of people regarding the Riding Academy, the Department Head is currently writing out an email, and sending it individually to each recipient, changing the email address every time.

As can be seen in the descriptions above, the current processes used by Precision Riding Academy are very time consuming. It will be especially noticeable as the business expands its operations in the coming years. As more consumers seek their services, more time will be required to complete the processes. Their current processes are not easily scalable, thereby hindering the potential growth of the Academy.

## Effects of the New System

With the new system, registration will be managed by Students and the Information System. Students will choose the weekend that they wish to register for (assuming there is availability) and fill out the online registration form while logged in to their account. The completed registration form, along with any pertinent student information, will be stored digitally, and the student's account will be flagged as paid or unpaid.

Payment will be available online for students, either through e-transfer or third-party plugins. If the student pays online, their account will be flagged as paid once the payment has been confirmed. If the student decides to pay in person, the employee that taking the payment will be able to make a note on the student's file that payment has been received.

A calendar on the main website will show class schedules and availability, as well as a more personalized calendar with reminders for the student once they are logged in to their account.

A mass email system will be put in place so that the Department Head can email all members in certain groups of people (past students, current students, instructors, etc.) all at once.

## Possible Conflicts

One possible conflict regarding implementation of the new system is regarding payment processing. Since payments will no longer only be processed in person or through e-transfer, the student's financial information may be shared with the third party that is facilitating the payment. While having a third-party handling payment processing can be a risk, it should not create too many issues so long as the businesses selected are those who have established themselves well with current financial and web security practices. In the end, it may provide more benefits than potential risks as increasing the number of options for processing payments may increase the likelihood that consumers will complete purchases.

Another possible conflict involves student personal information, such as Name, Driver's License Number, etc. Since this information will be stored in the system's database, it may be slightly more vulnerable to theft than if it was still only stored on-site in a file cabinet. We can reduce the risk of information theft by carefully researching and applying current IS security practices during the design phase of the project.

## 2.2.6. Investigation of Physical Site

### Physical Layout of Precision Riding Academy

The program is currently being delivered and managed primarily from a smaller office space with a projector, three rows of student tables, and an instructor desk which holds the currently used laptop that is connected to the projector and businesses network. The Riding Academy does have plans to move the program to being delivered from a different room, adjacent to the current area. The new space will be a good deal larger to facilitate bigger classes but is otherwise very similar to the current point of delivery. After delivering course content in one of these rooms the instructors will take students outside for a practical lesson.

### Map of Premises


*Image 2.2.6. - 1*

## Layout Significance Within Project Scope

In relation to our scope of relevance within this project the physical layout of the business is largely irrelevant, so long as there is at least one internet connected computer terminal, easily accessible by instructors. This requirement is met under the current circumstances and will need to be met should the programs delivery be moved to the other room. Our due diligence will be to ensure that the company is aware of the changes that will need to be made to their current infrastructure to ensure the continued delivery of the program, namely the rewiring of the ethernet connection currently connecting the instructor laptop to the internet to ensure that it reaches the new computer position.

## 2.3. Conclusion

The primary focus of the project that can be gleaned from the information presented in the System Requirements and Scope document is Business Process Management. With Precision Riding Academy having just finished their inaugural year as we begin development, they have little in the way of system processes to support the various services and features they wish to offer their staff and clientele. Once the new system is developed and implemented, system users will be able to interact more fully with the Academy. The new IS will provide a stable platform upon which the Precision Riding Academy can grow and expand into their preferred market.

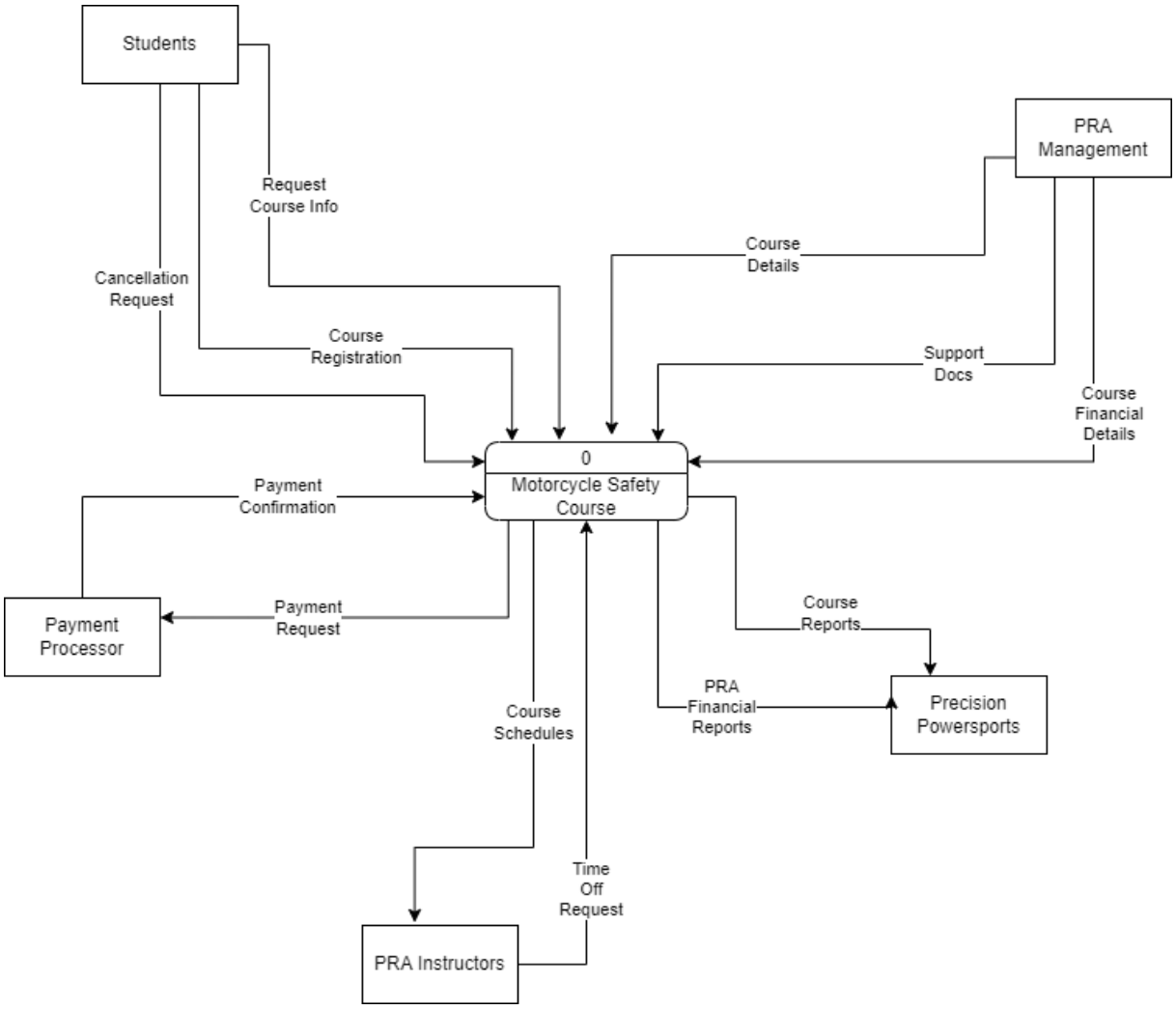
# 3. Milestone Three

## 3.1. Current System Models

As part of SIT SOFTware's analysis, we have created the following Contextual Diagram, Use Cases, and Data Flow Diagrams to better understand the current system and find gaps in our possessed knowledge.

### 3.1.1. Contextual Diagram

The following contextual diagram details the primary external entities and their interactions with the current system. As can be seen below (*figure 3.2.1. -1*), the current system's area of influence is very small and its ability to support growth and expansion is extremely limited.



### 3.1.2. Use Cases

#### 3.1.2.1. Summary Table

| UC ID | Ext/Temp | Priority | Use Case Name | Actor | Trigger | Use Case Outcome |
|---|---|---|---|---|---|---|
| CS-UC1 | Ext | High | Register for a Scheduled Course | Student | Initiate Contact with Riding Academy | Student is registered for course |
| CS-UC2 | Ext | Med | Registration Cancellation | Student Body | Student is unable to attend class | Student will be rescheduled or refunded |
| CS-UC3 | Ext | High | Schedule New Course Instance | Department Head | Sufficient interest in new course date | Course occurrence is scheduled |
| CS-UC4 | Ext | Med | Create New Course Type | Department Head | Sufficient interest in new course type | New course type is created |
| CS-UC5 | Ext | High | Reporting PRA Statistics | Department Head | Monthly/weekly report due | President is given completed report |
| CS-UC6 | Ext | High | Request Time Off | Instructor | Instructor wants a day off | Instructor has the day off |
| CS-UC7 | Ext | High | Email Registered Students | Department Head | One week/one day before scheduled course date | Email is sent to all participating students |

*Table 3.1.2.1. - 1*

## CS-UC1

| **Use Case Name:** Register for a scheduled course | **ID:** CS- UC1 | **Priority: High** |
|---|---|---|
| **Actor:** Student | | |
| **Description:** This use case describes how a student submits a Course Resgistration | | |
| **Trigger:** Initiate contact with riding acedmy | | |
| **Type:** ☒External ☐ Temporal | | |
| **Preconditions:** | | |
| 1. Student is at least 16 years old | | |
| 2. Student has a Class 5 license | | |
| 3. Student has active network connection | | |

| **Normal Course:** | **Data Flow for Steps:** | |
|---|---|---|
| 1.0 Course Registration | | |
| 1. Student looks up wed page | | |
| 2. Student fills out online"Request Information" form and send it to management | → | "Request Information" form |
| 3. Students wait for contact email from management | | |
| 4. Management sends a .zip file package containing registration forms | ← | "RegistrationPackage.zip" file |
| 5. Students ask for a course time that falls with in there time limits | → | Schedule Request |
| 6. Management looks to see the available course with in the time limit. | ← | Available Scheduled Courses |
| 7. Student confirms time of date | → | Selected Course |
| 8. Students pays for course | → | Payment Confirmation |
| a. Payment call in credit card number | | |
| b. Payment in person | | |
| 9. Management sends a remind email | → | Email Reminder |
| | | |

| **Postconditions:** |
|---|
| 1. Student is registered for a scheduled course |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| Registration Details | Student | Information Packet | Student |
| Registration Payment | Student | Course Reminder | Manager |
| | | | |
| | | | |

*Figure 3.1.2. - 1*

## CS-UC2

| **Use Case Name:** Regisration Cancellation | **ID:** CS-UC2 | **Priority:** Med |
|---|---|---|
| **Actor:** Student Body | | |
| **Description:** This use case is the discription on how you cancel a registration | | |
| **Trigger:** Student can't make it to the class | | |
| **Type:** X External □ Temporal | | |
| **Preconditions:** | | |
| 1. Student can't make it to the class | | |
| 2. Student didn't pay for the class in time | | |
| | | |

| **Normal Course:** | **Data Flow for Steps:** | |
|---|---|---|
| 1.0 Regisration Cancelled | | |
| 1. Students can't make it to the class | → | Cancel request |
| a. Class is canceld due to unpaid invoices | | |
| b. Student calls to cancel a class | | |
| 2. Management Recives a cancellation notifcation | ← | Canellation email |
| a. Management Reopened Spot for new student | | |
| 3. Management look at date of cancelation and determins if that refund is within time policy. | | |
| a. If it's within policy the refund will proceed | | |
| b. Reschdule the course time. | | |
| 4. Management If they paid they get there money back or a reschedule to another the class | ← | Confirmation of refund |
| | | |

| **Postconditions:** |
|---|
| 1. Student will be rescheduled or Student will be refunded |
| |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| Cancel request | Student | Cancelation confirmation email | FileCabit |
| | | Confirmation of refund | Accounting |
| | | | |

*Figure 3.1.2. - 2*

## CS-UC3

| **Use Case Name:** Request For Time off | **ID:** CS-UC3 | **Priority:** Med |
|---|---|---|
| **Actor:** Instructor | | |
| **Description:** This use case describes how an Instructor submits a request for time off. | | |
| **Trigger:** Indviduale wants a day off, and sends an email to Admin | | |
| **Type:** ☒ External ☐ Temporal | | |
| **Preconditions:** | | |
| 1. Instructor requires time off | | |

| **Normal Course:** | **Data Flow for Steps:** |
|---|---|
| 1.0 Instructors request for time off | |
| 1. Instructor request time off → | Date is requested |
| 2. Admin checks there calander | |
| 3. Admin makes a decision ← | Date is decied upon |
| a. Admin can support the request | |
| b. Admin can't support the request, return to step 1 | |
| 4. Admin schdules Instructor off. ← | Date is confirmed |

| **Postconditions:** |
|---|
| 1. Instructor now has the time off |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| Resuested date | Instructor | Approved RTO | Instructor |

*Figure 3.1.2. - 3*

## CS-UC4

| **Use Case Name:** Schedule a new course occurrence | **ID:** CS-UC4 | **Priority:** High |
|---|---|---|
| **Actor:** Department Head | | |
| **Description:** This Use Case describes how an administrator schedules a new course date | | |
| **Trigger:** Sufficient interest in a new course date | | |
| **Type:** ☒ External ☐ Temporal | | |
| **Preconditions:** | | |
| 1. Instructors available on course date | | |
| 2. Course materials printed for students | | |

| **Normal Course:** | **Data Flow for Steps:** | |
|---|---|---|
| 1.0 Schedule a new course occurrence | | |
| 1. Select an appropriate weekend | ← | Course Date |
| 2. Record new course instance using approriate paperwork | → | New Course Occurrence |
| 3. Alert Precision Powersports of newly scheduled course | → | New Course Alert |
| 4. Email students who have registered for a seat but no course was available yet | → | New Course Date |
| 5. Adjust student registration as needed | → | Student Registrations |

| **Postconditions:** |
|---|
| 1. Newly scheduled course occurrence |
| 2. Students who wish to participate in new occurrence have had registration changed over |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| Course Date | Department Head | New Course Occurrence | File Cabinet |
| Student Registrations | File Cabinet | New Course Alert | Precision Powersports |
| | | | |

*Figure 3.1.2. - 4*

## CS-UC5

| **Use Case Name:** Generate PRA statistics report | **ID:** CS-UC5 | **Priority:** Med |
|---|---|---|
| **Actor:** Department Head | | |
| **Description:** describes how an administrator reports statistics | | |
| **Trigger:** Monthly / Weekly Report due | | |
| **Type:** ☒ External ☐ Temporal | | |
| **Preconditions:** | | |
| 1. Information to be reported | | |
| 2. Department head is using a document processing software | | |
| | | |

| **Normal Course:** | **Data Flow for Steps:** | |
|---|---|---|
| 1.0 Generate PRA statistics report | | |
| 1. Retreive relevant data | ← | Report data |
| 2. Enter period of report | | |
| 3. Enter data | | |
| 4. Format report | | |
| 5. Save file | | |
| 6. Send file to President | | |
| | → | Generated Report |
| | | |
| | | |

| **Postconditions:** |
|---|
| 1. Report has been generated |
| 2. President has received report |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| RelevantData | File Cabinet | Report | President |
| | | | |
| | | | |

*Figure 3.1.2. - 5*

## CS-UC6

| **Use Case Name:** Send email reminder to registered students | **ID:** CS-UC6 | **Priority:** Med |
|---|---|---|
| **Actor:** Department Head | | |
| **Description:** describes how an Department Head sends out a mass email to register students of upcoming course | | |
| **Trigger:** One week or one day before course date | | |
| **Type:** ☒ External ☐ Temporal | | |
| **Preconditions:** | | |
| 1. Students registered for upcoming course | | |
| 2. Department Head has active network connection | | |
| 3. Department Head has email correspondence software open | | |

| **Normal Course:** | **Data Flow for Steps:** | |
|---|---|---|
| 1.0 Send email reminder to registered students | | |
| 1. Retreive list of registered students | ← | List of Registered Students |
| 2. Write body of email | | |
| 3. Enter address of target student | ← | Student Email |
| 4. Send email | → | Reminder Email |
| 5. Check if all students have been emailed | | |
| a. If fail, return to step 2 | | |
| b. If pass, exit process | | |

| **Postconditions:** |
|---|
| 1. Reminder email is sent to all applicable students |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| List of Registered Students | File Cabinet | Reminder Email | Student |
| Student Email | List of Registered Students | | |
| | | | |

*Figure 3.1.2. - 6*

### 3.1.3. Data Flow Diagrams

Processes:

1. Sign up for Course
2. Running the Course

Data Stores:

1. File Cabinet

Section 1.01 Student Registration forms
Section 1.02 Student Invoices/Receipts
Section 1.03 Course Documents/Manuals
Section 1.04 Operations Receipts

## Level 0

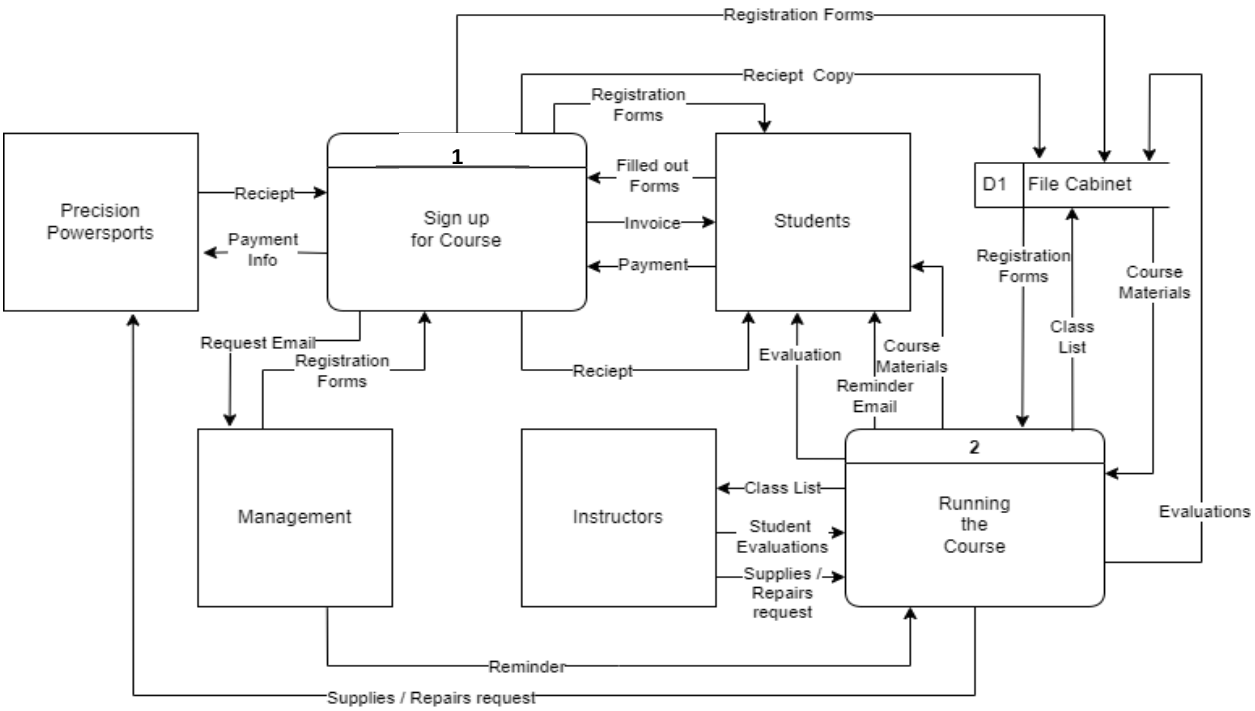



*Figure 3.1.3. - 1*

## 3.2. Proposed System

As part of SIT SOFTware's analysis, we have created the following Contextual Diagram, Use Cases, and Data Flow Diagrams to better understand the proposed system and find gaps in our possessed knowledge.

### 3.2.1. Contextual Diagram

The following contextual diagram details the primary external entities and their interactions with the proposed system. As can be seen below (*figure 3.2.1. -1*), the proposed system's area of influence is much greater and should provide a wider range of benefits and features to all parties involved.

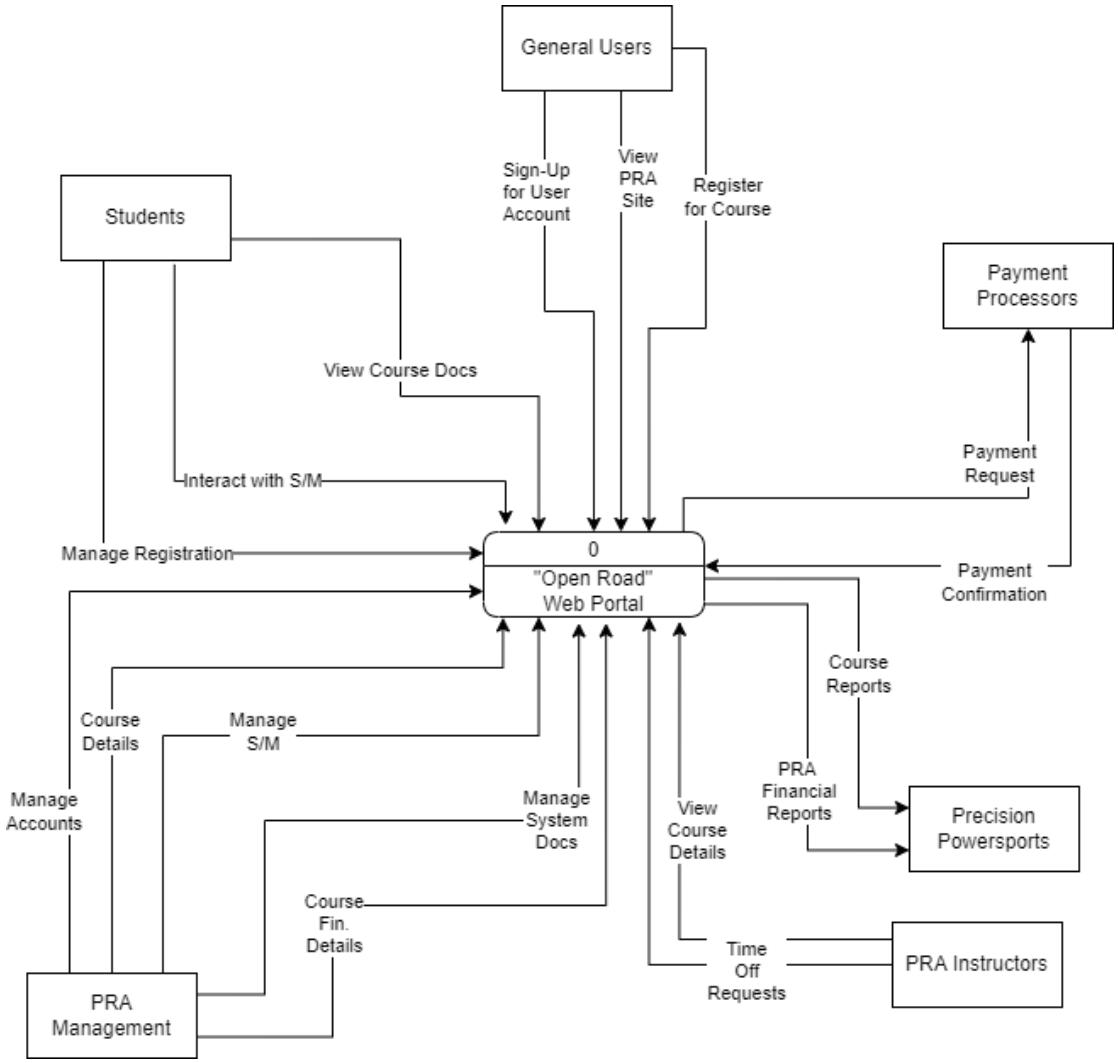



*Figure 3.2.1. - 1*

## 3.2.2. Use Cases

### 3.2.2.1. Summary Table

| UC ID | Ext/Temp | Priority | Use Case Name | Actor | Trigger | Use Case Outcome |
|---|---|---|---|---|---|---|
| PS-UC1 | Ext | High | Sign Up for a User Account | Guest | Sign-Up Button is pressed | User Account is Created |
| PS-UC2 | Ext | High | Log/Sign into User Account | User | Log in/Sign in Button is pressed | User is logged into account |
| PS-UC3 | Ext | Med | Edit Account Information | User | Edit Account Button is pressed | User is able to edit the details of their account |
| PS-UC4 | Ext | High | Log/Sign Out of online account | User | User Presses "Sign Out" Button | User is signed out of current session |
| PS-UC5.1 | Ext | High | Pre-Register for Scheduled Course | Guest | Guest Presses "Register" Button | Guest is Pre-Registered for Course |
| PS-UC5.2 | Ext | High | Pay and Confirm | Guest/Student | Guest/Student Presses "Pay and Confirm Registration" Button | Student is registered for course |
| PS-UC6 | Ext | High | Create New Course Type | PRA Management | Create New Course Type Button is pressed | New course type is created |
| PS-UC7 | Ext | High | Update Course Details | PRA Management | Update Course Details Button is pressed | Course details have been updated |
| PS-UC8 | Ext | High | Remove Course from System | PRA Management | Remove Course Button is pressed | Selected course is removed from system |
| PS-UC9 | Ext | High | Schedule a Course | PRA Management | Schedule Course Button is pressed | Course is scheduled |
| PS-UC10 | Ext | High | Create Specialized User Accounts | PRA Management | Staff Member is hired and requires appropriate system priveleges | Employee is given an account with sufficient priveleges |
| PS-UC11 | Ext | High | Update User Account Permissions | PRA Management | Staff Member requires elevated system priveleges | Employee Account is given required priveleges |
| PS-UC12 | Ext | High | Remove User Accounts | PRA Management | A user account of any kind has been deemed redundant or threatening | User Account is deleted from system |
| PS-UC13 | Ext | High | Generate Course Statistics Report | PRA Management | Generate Course Report Button is pressed | Course report is generated |
| PS-UC14 | Ext | High | Review Assigned Course Info | Instructor | Instructors request to view course info | Instructor views desired course files |
| PS-UC15.1 | Ext | Med | Request Time Off | Instructor | Instructor presses "RTO Request" Button | RTO Request is added to the pool |
| PS-UC15.2 | Ext | Med | Approve Time Off | PRA Management | Manager accesses RTO pool | Manager approves/denies time off request |
| PS-UC16 | Ext | High | Email Users | PRA Management | Management identifies information that is worth sharing to multiple users or specific groups | Manager sends message to specified users |
| PS-UC17 | Ext | Med | Email Students | Instructor | Instructor opens message box within web portal | Desired group is sent an Email |

*Table 3.2.2.1. - 1*

# PS-UC1

| **Use Case Name:** Sign up for a user account | **ID:PS-US1** | **Priority:** High |
|---|---|---|
| **Actor:** Guest | | |
| **Description:** describes the process for creating a new user account | | |
| **Trigger:** the sign-up button is pressed | | |
| **Type:** ☒External ☐ Temporal | | |
| **Preconditions:** | | |
| 1. The user has an active network connection | | |
| 2. The user does not already have a user account | | |
| 3. The user is not part of the "blacklist" | | |
| 4. There is not a user account already under that username | | |
| 5. The "Open Road" web application is online | | |

| **Normal Course:** | **Data Flow for Steps:** | |
|---|---|---|
| 1.0 sign up for a user account | | |
| 1. The system brings up the "create user account" form | ← | "Create user account" form |
| 2. The user fills out the "create user account" form and presses "submit" button | → | User Information |
| 3. The system checks the entered information against the "blacklist" | ← | "Blacklist" query result |
| a. If the information matches an entry in the "blacklist", proceed to Alternate Path 1.0 | | |
| b. If the information does not match an entry in the "blacklist", proceed to step 4 | | |
| 4. The user enters an appropriate username and password and presses "submit" button | → | Username and Password |
| 5. They system checks to make sure the username does not already exist | ← | User Accounts query result |
| a. If the username already exists, return to step 4 | | |
| b. If the username does not already exist, proceed to step 6 | | |
| 6. The system checks to make sure the password entered follows set standards | ← | Password standards query result |
| a. If standards are not met, alert the user and halt progress until an appropraite password is entered | | |
| b. If standards are met, proceed to step 7 | | |
| 7. System prompts user to read and accept "terms of use" | ← | Site Terms of Use form |
| 8. User accepts/denies "terms of use" | | |
| a. If the user denies "terms of use", halt progress, return to step 4 | | |
| b. If the user accepts "terms of use", proceed to step 9 | → | Accept record |
| 9. The system creates a new entry in the User Account DB | → | New user account |
| 10. The system sends a account confirmation email to the email attached to the account | → | email with confirmation link |

| **Postconditions:** |
|---|
| 1. New user account entry in User Accounts DB |
| 2. email with confirmation link sent to appropriate email |

| **Exceptions:** |
|---|
| E1. User does not complete sign-up process (at any point before step 7) |
| 1. System holds entered information temporarily (not including username/password) |
| a. If the user does not return (using same device) to complete sign up within 48 hrs, sign-up attempt is logged and cleared |
| b. If user does return (using same device) to complete sign up within 48 hrs, system autofills form using save attempt, return to step 2 |

| **Alternate Paths:** |
|---|
| |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| User Information | Guest | User Account entry | User Accounts DB |
| Username/password | Guest | Account creation attempt log | U/A creation log |
| Terms of Use form | File Repository | Accepted Terms of Use log | User Account DB |

*Figure 3.2.2. - 1*

# PS-UC2

| **Use Case Name:** Sign into User account | **ID:** PS-UC2 | **Priority:** High |
|---|---|---|
| **Actor:** User | | |
| **Description:** describes the process of signing into a user account | | |
| **Trigger:** login/signin button is pressed | | |
| **Type:** ✖External □ Temporal | | |
| **Preconditions:** | | |
| 1. User has an active, athenticated account | | |
| 2. User has an active network connection | | |
| 3. "Open Road" web application is online | | |

| **Normal Course:** | | **Data Flow for Steps:** |
|---|---|---|
| 1.0 Sign into user account | | |
| 1. System retrieves sign in form | ← | Sign in form |
| 2. User enters username and password | → | Username/Password |
| 3. System checks if username is part of "blacklist" | ← | "Blacklist" query result |
| a. If fail, log attempt alert user and deny access to web application | → | Sign in log |
| b. If pass, proceed to step 4 | | |
| 4. System authenticates username and password | ← | Authorization query result |
| a. If fail, alert user, log attempt return to step 1 | → | Sign in log |
| b. If pass, proceed to step 5 | | |
| 5. System prompts 2-step authorization service to send request | → | 2-Step authorization prompt |
| 6. User receives 2-step authorization request | ← | 2-Step authorization request |
| a. If fail, send deny to system | → | 2-Step authorization response |
| b. If pass, send approval to system | → | 2-Step authorization response |
| 7. System receives 2-step authorization response | ← | |
| a. If fail, proceed to Alternate Path 1.0 | | |
| b. If pass, approve and log sign in attempt | → | Sign in log |
| 8. System verifies account roles and permissions | ← | Roles/Permissions query result |
| 9. System directs client device to appropriate "homepage" | → | Homepage |

| **Postconditions:** |
|---|
| 1. User is signed into their account on the web application |
| 2. User can view pages/data and perform actions authorized by their account roles/permissions |

| **Exceptions:** |
|---|

| **Alternate Paths:** | | |
|---|---|---|
| 1.0 User account falls under "blacklist" | | |
| 1. Sign in attempt is denied | | |
| 2. System logs sign in attempt | → | Sign in log |
| 3. System alerts user to contact customer service for more help | ← | Pop-up alert |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| Username/Password | User | Homepage | Client Device |
| "Blacklilst" query result | User Account DB | Sign in log | Log file |
| Account roles/permissions | User Account DB | | |
| 2-Step Auth. Result | 2-Step Auth. Provider | | |

*Figure 3.2.2. - 2*

## PS-UC3

| **Use Case Name:** Edit account information | | **ID: PS-UC3** | **Priority:** Med |
|---|---|---|---|
| **Actor:** User | | | |
| **Description:** User wants to update online account information (i.e., address, phone number, etc.) | | | |
| **Trigger:** User presses "edit" button in account details | | | |
| **Type:** ☒External ☐ Temporal | | | |
| **Preconditions:** | | | |
| 1. User has active network connection | | | |
| 2. User is signed in to web application | | | |
| 3. User has appropriate permissions to edit account details | | | |
| 4. "Open Road" web application is online | | | |
| | | | |
| **Normal Course:** | | **Data Flow for Steps:** | |
| 1.0 Edit account information | | | |
| 1. Authenticated user selects "edit account" option. | ← | Edit input | |
| 2. The system displays account information as editable fields filled with current info. | → | New account information | |
| 3. The user inputs new/edited data in appropriate fields. | ← | Account information | |
| 4. The user commits changes by clicking "submit". | ← | Submit input | |
| a. All information is acceptable, proceed to next step. | | | |
| b. Any one piece of information is incorrect or unacceptable. (return to step 3 with error notification) | | | |
| 5. The system stores the edited information in the user account data store. | → | Edited account information | |
| **Postconditions:** | | | |
| 1. Any piece of editable account information may be changed permanently. | | | |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| Edit input | User | New account information | User information form |
| Account information | User | Edited account information | User account database |
| Submit input | User | | |

*Figure 3.2.2. - 3*

# PS-UC4

| **Use Case Name:** Log/sign out of online account | **ID:** PS-UC4 | **Priority:** High |
|---|---|---|
| **Actor:** User | | |
| **Description:** describes the process of logging/signing out of an online user account | | |
| **Trigger:** User presses "log/sign out" button | | |
| **Type:** ☒ External ☐ Temporal | | |
| **Preconditions:** | | |
| 1. User has an active network connection | | |
| 2. User is currently logged/signed in to account | | |
| 3. "Open Road" web application is online | | |

| **Normal Course:** | **Data Flow for Steps:** | |
|---|---|---|
| 1.0 Log/Sign out of online account | | |
| 1. System prompts to confirm if the user wants to log/sign out | → | Log/Sign out prompt |
| a. If user rejects, session continues | ← | User response |
| b. If user accepts, proceed to step 2 | ← | User response |
| 2. system ends and invalidates current session | → | Session closure |

| **Postconditions:** |
|---|
| 1. User has been logged/signed out of current session |
| 2. System has closed and invalidated current session |
| 3. User is presented with new log/sign in screen |

| **Exceptions:** |
|---|

| **Alternate Paths:** |
|---|
| 1.0 User closes web browser before logging/signing out of account or is inactive |
| System checks current time against time of last client request (frequent intervals) |
| a. If "x" amount of time has not passed, do nothing |
| b. If "x" amount of time has passed, end and invalidate session |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| User response | User | log/sign out promp | Client device |
| | | Session Closure | System |

*Figure 3.2.2. - 4*

# PS-UC5.1

| **Use Case Name:** Pre-register for scheduled course | **ID:** PS-UC5.1 | **Priority:** High |
|---|---|---|
| **Actor:** Guest | | |
| **Description:** description of process to pre-register for a scheduled course | | |
| **Trigger:** Guest presses "Register" button | | |
| **Type:** ■External □ Temporal | | |
| **Preconditions:** | | |
| 1. Guest has an active network connection | | |
| 2. Selected course has a seat available | | |
| 3. "Open Road" web application is online | | |

| **Normal Course:** | **Data Flow for Steps:** |
|---|---|
| 1.0 Pre-register for scheduled course | |
| 1.System directs client device to "Student Information" form | |
| 2.Guest fills out form and presses "submit" | |
| 3.System checks validity of information on form | |
| a. If fail, return to step 2 with error message | |
| b. If pass, proceed to step 4 | ← Student Information |
| 4.System directs client device to "Select Course" form | |
| 5.Guest selects scheduled course and presses "submit" | ← Course Selection |
| 6.System prompts Guest to confirm selection | |
| a. If fail, return to step 5 with error message | |
| b. If pass, proceed to step 7 | |
| 7.System creates new entry in "Temporary Student Registration" table | |
| 8.System updgrades Guest account to Student account | → New Student Entry |

| **Postconditions:** |
|---|
| 1. Guest has authenticated online account |
| 2. Guest has pre-registered for a specified course |
| 3. "Temporary Student Registration" has a new entry |
| 4. Guest account is upgraded to Student account |

| **Exceptions:** |
|---|

| **Alternate Paths:** |
|---|

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| Student Information | Guest | New Entry | Course Details DB |
| Course Selection | Guest | | |

*Figure 3.2.2. - 5*

# PS-UC5.2

| **Use Case Name:** Pay and Confirm Registration | **ID: PS-UC5.2** | **Priority:** High |
|---|---|---|
| **Actor: Student** | | |
| **Description:** Description of the process by which a student confirms registration. | | |
| **Trigger:** Student presses "pay & confirm" button. | | |
| **Type: x** External ☐ Temporal | | |
| **Preconditions:** | | |
| 1. A student must be connected to the internet, authenticated, and able to execute the process. | | |
| 2. A student has pre-registered for a specific course. | | |
| 3. Web portal is online & accessible. | | |
| 4. Course details databse is online and accessible. | | |
| 5. User account information database is online and accessible. | | |
| **Normal Course:** | **Data Flow for Steps:** | |
| 1.0 Pay for and finalize registration. | | |
| 1. Student selects the "finalize registration" button on the appropriate course. | ← Registration confirmation | |
| 2. Student selects a payment method. | ← Payment type | |
| a. Student opts to pay in-person. (proceed to step 4) | | |
| b. Student opts to pay online. (proceed to step 3) | | |
| 3. Student inputs payment information. | ← Payment details | |
| a. Payment information is incorrect. (return to step 3 with error message) | | |
| b. Payment information is correct. (proceed to next step) | | |
| 4. A PPS employee processes payment and changes course status to "registered". (in-person payment only | → Registration status | |
| 5. The system adds the student to the specific courses list of registered students. | → Student information | |
| 6. The student is sent a confirmation email outlining registered course specifics. | → Course information | |
| **Postconditions:** | | |
| 1. A students info is added to a courses details as a registered student. | | |
| 2. A student is registered to attend a class and has any pertinent information. | | |
| 3. Payment has been received for attending a specific course. | | |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| Registration confirmation | Student | Registration status | Student account database |
| Payment type | Student | Student information | Course details database |
| Payment details | Student | Course information | Student email address |

*Figure 3.2.2. – 6*

# PS-UC6

| **Use Case Name:** Create New Course Type | **ID:** PS-UC6 | **Priority: High** |
|---|---|---|
| **Actor:** PRA Management | | |
| **Description:** This use case describes how PRA Management can create a new type of course | | |
| **Trigger:** Create New Course Type button is pressed | | |
| **Type:** ☒External ☐ Temporal | | |
| **Preconditions:**<br>- Management is authenticated by logging in to their account<br>- The type of course being created does not yet exist<br>- User has a network connection<br>- Web portal is online and running | | |

| **Normal Course:** | **Data Flow for Steps:** |
|---|---|
| 1.0 Create New Course Type | |
| 1. PRA Management selects option to Create New Course Type | |
| 2. PRA Management inputs data for new course type | → "Create Course Type" form |
| 3. PRA Management uploads all files needed for new course type | → Course Files |
| 4. System checks input data against current course types | |
| 5. System returns acceptance/rejection of new course type based on current ones | ← Acceptance/Rejection Message |
| 6. PRA Management presses the Finish button | |

**Postconditions:**
- A new course type has been created or an error message has been shown

**Exceptions:**
E1. New course type info matches an existing course type
1. An error message is shown, asking PRA Management if they would like to revise info or quit the process
   a. If PRA Management chooses to revise info, return to step 3
   b. If PRA Management chooses to quit the process, the form is cleared

**Alternate Paths:**

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| Create Course Type form | PRA Management | Acceptance/Rejection message | PRA Management |
| Course Files | PRA Management | New Course Type entry | Course DB |
| | | | |

*Figure 3.2.2. - 7*

# PS-UC7

| **Use Case Name:** Update Course Details | **ID:** PS-UC 7 | **Priority: High** |
|---|---|---|
| **Actor:** PRA Management | | |
| **Description:** This use case describes how PRA Management can update course details | | |
| **Trigger:** Update Course Details button is pressed | | |
| **Type:** ✖External □ Temporal | | |
| **Preconditions:**<br>- Management is authenticated by logging in to their account<br>- A course exists to be edited<br>- User has a network connection<br>- PRA Web Portal is online and running | | |
| **Normal Course:** | **Data Flow for Steps:** | |
| 1.0 Update Course Details | | |
| 1. PRA Management selects option to update details on a course | | |
| 2. PRA Management selects course to be updated | | |
| 3. PRA Management selects details to be changed and inputs new details | → New course info | |
| 4. PRA Management uploads any new files associated with the course | → Course Files | |
| 5. System checks updated data for inconsistencies | | |
| 6. System returns Acceptance/Rejection of updated course details | ← Acceptance/Rejection message | |
| 7. PRA Management presses the Finish button | | |
| **Postconditions:**<br>- Course details have been updated, or an error message has been shown | | |
| **Exceptions:**<br>E1. Updated course matches details for existing course<br>1. An error message is shown, asking PRA Management if they would like to revise info or quit the process<br>a. If PRA Management chooses to revise info, return to step 4<br>b. If PRA Management chooses to quit the process, the data is cleared | | |
| **Alternate Paths:** | | |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| New Course Info | PRA Management | Acceptance/Rejection message | PRA Management |
| Course Files | PRA Management | Updated Course Info | Course DB |

*Figure 3.2.2. - 8*

## PS-UC8

| Use Case Name: Remove Course from System | ID: PS-UC8 | Priority: High |
|---|---|---|
| **Actor:** PRA Management | | |
| **Description:** This use case describes the process of removing a course from the system | | |
| **Trigger:** Remove Course button is pressed | | |
| **Type:** ☒ External ☐ Temporal | | |
| **Preconditions:**<br>- Management is authenticated by logging in<br>- PRA Web Portal is online and running<br>- A course exists in the system<br>- User has a network connection | | |
| **Normal Course:** | **Data Flow for Steps:** | |
| 1.0 Remove Course from System | | |
| 1. PRA Management selects option to remove a course from the system | | |
| 2. PRA Management selects which course they would like to remove | → Course info to be removed | |
| 3. System asks for confirmation for course removal | ← Confirmation Request | |
| 4. PRA Management selects Confirm | → Confirmation | |
| 5. System removes course info from the system | | |
| **Postconditions:**<br>- Selected course has been removed from the system | | |
| **Exceptions:**<br>E1. PRA Management chooses to cancel process (at any step before step 6)<br>1. PRA Management presses the "Cancel" option<br>2. System asks for confirmation of cancel<br>a. If PRA Management confirms the cancel, the process is cancelled and the course data will continue to be saved in the system<br>b. if PRA Management does not confirm the cancel, the removal process is continued | | |
| **Alternate Paths:** | | |

| Summary Inputs | Source | Summary Outputs | Destination |
|---|---|---|---|
| Course info to be removed | PRA | | |

*Figure 3.2.2. - 9*

# PS-UC9

| **Use Case Name:** Schedule a Course | **ID:** PS-UC9 | **Priority: High** |
|---|---|---|
| **Actor:** PRA Management | | |
| **Description:** This use case describes how PRA Management can schedule a course | | |
| **Trigger:** Schedule a Course button is pressed | | |
| **Type:** ✖ External □ Temporal | | |
| **Preconditions:** | | |
| - PRA Management is authenticated and logged in to their account | | |
| - PRA Web Portal is online and running | | |
| - User has a network connection | | |
| **Normal Course:** | **Data Flow for Steps:** | |
| 1.0 Schedule a Course | | |
| 1. PRA Management selects the option to schedule a course | | |
| 2. System returns a list of courses waiting to be scheduled | ← Available course list | |
| 3. PRA Management selects which course to schedule | → Selected Course | |
| 4. System returns a list of available dates to schedule the course | ← Available date list | |
| 5. PRA Management selects a date to schedule the course | → Selected Date | |
| 6. PRA Management presses the "Schedule Course" button | | |
| 7. System runs a check against all current course, student, and instructor schedules | | |
| a. If scheduling conflict occurs, display conflict location and return to step 3 | | |
| b. If no scheduling conflict occurs, continue to step 8 | | |
| 8. System returns a confirmation/rejection message | ← Confirmation/Rejection message | |
| **Postconditions:** | | |
| - Course has been scheduled for the desired weekend with the desired students | | |
| **Exceptions:** | | |
| **Alternate Paths:** | | |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| Selected Date | PRA Management | Available Course List | Course DB |
| Selected Course | PRA Management | Available Date List | Course DB |

*Figure 3.2.2. - 10*

## PS-UC10

| Use Case Name: Create Specialized User Accounts | ID: PS-UC10 | Priority: High |
|---|---|---|
| **Actor: Management** | | |
| **Description: Describes how management creates non general user accounts such as Admin, Instructor, etc.** | | |
| **Trigger: Staff member of some category is hired and must have an account in the system with appropriate priviliges.** | | |
| **Type: x** External ☐ Temporal | | |
| **Preconditions:** | | |
| 1) A user with Admin permissions must be connected to the internet, authenticated and able to execute the process. | | |
| 2) A new employee with unique information must exist. | | |
| 3) User databases are online and accessible. | | |
| 4) Web portal is online and accessible. | | |
| **Normal Course:** | **Data Flow for Steps:** | |
| 1.0 Create specialized user account for new employee based on position. | | |
| 1. Management creates a user based on personal information, with placeholder password. | ← Employee information | |
| 2. Management applies appropriate types and roles based on employee position. | ← Employment type<br>↘ Employee account information | |
| 5. Employee logs in for the first time with new account. | ← Employee account information | |
| 6. System authenticates login request, recognizing the employees first time login. | ← First time login flag | |
| 7. System sends two step authentication link via email. | → Authentication email | |
| 8. User clicks authentication link and their device is verified for X amount of time. | → Verification flag | |
| 9. System Prompts user to change password. | → Password change prompt | |
| 10. User Enters new password. | ← New password | |
| **Postconditions:** | | |
| 1. A new employee account has been created and stored in the system. | | |
| 2. The new employee has their account login information. | | |

| Summary Inputs | Source | Summary Outputs | Destination |
|---|---|---|---|
| Employee information | Employment documentation | Employee account information | Account database |
| Employment type | Employment documentation | Password change prompt | Employee access point |
| Employee account information | Employee | Authentication email | Employee email client |
| First time login flag | User account database | Verification flag | User account database |
| New password | Employee | | |

*Figure 3.2.2. - 11*

# PS-UC11

| Use Case Name: Update User Account Permissions | ID: PS-UC11 | Priority: High |
|---|---|---|
| **Actor: Management** | | |
| **Description: Describes how management updates the roles associated with employee accounts to change permissions and resource access.** | | |
| **Trigger: An employee has had new responsibilities added and/or removed from their position.** | | |
| **Type: x** External ☐ Temporal | | |
| **Preconditions:** | | |
| 1. A user with Admin permissions must be connected to the internet, authenticated, and able to execute the process. | | |
| 2. An employee account exists. | | |
| 3. User databases are online and accessible. | | |
| 4. Web portal is online and accessible. | | |
| **Normal Course:** | **Data Flow for Steps:** | |
| 1.0 Management removes or adds employee account roles to reduce or enhance permissions. | | |
| 1. Management searches the system for a specific employee account. | ← Search query | |
| 2. Specified user account information with current and eligible roles are displayed. | → Employee account role | |
| 3. The appropriate role is selected and either added to or removed from the account. | ← Employee account role | |
| | | |
| **Alternative Course:** | | |
| 1.1 The user account type must be adjusted to be eligible for a requested role. | | |
| 1. Current and eligible account types are displayed. | → Employee account type | |
| 2. The appropriate account type is selected and applied. | ← Employee account type | |
| 3. Return to Normal Course (step 3). | | |
| **Postconditions:** | | |
| 1. One or more account roles have been added and/or removed from an employee account. | | |
| 2. An employees account type may be adjusted. | | |

| Summary Inputs | Source | Summary Outputs | Destination |
|---|---|---|---|
| Search query | Management | Employee account role | Management |
| Employee account role | User account database | Employee account type | Management |
| Employee account type | User account database | | |

*Figure 3.2.2. - 12*

# PS-UC12

| Use Case Name: Remove User Accounts | ID: PS-UC12 | Priority: High |
|---|---|---|
| **Actor: Management** | | |
| **Description: Describes how a user account of any type is removed from the system by management.** | | |
| **Trigger: A user account of any type has been deemed redundant or threatening.** | | |
| **Type: x** External ☐ Temporal | | |
| **Preconditions:** | | |
| 1. A threatening or redundant account has been identified and exists on the system. | | |
| 2. A user with Admin permissions must be connected to the internet, authenticated, and able to execute the removal process. | | |
| 3. User databases are online and accessible. | | |
| 4. Web portal is online and accessible. | | |
| **Normal Course:** | **Data Flow for Steps:** | |
| 1.0 Managent searches the system for a user account which is deleted. | | |
| 1. Management searches the system for a specific employee account. | ← Search query | |
| 2. Query result is displayed. | → Query result | |
| a. Matching users are displayed. | | |
| b. System displays a "no matching user" error page. | | |
| 3. Management selects an option to delete the account. | ← Deletion input | |
| 4. The system requests admin account authentication. | → Authentication request | |
| 5. An admin user enters their information. | ← Admin account credentials | |
| 6. The account is deleted from the system. | → Confirmation of deletion | |
| 7. The system displays an option to blacklist the account. | | |
| a. Management chooses not to blacklist. (end case here) | | |
| b. Management chooses to blacklist. (continue to step 8) | ← User Blacklist Request | |
| 8. The deleted users information is added to the blacklist of users not permitted to recreate their accoun | → User Blacklist Entry | |
| **Postconditions:** | | |
| 1. An account has been deleted from the system. | | |
| 2. A specific user may no longer able to access the system. | | |

| Summary Inputs | Source | Summary Outputs | Destination |
|---|---|---|---|
| Search query | Management | User account information | Management |
| Deletion input | Management | Authentication request | Admin employee |
| User Blacklist Request | Management | User Blacklist Entry | User Blacklist |
| Admin account credentials | Admin employee | Confirmation of deletion | Management |

*Figure 3.2.2. - 13*

# PS-UC13

| **Use Case Name:** Generate Course Statistics Report | **ID:** PS-UC13 | **Priority: High** |
|---|---|---|
| **Actor:** PRA Management | | |
| **Description:** Thise case describes how PRA Management can generate a **Statistics Report for a chosen course** | | |
| **Trigger:** Generate Course Statistics Report button pressed | | |
| **Type:** ✖External □ Temporal | | |
| **Preconditions:**<br>- PRA Management is authenticated and logged in<br>- PRA Web Portal is online and running<br>- User has a network connection<br>- Course exists | | |
| **Normal Course:** | **Data Flow for Steps:** | |
| 1.0 Generate Course Statistics Report | | |
| 1. PRA Management selects the option to generate course report | | |
| 2. PRA Management selects which course to generate a report from | → Chosen course | |
| 3. PRA Management selects which details to show in the report | → Chosen course details | |
| 4. System returns report on course statistics | ← Course Statistics Report | |
| **Postconditions:**<br>- Course Statistics Report has been generated | | |
| **Exceptions:** | | |
| **Alternate Paths:** | | |

| **Summary Inputs** | **Source** | **Summary Outputs** | **Destination** |
|---|---|---|---|
| Chosen Course | PRA Management | Course Statistics Report | Course DB |
| Chosen Details | PRA Management | | |

*Figure 3.2.2. - 14*

## PS-UC14

| Use Case Name: Review Assigned Course Info | | | ID: PS-UC14 | Priority: High |
|---|---|---|---|---|
| **Actor: Instructors** | | | | |
| **Description: Instructors can review student lists, important documents, and schedules for classes assigned to them** | | | | |
| **Trigger: Instructors request to view course info** | | | | |
| **Type:** □ External □ Temporal | | | | |
| **Preconditions:** | | | | |
| 1. Instructor has an active network connection | | | | |
| 2. Instructor has been assigned a course | | | | |
| 3. Instructor is Logged into web portal | | | | |
| **Normal Course:** | | | **Data Flow for Steps:** | |
| 1.0 Review Assigned Course Info | | | | |
| 1. Instructor querys system for course details | → | | Query (Course ID) | |
| 2. System retrieves files and data from Course Details DB | ← | | Query Results | |
| 4. System sends files for instructor to view | ← | | Course Files, Student List, Schedule | |
| a. If instructor wishes to view another course, return to step 1 | | | | |
| b. If instructor does not wish to view another course, exit process | | | | |
| **Postconditions:** | | | | |
| 1. Instructor Views Desired Files | | | | |

| Summary Inputs | Source | Summary Outputs | Destination |
|---|---|---|---|
| Query (Course ID) | Instructor | Query Results | Instructor |
| | | Course Files, Student List, and Schedule | |
| | | | |

*Figure 3.2.2. - 15*

## PS-UC15.1

| Use Case Name: Request Time Off | | ID: PS-UC15.1 | Priority: Medium |
|---|---|---|---|
| **Actor: Instructor** | | | |
| **Description: Instructor can request for time off** | | | |
| **Trigger: Instructor Presses "RTO Request" Button** | | | |
| **Type:** □ External □ Temporal | | | |
| **Preconditions:** | | | |
| 1. Instructor has an active internet connection | | | |
| 2. System is online | | | |
| 3. Instructor is signed into PRA Web Portal | | | |
| **Normal Course:** | | **Data Flow for Steps:** | |
| 1.0 Request Time Off | | | |
| 1. System opens new RTO Form | ← | Time off request form template | |
| 2. Instructor Fills RTO Form | | | |
| 4. Instructor presses Submit button | → | Filled Time off request form | |
| 5. System sends filled form to RTO Pool | ← | Filled Time off request form | |
| **Postconditions:** | | | |
| 1. RTO Request is Added to pool | | | |

| Summary Inputs | Source | Summary Outputs | Destination |
|---|---|---|---|
| Time off request form template | Instructor | Filled time off request form | Management |
| | System | | |

*Figure 3.2.2. - 16*

# PS-UC15.2

| Use Case Name: Approve RTO | | ID: PS-UC15.2 | Priority: Medium |
|---|---|---|---|
| **Actor: Management** | | | |
| **Description: Management Reviews and Approves/Denies RTO Requests** | | | |
| **Trigger: Manager Accesses RTO Pool** | | | |
| **Type:** □ External □ Temporal | | | |
| **Preconditions:** | | | |
| 1. Manager has an active internet connection | | | |
| 2. System is online | | | |
| 3. Manager is signed into PRA Web Portal | | | |
| **Normal Course:** | | **Data Flow for Steps:** | |
| 1.0 Approve RTO | | | |
| 1. System lists RTO Submissions | ← | Query Result (RTO Pool) | |
| 2. Manager Selects RTO Request | → | Query (RTO Request) | |
| 3. Manager Approves/Denies Request | → | Request Status | |
| 4. System removes request from pool | | | |
| a. If management wishes to view another RTO Request, return to step 1 | | | |
| b. if management does not wish to view another RTO Request, exit process | | | |
| 5. Instructor is notified of their Approval/Denial | ← | Request Status | |
| **Postconditions:** | | | |
| 1. RTO Request is approved and deleted from pool | | | |

| Summary Inputs | Source | Summary Outputs | Destination |
|---|---|---|---|
| Query (RTO Request) | Manager | Query Result | Management |
| Request Status | | Request Status | Instructor |
| | | | |

*Figure 3.2.2. - 17*

## PS-UC16

| Use Case Name: Emailing Users | ID: PS-UC16 | Priority: High |
|---|---|---|
| **Actor: Management** | | |
| **Description: Describes how mass and/or targeted emails are sent to current, past, and prospective students.** | | |
| **Trigger: Information that is worth sharing with multiple people or specific groups of people has been identified.** | | |
| **Type: x** External ☐ Temporal | | |
| **Preconditions:** | | |
| 1. Information exists to be delivered. | | |
| 2. A list of users and/or user groups to whom it would be appropriate to deliver the information to exists. | | |
| 3. User databases are online and accessible. | | |
| 4. Web portal is online and accessible. | | |
| 5. Management user is connected to the internet and authenticated to their email account. | | |
| **Normal Course:** | **Data Flow for Steps:** | |
| 1.0 Management queries the system for a list of user email addresses to whom a message is dispersed. | | |
| 1. Management queries the system for a list of specific user accounts and/or user account groups. | ← | Search query |
| 2. The system exports a list of valid email addresses to an email client. | → | Valid user email addresses |
| 3. Management attaches and sends the message to be dispersed. | ← | Email message |
| 4. Email client disperses the message. | → | Email message |
| **Postconditions:** | | |
| 1. A message has been sent to an appropriate group of users with accounts in the system. | | |

| Summary Inputs | Source | Summary Outputs | Destination |
|---|---|---|---|
| Search query | Management | Valid user email addresses | Email dient |
| Email message | Management | Email message | Appropriate users |
| | | | |

*Figure 3.2.2. - 18*

# PS-UC17

| Use Case Name: Email Students | ID: PS-UC17 | Priority: High |
|---|---|---|
| **Actor: Instructors** | | |
| **Description: Instructors can email students about delays or cancellations within their scheduled course** | | |
| **Trigger: Instructor opens message box within web portal** | | |
| **Type:** □ External □ Temporal | | |
| **Preconditions:** | | |
| 1. Instructor has an active network connection | | |
| 2. Instructor has been assigned a course | | |
| 3. Instructor is Logged into web portal | | |
| **Normal Course:** | **Data Flow for Steps:** | |
| 1.0 Email Students | | |
| 1. Instructor selects Group or Individual for the students they wish to message | → Group ID/Invididual Name | |
| 2. System Queries Student Database for Email Addresses | → Query Results (Email List) | |
| 3. System Generates Email template | ← Email Template | |
| 4. Instructor composes email and presses send | | |
| 5. System forwards email to email server | → Email Message | |
| **Postconditions:** | | |
| 1. Selected recipients are sent email | | |

| Summary Inputs | Source | Summary Outputs | Destination |
|---|---|---|---|
| Group ID/Individual Name | Instructor | Email Message | Email Recipients |
| Query Results (Email List) | User Accounts DB | | |
| Email Template | File Repository | | |

*Figure 3.2.2. - 19*

### 3.2.3. Data Flow Diagrams

| Level | Process ID | Process Name |
|---|---|---|
| Level 0 | Process 0 | Core System |
| Level 1 | Process 1 | Manage User Account |
| Level 1 | Process 2 | Manage Training Course |
| Level 2 | Process 2.1 | Manage Course Types |
| Level 2 | Process 2.2 | Manage Course Schedule |
| Level 2 | Process 2.3 | Register for a Course |
| Level 3 | Process 2.3.1 | Pre-Register for Course |
| Level 2 | Process 2.4 | Generate Course Report |
| Level 3 | Process 2.4.1 | Generate Report Query |
| Level 1 | Process 3 | Manage Website Content |
| Level 2 | Process 3.1 | Edit Webpage Content |
| Level 2 | Process 3.2 | Update Advertisement Space |

*Table 3.2.3. - 1*

## Level 0 – Core System

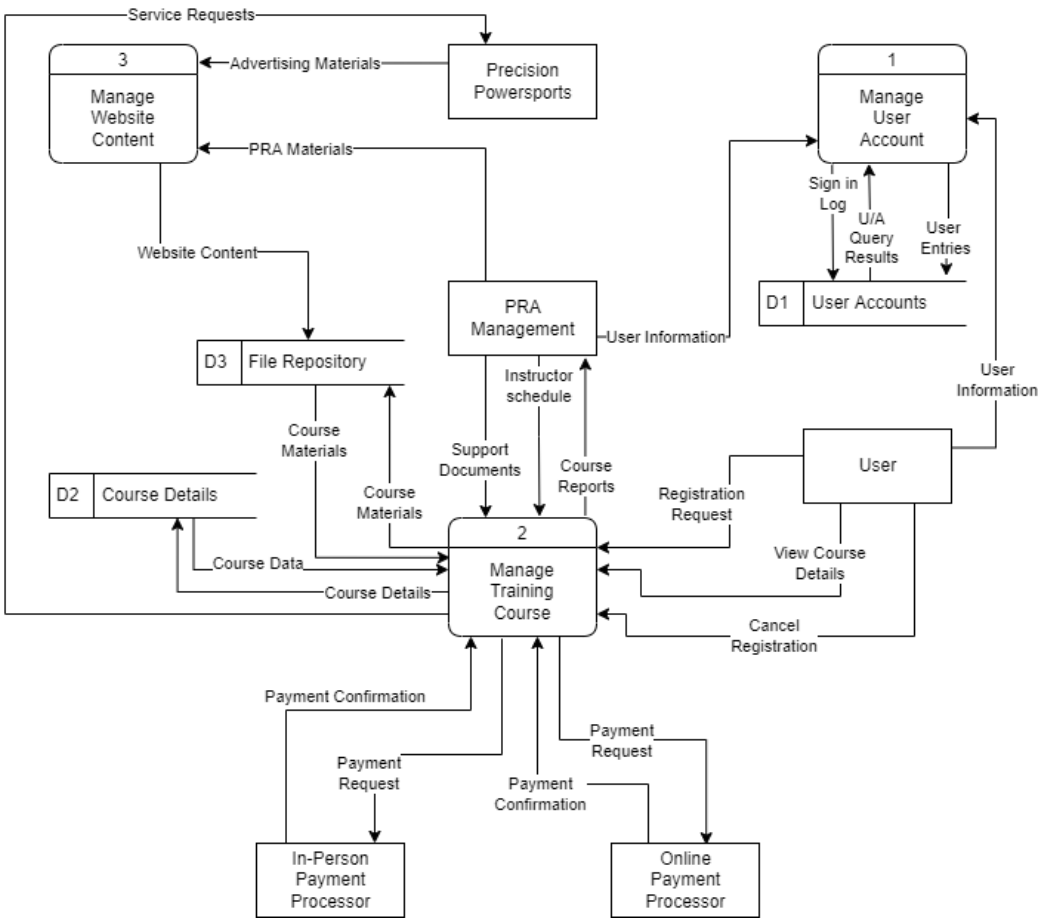



*Figure 3.2.3. - 1*

## Level 1 (Process 1) – Manage User Accounts

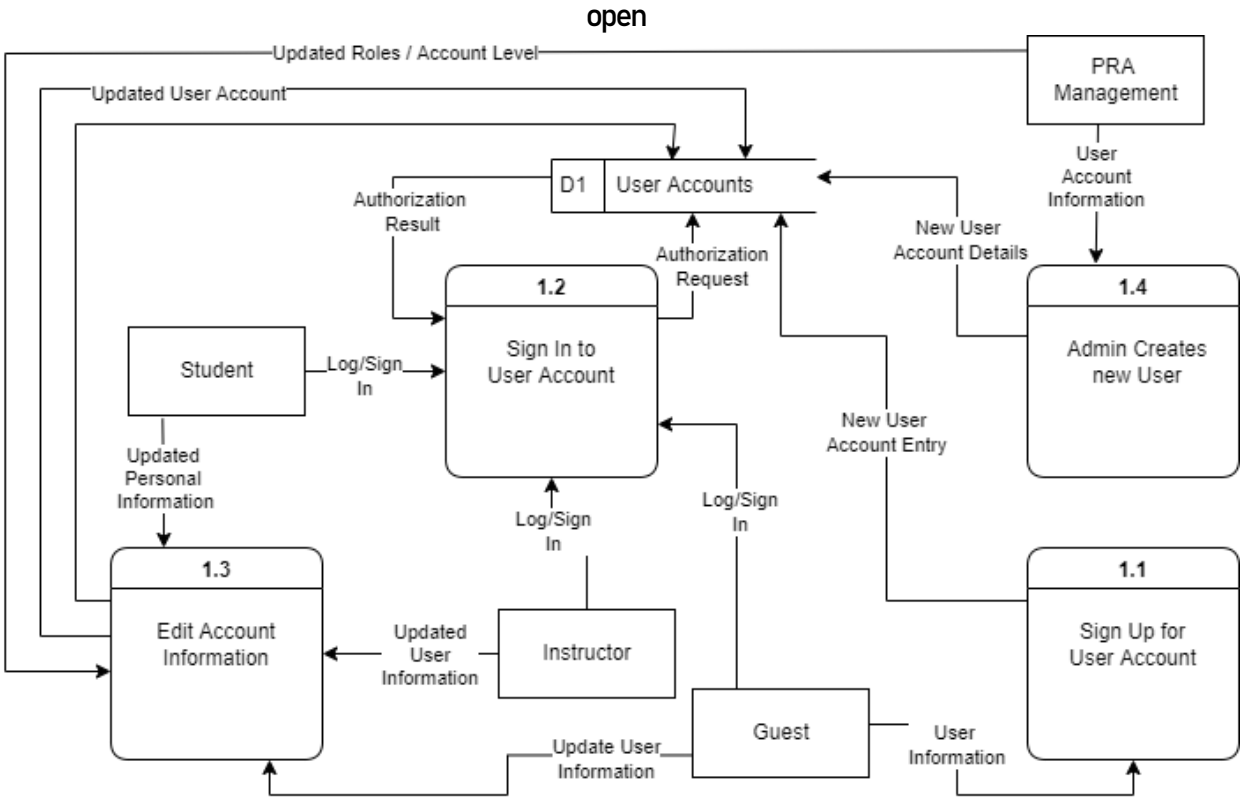



*Figure 3.2.3. - 2*

## Level 1 (Process 2) – Manage Training Courses

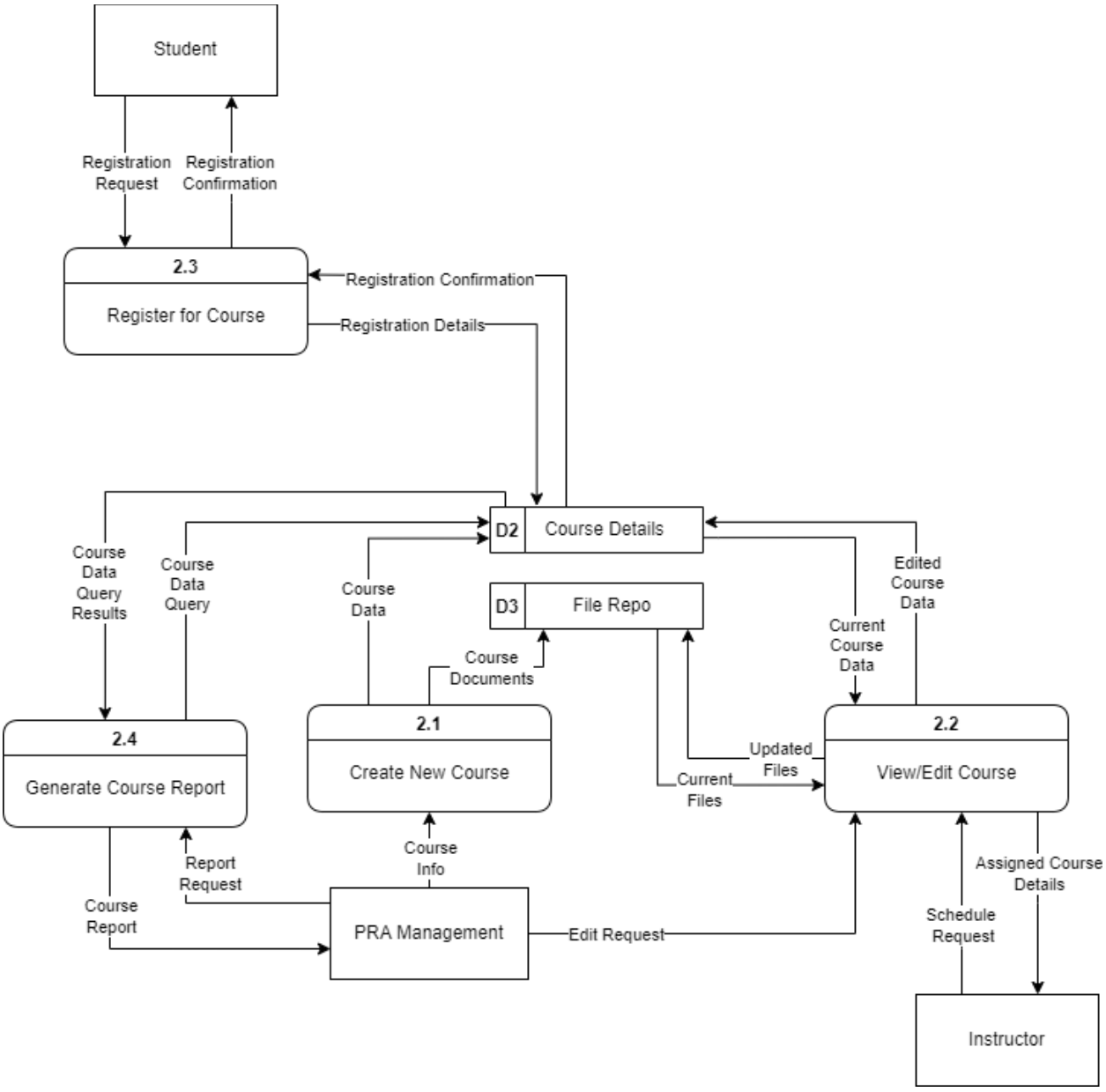



*Figure 3.2.3. - 3*

## Level 2 (Process 2.1) – Manage Course Types

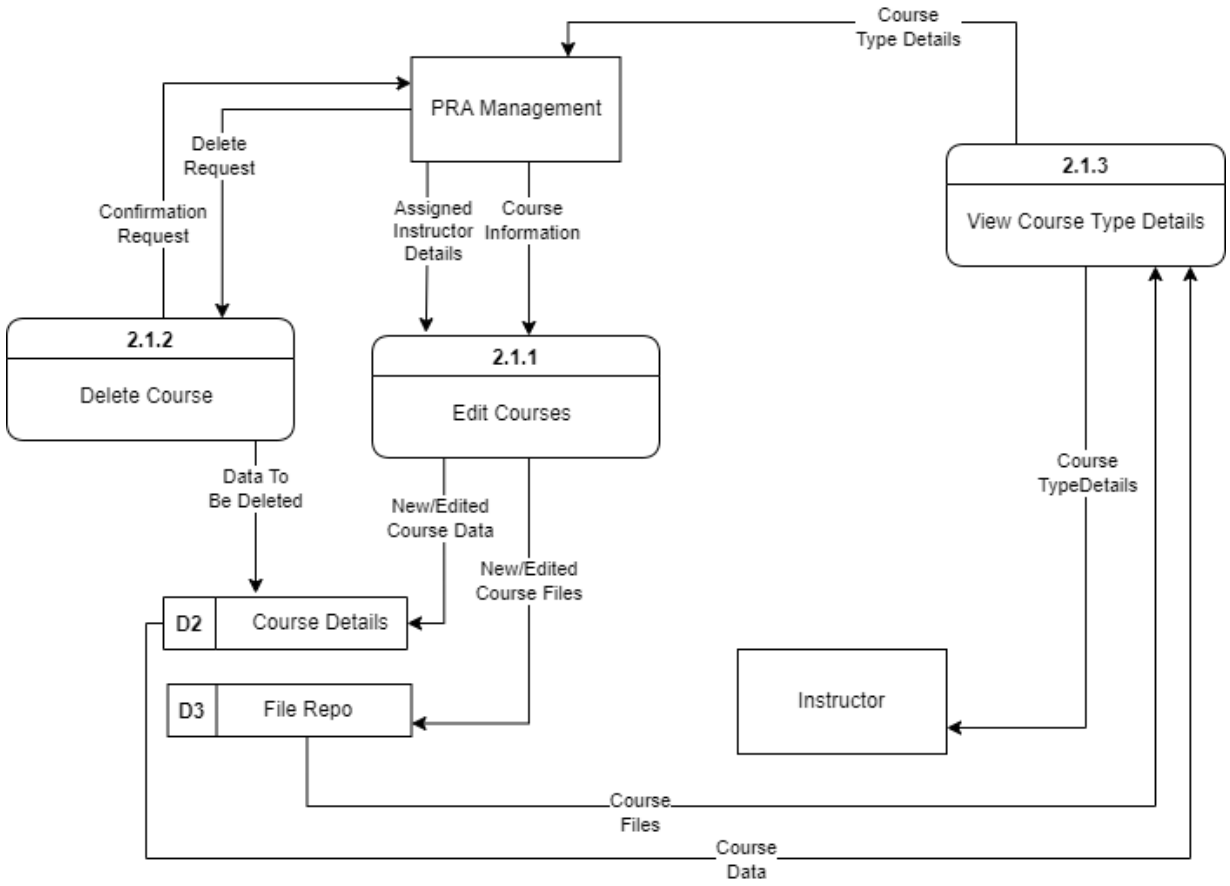



*Figure 3.2.3. - 4*

## Level 2 (Process 2.2) – Schedule a Course

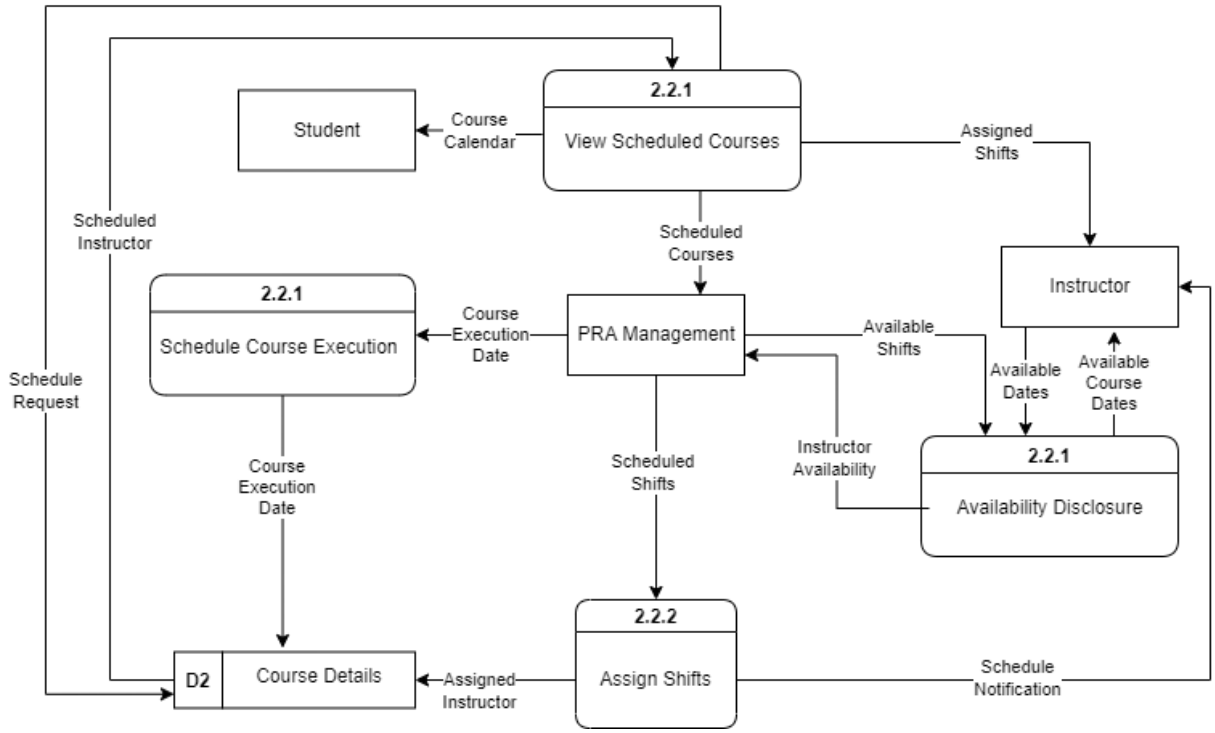



*Figure 3.2.3. - 5*

## Level 2 (Process 2.3) – Register for a Course

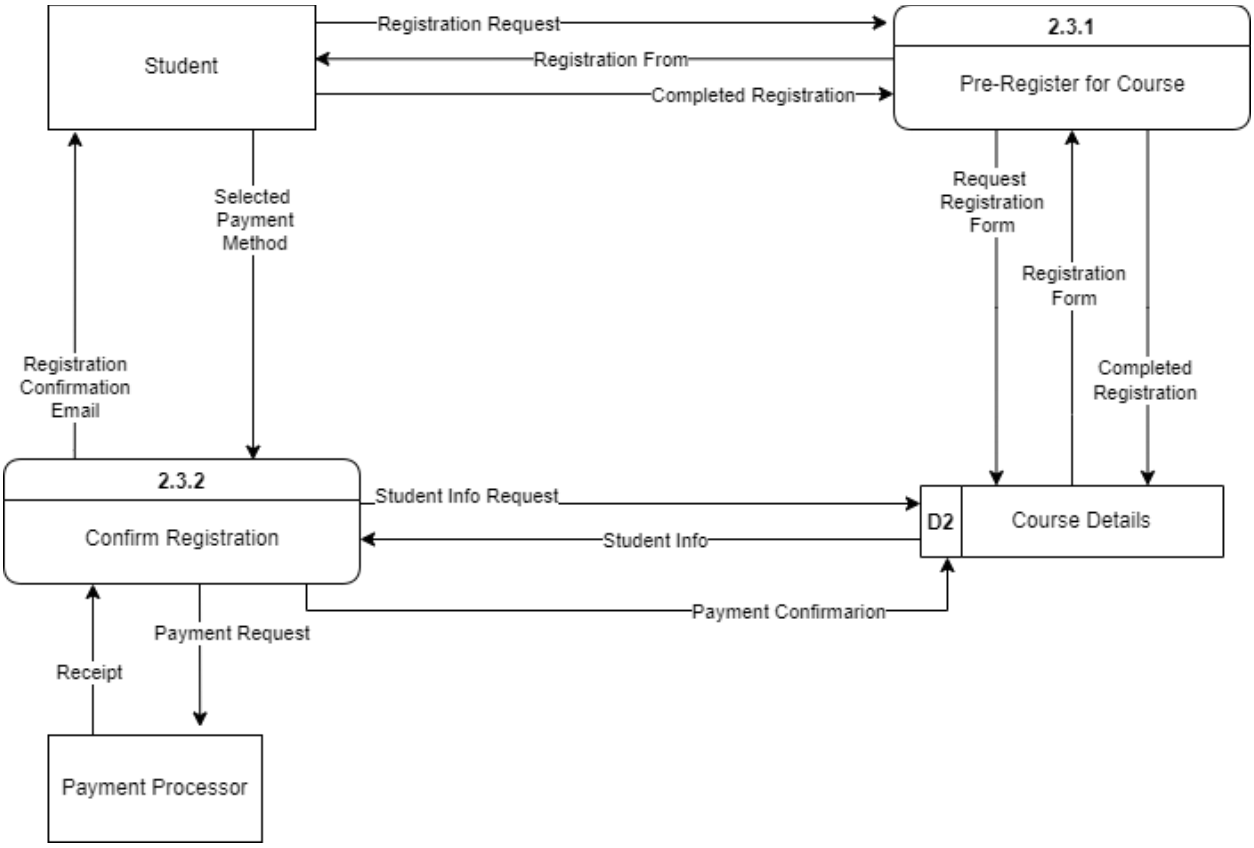



*Figure 3.2.3. – 6*

## Level 3 (Process 2.3.1) – Pre-Register for Course

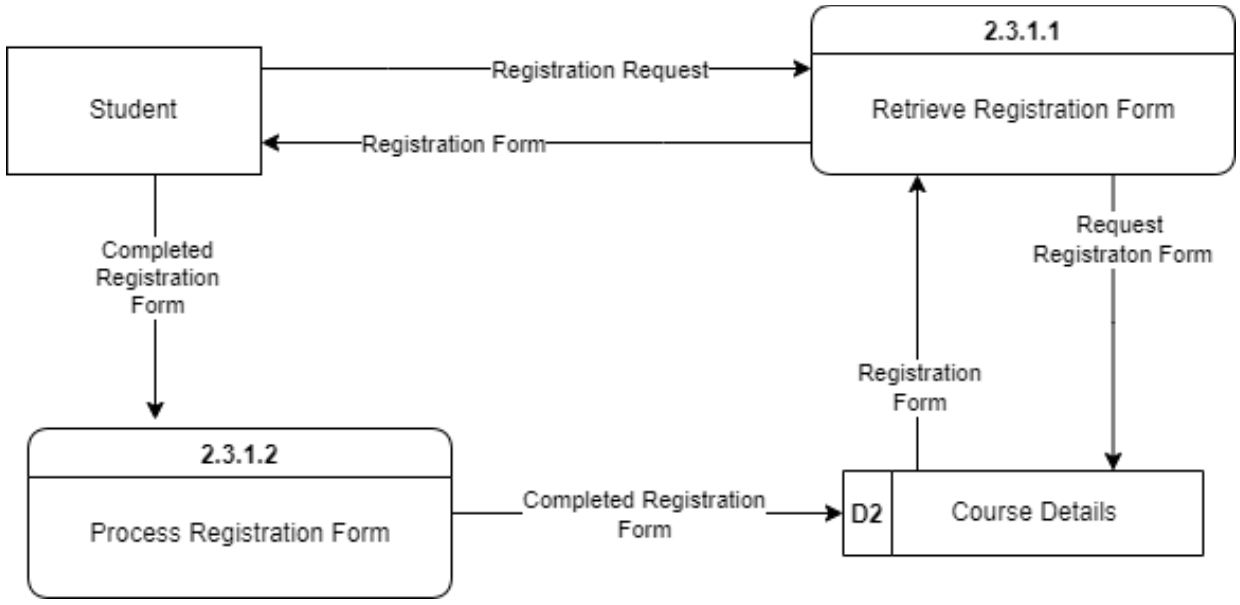



*Figure 3.2.3. - 7*

## Level 2 (Process 2.4) – Generate Course Report

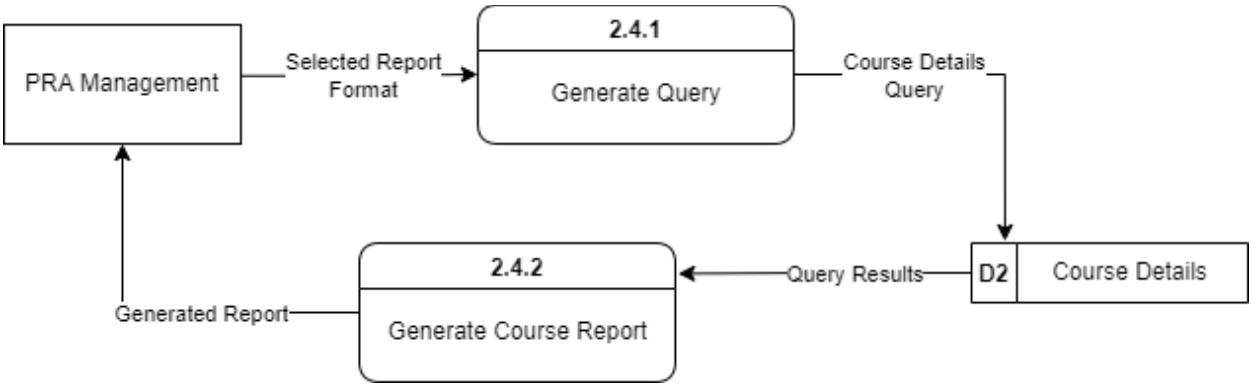



*Figure 3.2.3. - 8*

## Level 3 (Process 2.4.1) – Generate Query

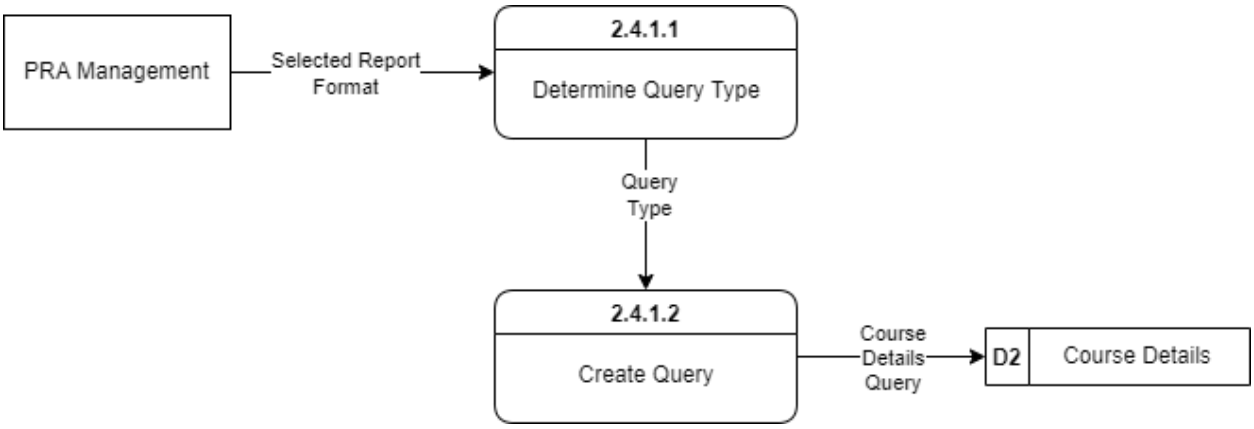



*Figure 3.2.3. - 9*

## Level 1 (Process 3) – Manage Website Content

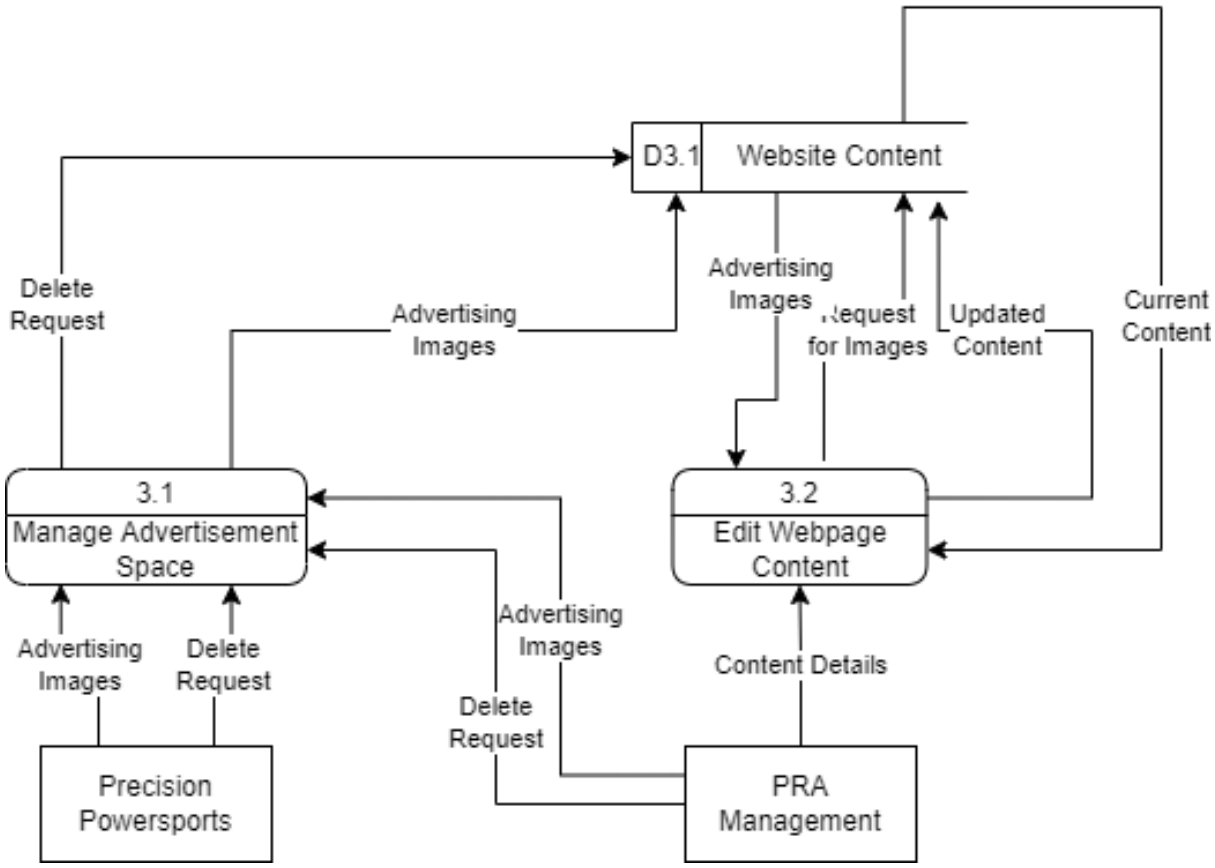



*Figure 3.2.3. - 10*

## Level 2 (Process 3.1) – Edit Webpage Content

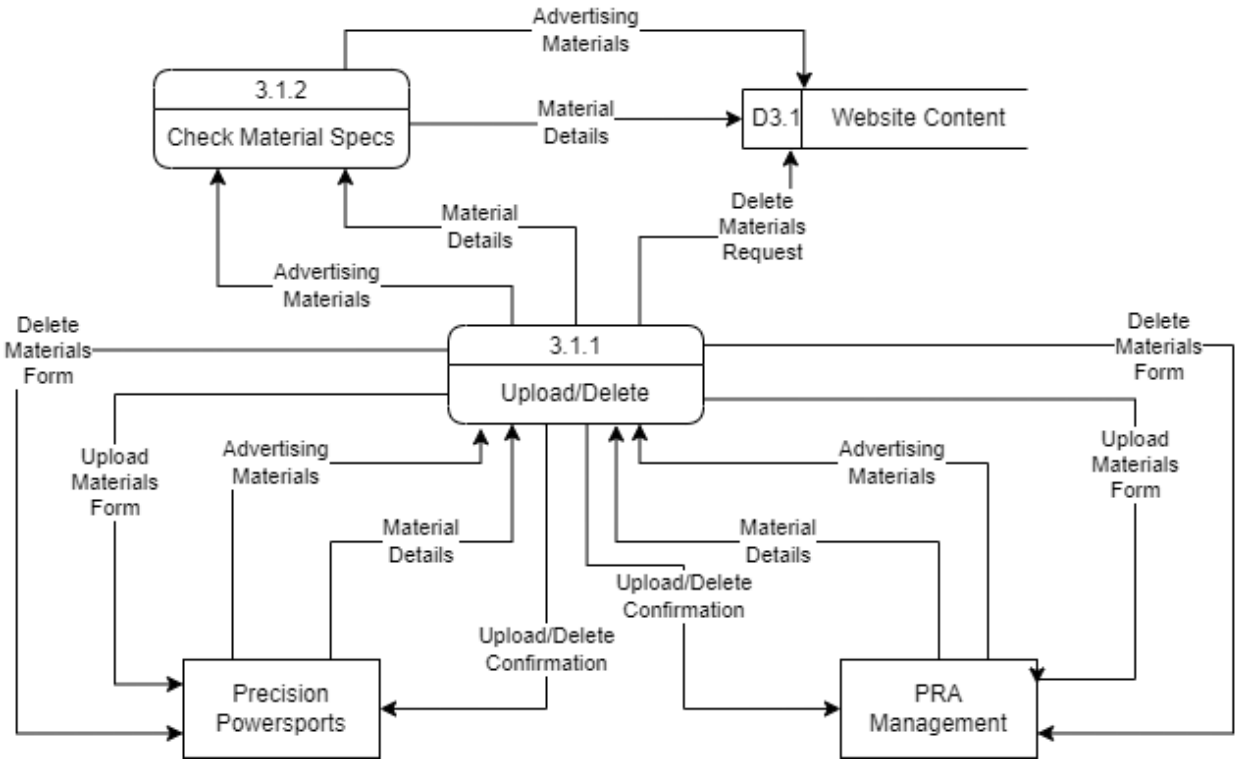



*Figure 3.2.3. - 11*

## Level 2 (Process 3.2) – Manage Advertisement Space

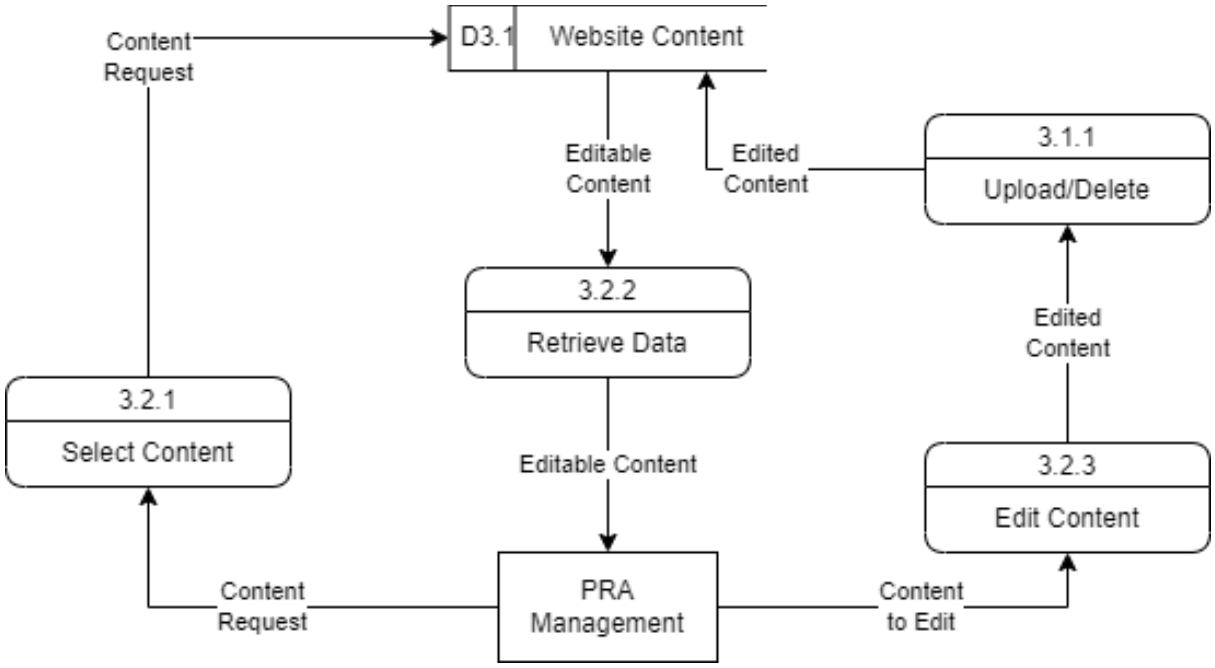



*Figure 3.2.3. - 12*

## Conclusion

After analyzing both the current and proposed systems using the materials above, it can be seen how much more robust the proposed system will be. The Use Cases helped provide a vision of how some of the features might be implemented in the proposed system, while ensuring that any important tasks currently being performed are included in the new processes. The Data Flow Diagrams added on to what the Use Cases showed by revealing what kinds of data are handled by the system and how they will flow throughout the various processes. Together they construct the initial design framework of a system that is not very deep but has a wide reach in its functionality and ability. The current system served its purpose during Precision Riding Academy's inaugural year. Ultimately, it will not be able to support the kind of student flow and business growth that the Academy wishes to see in the near future. SIT SOFTware hopes that project "Open Road" will give the Academy a much strong foundation upon which to build their legacy.

# 4. Milestone Four

## 4.1. System Data Model

The following milestone deliverables focus on the description and justification of a proposed logical model for the database design. Where the data is coming from and how it will be logically stored within the database will be discussed in the following sections.

### 4.1.1. Business Components, Role & Purpose

The following written description discusses relevant business components and instances thereof, accounting for changes to the structure of the business because of implementing our information system. The intention of this description is to accurately analyze and relate the businesses entities, rules, and metadata in a context broad enough to benefit any level of technical ability, yet specific enough to benefit advanced user roles such as administrators and developers in more accurately understanding the proposed system.

Format:

1) The business component/entity being described.
    a. Realistic examples, instances and significant attributes of the component.
        i. Description of the component in the context of our system.

#### 4.1.1.1. Business Components

1) Course:
    a. Beginner, intermediate, and ATV.
        i. Describes the categories from which specific classes can be instantiated, laying out course materials, delivery timeline, fees, and target student level.
2) Payment Details:
    a. Requests for payment, receipts.
        i. Stores information regarding the payment that must be received for a user's registration to be finalized, as well as the information to be sent back to the user as a receipt.
3) Incident:
    a. Physical altercations, collisions, injuries, near misses with potential for serious harm.
        i. Description of events that have happened between one or more students and instructors which may or may not involve a piece of equipment. Incidents provide reporting potential regarding persons and equipment involved, class of occurrence, as well as whether emergency contacts were involved

4) Staff:
    i. Administrators, instructors, store clerks, and developers. Describes all users who are employed by either PRA or PPS and have permissions to interact with the backend of the system, including: scheduling classes, changing course details, generating reports, and approving RTO.
5) Student:
    a. Personal/contact information, security waiver status, and emergency contact details.
        i. All user accounts that are currently registered to take a class or have taken a class in the past, who are not also staff members (who would have access to similar resources and more advanced privileges). While staff can be students, this delineation would be arbitrary within the context of PRA.
6) Guest:
    a. Personal information and the necessary degree of contact details.
        i. Describes user accounts that have been created with a valid email address and minimal personal information but have not yet registered for a class or been onboarded as staff and been assigned appropriate privileges.
7) Login Information:
    a. Usernames, ID, activity status and passwords.
        i. All accounts create an instance of login information, much of which is sensitive data that must be encrypted to ensure privacy of personal information and prevent fraud.
8) Availability:
    a. E1625 (employee ID), 2022-09-18, Approved
        i. Listings of employee's individual dates requested off and their approval status. A manager will communicate to instructors the programs operational weeks within the typical season (typically May to Oct, weather permitting), instructors are required to request weekends off a minimum of two weeks prior, which is when class dates are officially scheduled. Employees are assumed available unless otherwise stated.
9) Transaction Log:
    a. 0987 (user ID), 2023-11-18, 192.168.1.122, 18:32:02,
        i. Records details about authentication attempts to the system, regardless of user type. The log is intended to store enough pertinent information about interactions that they can be used in auditing the site. Logs will be incrementally archived to prevent storage of redundant data over time.

10) Vehicle:
    a. Yam-231 (stock number), 1HG CM82633 A 004352 (VIN)
        i. Describes any vehicles that are used in the execution or preparation of a class, containing information regarding the physical appearance of the vehicle, it's mechanical and manufacturing specifications as well as the type of vehicle it is and whether it is available for use.

4.1.1.2. Bridging Entities

1) Registration:
    a. 22-010 (Class ID), S0102 (Student ID)
        i. Instances of classes to which students have registered. This entity exists to eliminate the relationship whereby many students can take many classes, replacing it with a relationship whereby many classes can be registered with each registration being a single class, with many or one registration instance being linked to single users.
2) Class:
    a. Intermediate motorcycle course taught by Bruce S. on May 10, 2022, using vehicles: X, Y and Z.
        i. An instance of a course that has been scheduled to be carried out during a specific week for which students can register. For the instance to be complete it must list a primary instructor, start and end dates, and may list a secondary instructor as well as vehicles being used.
3) Invoice Item:
    a. Registration & payment ID.
        i. Describes the individual registration components that comprise an instance of payment details, akin to the items within a shopping cart, designed to eliminate data redundancy created by shared payment details for registration instances.
4) Damage Report:
    a. Incident ID & vehicle stock number.
        i. Relates unique incident details to specific vehicles with minimal data redundancy and without jeopardizing sensitive data integrity.

#### 4.1.1.3. Relation of Established Requirements to Proposed System Diagrams

| System Requirement | Entity(s) | Relevant Information |
|---|---|---|
| Online Registration | Registration, class, course | Users' personal info, class instance details |
| Online Payment | invoice_item, payment | Payment info, invoices/receipts, payment status |
| Course Management | Registration, course, class, staff, vehicle | Course details/materials, eligible staff/vehicles, scheduled classes |
| Basic Accounting Reports | class, registration, payment | Payment ratios, course fees, scheduled & executed classes |
| Mass Emailing | student, staff | User account type & status, user personal & contact information |
| Data Security & Integrity | user_login, transaction_log | Strongly encrypted authentication details & personal information |
| Class Scheduling | course, class, registration, staff, student, availability, vehicle | Instructor RTO, course details, registered students, available vehicles, class date & time |
| User Accounts | student, staff | Personal information, account type and details. |

### 4.1.2. Business Rules

The following business rules have been identified through thorough requirement elicitation and analysis by both the SIT SOFTware team and the client. The business rules describe the relationships that exist between the different entities that will exist in the proposed system. The list is based off some assumptions that in and of themselves are not business rules, but affect what data is collected through the various business processes.

- A Course may have zero or more Classes.
- A Class belongs to only one Course.
- A Class may have zero or more Registrations.
- Each Registration belongs to only one Class.
- A Class may have zero or more Incidents.
- An Incident belongs to only one Class.
- A Class must have one Staff assigned.
- A Staff may be assigned to zero or more Classes.
- A Staff may generate zero or more RTOs (Availability).
- Each RTO (Availability) belongs to only one Staff member.
- A Student may Register one or more times.
- Each Registration belongs to only one Student.
- Each Registration may be assigned a Vehicle.
- Each Vehicle may be assigned to zero or more Registrations.
- A Registration will generate only one Invoice Item.
- Each Invoice Item belongs to only one Registration.
- A Payment may involve one or more Invoice Items.
- Each Invoice Item belongs to only one Payment.
- An Incident may generate zero or more Damage Reports.
- Each Damage Report belongs to only one Incident.
- A Vehicle may have zero or more Damage Reports.
- Each Damage Report belongs to only one Vehicle.
- A User Login may belong to a Student or Staff.
- A Student or Staff can have only one User Login.
- Each User Login may have zero or more Transaction Logs.
- Each Transaction Log belongs to only one User Login.

Assumptions:

- Students are at least 16 years of age.
- Students have a valid Alberta Class 5 Drivers License (or equivalent).
- If a student is underage, consent from a parent/legal guardian has been received.
- Parent/legal guardians are not required to have a User Account.

Summary Table

The following table provides an easy-to-read summary of the business described above.

### 4.1.3. Entity Relationship Diagram

The following entity relationship diagram (*figure 4.1.3. – 1*) was made using the Crow's Foot stye notation and illustrates the logical design of the proposed system's database. It shows not only that a relationship may exists between to tables, but also the type of the relationship (i.e., zero to many, one to many, zero or one, one and only one). To facilitate a concise presentation of the information in the diagram, only the attributes that are either a Primary or Foreign table key are shown. All other attributes that have been identified as a required part of the database design are shown later in the data dictionary (section 4.1.4.).

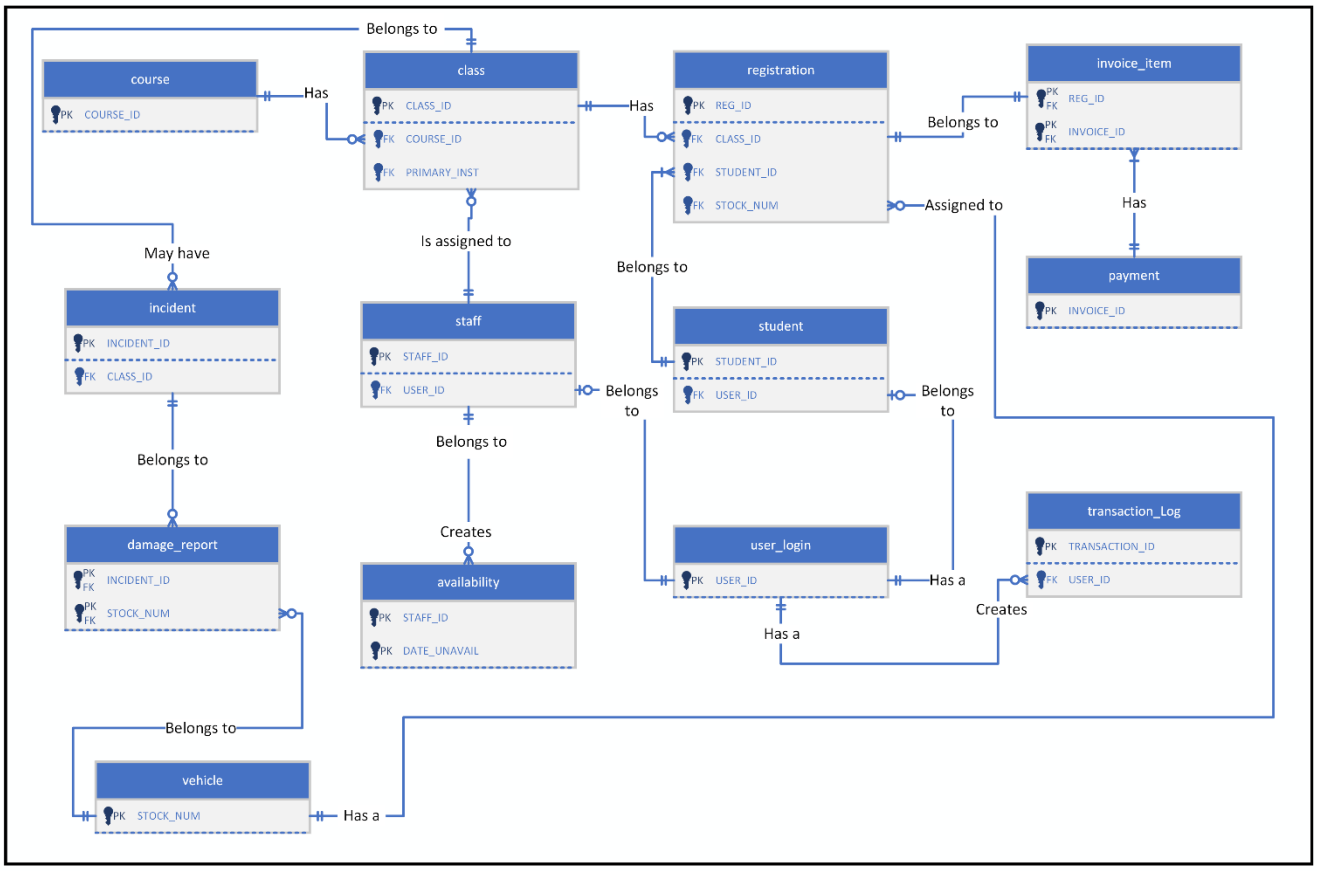



*Figure 4.1.3. - 1*

### 4.1.4. Data Dictionary

The following data dictionary was constructed using information collected through regular team discussions and client meetings. It describes the metadata (data about data) of the database design. The dictionary will be presented in a landscape layout fashion and split up among multiple pages to ensure readability. Before presenting the data dictionary, it would be helpful to review the meaning of each column used to organize the information.

Table Name

This column allows for each of the groups of attributes to be organized based on which table they belong to.

Attribute Name

This column provides the name for each attribute of each table. It is a best practice in data base design to include the name of the table in the attribute name where possible.

Description

This column allows for a short description of the source of the data being held by the attribute.

Data Type

This column describes the functional data type for the data being held in the attribute, i.e., data types such as Char, Varchar, and LongText allow for various types of text data to be stored.

Format/Mask

This column provides information regarding if the data entered should be done so using a specific pattern (i.e., a phone number (999-999-9999).

Required

This column shows if the attribute will be required to have data when the entry is added to the table.

PK or FK

This column shows whether the attribute is a Primary or Foreign key for that table.

FK Ref Table

This column provides information related to which table a Foreign key is referencing.

Sample Data

This column provides an example of how the data might appear in the table.

| Table Name | Attribute Name | Description | Data Type | Format/Mask | Required | PK or FK | FK Ref Table | Sample Data |
|---|---|---|---|---|---|---|---|---|
| user_login | USER_ID | Unique Code to identify users | UnsignedInt(4) | 9999 | Y | PK | | 1234 |
| | USER_PW | Password chosen by the user | Varchar(20) | | Y | | | Abc$123 |
| | USER_NAME | Username chosen by the user | Varchar(20) | | Y | | | BobSmith |
| | USER_STATUS | Whether the user is Active or Inactive | UnsignedInt(1) | 9 | Y | | | 1 |
| student | STUDENT_ID | Unique Code to identify students | Char(5) | X9999 | Y | PK | | S4321 |
| | USER_ID | Unique Code to identify users | UnsignedInt(4) | 9999 | Y | FK | login_info | 1234 |
| | STUDENT_PNAME | Preferred name of the Student | Varchar(20) | | | | | Bobby |
| | STUDENT_FNAME | First name of the Student | Varchar(20) | | Y | | | Bob |
| | STUDENT_LNAME | Last name of the Student | Varchar(20) | | Y | | | Smith |
| | STUDENT_PHONE | Phone number of the Student | Char(12) | 999-999-9999 | Y | | | 403-123-4567 |
| | STUDENT_EMAIL | Email address of the Student | Varchar(40) | | Y | | | bobsmith@bob.com |
| | STUDENT_DOB | Students Date of Birth | Date | yyyy-mm-dd | Y | | | 2/29/1932 |
| | STUDENT_DLN | Students Drivers License Number | Char(10) | 999999-999 | Y | | | 123456-789 |
| | STUDENT_ADDR1 | Line one of the Students address | Varchar(30) | | Y | | | 123 Cherry Lane |
| | STUDENT_ADDR2 | Line two of the Students address | Varchar(30) | | | | | Appt 64 |
| | STUDENT_CITY | City the Student lives in | Varchar(30) | | Y | | | Lethbridge |
| | STUDENT_PR_ST | Province or State the Student lives in | Char(2) | XX | Y | | | AB |
| | STUDENT_COUNTRY | Country the Student lives in | Varchar(30) | | Y | | | Canada |
| | STUDENT_POST_ZIP | Postal or Zip Code of the Student | Varchar(7) | | Y | | | A1B C2D |
| | EMERGCONT_NAME | Name of the Students emergency contact | Varchar(40) | | | | | John Doe |
| | EMERGCONT_PHONE | Phone number of the Students emergency contact | Char(12) | 999-999-9999 | | | | 403-987-6543 |
| | STUDENT_WAIVER | Whether the student has a signed waiver | UnsignedInt(1) | 9 | Y | | | 1 |
| staff | STAFF_ID | Unique Code to identify staff members | Char(5) | X9999 | Y | PK | | E4321 |
| | USER_ID | Unique Code to identify users | UnsignedInt(4) | 9999 | Y | FK | login_info | 1234 |
| | STAFF_LEVEL | Level of site access for the staff member | Char(1) | X | Y | | | A |
| | STAFF_PNAME | Preferred name of the Staff member | Varchar(20) | | | | | Bobby |
| | STAFF_FNAME | First name of the Staff member | Varchar(20) | | Y | | | Bob |
| | STAFF_LNAME | Last name of the Staff member | Varchar(20) | | Y | | | Smith |
| | STAFF_PHONE | Phone number of the Staff member | Char(12) | 999-999-9999 | Y | | | 403-123-4567 |
| | STAFF_EMAIL | Email address of the Staff member | Varchar(40) | | Y | | | bobsmith@bob.com |
| | STAFF_DOB | Staff members Date of Birth | Date | yyyy-mm-dd | Y | | | 2/29/1932 |
| | STAFF_DLN | Staff members Drivers License Number | Char(10) | 999999-999 | Y | | | 123456-789 |
| | STAFF_ADDR1 | Line one of the Staff members address | Varchar(30) | | Y | | | 123 Cherry Lane |
| | STAFF_ADDR2 | Line two of the Staff members address | Varchar(30) | | | | | Appt 64 |
| | STAFF_CITY | City the Staff member lives in | Varchar(30) | | Y | | | Lethbridge |
| | STAFF_PR_ST | Province or State the Staff member lives in | Char(2) | XX | Y | | | AB |
| | STAFF_COUNTRY | Country the Staff member lives in | Varchar(30) | | Y | | | Canada |
| | STAFF_POST_ZIP | Postal or Zip Code of the Staff member | Varchar(7) | | Y | | | A1B C2D |
| | EMERGCONT_NAME | Name of the Staff members emergency contact | Varchar(40) | | | | | John Doe |
| | EMERGCONT_PHONE | Phone number of the Staff members emergency contact | Char(12) | 999-999-9999 | | | | 403-987-6543 |

*Figure 4.1.4. - 1*

| Table Name | Attribute Name | Description | Data Type | Format/Mask | Required | PK or FK | FK Ref Table | Sample Data |
|---|---|---|---|---|---|---|---|---|
| availability | STAFF_ID | Unique Code to identify staff members | Char(5) | X9999 | Y | PK/FK | staff | E4321 |
| | DATE_UNAVAIL | Date the staff member is unavailable to work | Date | yyyy-mm-dd | Y | PK | | 2/29/1932 |
| | APPROVED | Whether the request has been approved | UnsignedInt(1) | 9 | Y | | | 1/0/1900 |
| transaction | TRANSACTION_ID | Autogenerated number | UnsignedInt(5) | 99999 | Y | PK | | 54321 |
| | USER_ID | ID of the user initiating transaction | UnsignedInt(4) | 9999 | Y | FK | user | 1234 |
| | TRANSACTION_DATE | Date the transaction occurred | Date | yyyy-mm-dd | Y | | | 2/29/1932 |
| | TRANSACTION_STIME | Time the user logged in to their account | Time | 99:99:99 | Y | | | 15:26:50 |
| | TRANSACTION_IP_ADDR | IP address of the user initiating transaction | Char(15) | 999.999.999.999 | Y | | | 172.022.002.029 |
| course | COURSE_ID | ID number for the course | Char(3) | 999 | Y | PK | | 101 |
| | COURSE_NAME | Name associated with the course | Varchar(15) | | Y | | | Beginner |
| | COURSE_DOCS | File path to docs related to the course | Varchar(256) | | | | | \Beginner\Docs\ |
| | COURSE_MAX_SEATS | Maximum number of seats for the course | UnsignedInt(1) | 9 | Y | | | 5 |
| | COURSE_FEE | Cost to register for the course | Double(5,2) | 999.99 | Y | | | 259.99 |
| class | CLASS_ID | ID number for the class | Char(6) | yy-999 | Y | PK | | 23-101 |
| | COURSE_ID | Course type of the class | Char(3) | 999 | Y | FK | course | 101 |
| | PRIMARY_INSTRUCTOR_ID | Staff ID of the main instructor | Char(5) | X9999 | Y | FK | staff | E4321 |
| | SECONDARY_INSTRUCTOR_ID | Staff ID of the assistant instructror | Char(5) | X9999 | | FK | staff | E4321 |
| | CLASS_START | Start date of the class | Date | yyyy-mm-dd | Y | | | 5/20/2023 |
| | CLASS_END | End date of the class | Date | yyyy-mm-dd | Y | | | 5/23/2023 |
| registration | REGISTRATION_ID | Autogenerated number | Char(4) | 9999 | Y | PK | | 12/4/1917 |
| | CLASS_ID | ID number for the class | Char(6) | yy-999 | Y | FK | class | 23-101 |
| | STUDENT_ID | ID number for the student | Char(5) | X9999 | Y | FK | student | S1234 |
| | VEHICLE_STOCK_NUM | Vehicle the student is initially assigned | Varchar(17) | XXX-999 | Y | FK | vehicle | YAM-123 |
| | REGISTRATION_CANCELLED | Whether the registration has been cancelled | UnsignedInt(1) | 9 | Y | | | 0 |
| | REGISTRATION_WAITLIST | Whether the registration has been waitlisted | UnsignedInt(1) | 9 | Y | | | 0 |
| payment | INVOICE_ID | ID number of the invoice | UnsignedInt(4) | 9999 | Y | PK | | 5678 |
| | PAYMENT_SUBTOTAL | Subtotal before tax | Double(5,2) | 999.99 | Y | | | 259.99 |
| | PAYMENT_TAX_AMT | Percentage of Sales Tax to be applied | Double(3,2) | 0.99 | Y | | | 0.05 |
| | PAYMENT_TYPE | Method of payment | Varchar(15) | | Y | | | PayPal |
| invoice_item | REGISTRATION_ID | Autogenerated number | Char(4) | 9999 | Y | PK/FK | registration | 12/4/1917 |
| | PAYMENT_ID | ID number of the payment | UnsignedInt(4) | 9999 | Y | PK/FK | payment | 5678 |
| vehicle | VEHICLE_STOCK_NUM | Stock Number of the vehicle | Varchar(17) | XXX-999 | Y | PK | | YAM-123 |
| | VEHICLE_VIN | VIN of the vehicle | Varchar(17) | 9XXXX99XXXX999999 | Y | | | 5YJSA1DG9DFP14705 |
| | VEHICLE_YEAR | Year the vehicle was manufactured | Char(4) | 9999 | Y | | | 2002 |
| | VEHICLE_MAKE | Make of the vehicle | Varchar(20) | | Y | | | Kawasaki |
| | VEHICLE_MODEL | Model of the vehicle | Varchar(40) | | Y | | | Ninja |
| | VEHICLE_ODO | Odemeter reading of the vehicle | Char(10) | 9999999999 | Y | | | 0002564821 |
| | VEHICLE_TYPE | Type of vehicle | Varchar(15) | | Y | | | Motorcycle |
| | VEHICLE_COLOR | Color of the vehicle | Varchar(10) | | | | | Red |
| | VEHICLE_SIZE | Engine displacement of the vehicle | Varchar(4) | 9999 | Y | | | 750 |
| | AVAIL_STATUS | Availability status of the vehicle | UnsignedInt(1) | 9 | Y | | | 1 |

*Figure 4.1.4 - 2*

| Table Name | Attribute Name | Description | Data Type | Format/Mask | Required | PK or FK | FK Ref Table | Sample Data |
|---|---|---|---|---|---|---|---|---|
| incident | INCIDENT_ID | ID# of the incident | UnsignedInt(4) | 9999 | Y | PK | | 1346 |
| | INCIDENT_DATE | Date the incident happened | Date | yyyy-mm-dd | Y | | | 6/17/2023 |
| | CLASS_ID | Registration ID of the student involved | Char(6) | yy-999 | Y | FK | class | 23-101 |
| | INCIDENT_DETAILS | Details of the incident | LongText | | Y | | | They crashed |
| | EMERG_SERV_CALLED | Whether or not an emergency services were called | UnsignedInt(1) | 9 | Y | | | 0 |
| | EMERG_CONTACT_CALLED | Whether or not an emergency contact was called | UnsignedInt(1) | 9 | Y | | | 1 |
| damage_report | INCIDENT_ID | ID# of the incident causing damage | UnsignedInt(4) | 9999 | Y | PK/FK | incident | 1346 |
| | VEHICLE_VIN | VIN of the vehicle involved | Char(4) | 9999 | Y | PK/FK | vehicle | 2002 |

*Figure 4.1.4. - 3*

## 4.2. Conclusion

To conclude this milestone report, it can be easily seen that all the information collected through previous analysis was used or revisited to design this logical model of the proposed system's database. All the data represented here is collected through one or more of the various business processes that make up the proposed system. Further requirement and elicitation and refinement was also performed while creating the database logical model, meaning the proposed system has become more fully understood by all development team members. With the work done during this milestone, the SIT SOFTware team is even more confident in its ability to provide Precision Riding Academy with an information system solution that will meet (if not exceed) their requirements.

# 5. Milestone Five

## 5.1. Feasibility Analysis

### 5.1.1. Technical

Technical feasibility describes the proposed solutions extensibility in the face of changing and growing business needs, the support available to users of the system when faced with inquiries or concerns, as well as the hardware requirements associated with the successful ongoing implementation of the solution. As such, this section of the feasibility assessment is separated into the subcategories: extensibility, available support, and hardware requirements.

#### Extensibility

##### Modular Design

The solution should be as modular as possible, in that it should allow for expansion of the system through the addition of new features or alternate configurations of preexisting components to better meet the businesses needs as they grow into routine operations. This would best be achieved through thorough and available documentation, professional design and implementation standards, and by prioritizing the use of universally accessible services and extensions.

##### Back-end Accessibility

Priority has been placed on the back end and other elements of the system which are typically abstracted from the user being accessible and customizable by specific PRA staff members. This concern is paramount to their ability to expand the academy as it matures and would be best met by accessible source code that could be modified by in-house or outsourced designers. Well documented and understandable design practices as well as readily available and commonly used development tools and resources are a necessity for the solution.

##### Scalability of System & Infrastructure

The systems' overall ability to grow over the years regarding the number of student, guest, and instructor accounts, as well as course variety supported concurrently without catastrophic business or system failure. This concern is best met using reliable, well-established services and tools that can meet more than current minimum business needs with the ability to increase or reduce allocated resources as needed. The businesses financial, time and resource commitment should all be minimal regarding changes to physical infrastructure when implementing modifications to the system.

To best aid in PRA's ability to grow in the future, all three concerns must be addressed. This would allow the Academy to continually customize an increasingly unique system that meets their exact needs without becoming an unwieldly and difficult to understand mess. With the company hoping to grow in multiple regards the variable nature of their needs is extended to their concurrent user expectations which should be able to comfortably fluctuate.

## Available Support

### Submitting/ Receiving responses to inquiries

Response time will be important to the client as keeping the system running will be incredibly important. The faster a response is received the better for our client. Even if its basic questions that don't affect the website being online the faster a response is received the happier our client will be.

### Overall ability to deal with potential issues

Our server choice will need to consider the ability of the service to deal with potential issues. As we will be handing the system over directly to the client when we are done, the skills, and willingness of the web hosting team will need to be considered. This is especially important as our client has had issues with web hosting companies in the past.

### Time required for a resolution

The faster potential issues can be resolved the better. Our client needs the website to stay online as much as possible for their business to function. Any issue that comes up will need to be dealt with in a timely manner.

Overall available support will be important to our client as the faster support is received, and the quality of that support will produce a better experience for our customer post launch. Our customers' experience with the available support will be a significant factor in our webhosting suggestions.

## Hardware Requirements

### Physical System requirements

The physical system requirements are important to look for in our system as the more our system can do with less the better. This should keep service requests down and lead to a more robust system overall. The less taxing our system is on our hardware the faster customers can access the website and sign up for courses.

### Availability of the system

The higher our systems' uptime the better for our client. A low uptime could lead to potential student disenfranchisement. Our system Ideally needs to be there for every potential student or staff member when they try to access it. If this is not possible it needs to be there as often as humanly possible to minimize this issue.

### Access To Hardware

Access to physical system resources such as hard drives that might contain sensitive student or staff information should be effectively restricted by reasonable physical precautions. Insurance must also be provided that these resources are protected from internal malicious intent, whether from bad actors within the academy itself or an external service provider. Appropriate solutions would include encryption locked devices, multiple factor authentication, and GPS tracking.

While all stipulated hardware requirements pose a significant concern to the feasibility of proposed solutions, care must be taken in this regard to ensure that we are effectively meeting these needs without going too far beyond what is necessary for effective operation of the system. Emphasis must be placed on accurately assessing the net value provided by the specific benefits of a solution rather than just considering the number of benefits proposed.

### 5.1.2. Economic

Tangible Benefits

The most important tangible benefit that many businesses will consider when determining the economic feasibility of a new information system is the potential for increased revenue. Our focus for the economic feasibility study for "Open Road" are two-fold; the ability to offer and support a larger number of services, and employee time saved through the automation of business processes within the system. In either case, Precision Riding Academy will most definitely see a tangible (financial) effect of the proposed system not long after it is implemented. The design of the system will allow for simple management of courses, classes, students, instructors, and other resources related to motorcycle safety training. This reduces the potential for errors associated with mainly paper based or manual process systems by replacing the human components with computer components that are faster, more reliable, and more accurate. This leads into the other part of the equation, which is the reduction in required man hours to support the various processes and services defining Precision Riding Academy. With their goal of being as close to "not-for-profit" as is fiscally possible, reducing the amount of funds required to pay staff leads to more funding that can be put into other aspects of the business. Any time saved through use of the proposed system will provide immediate, tangible benefits to PRA and Precision Powersports Ltd.

Intangible Benefits

With how much time society spends online, it comes as no surprise that companies rise or fall based on the extent of their presence on the world wide web. By building the "Open Road" project as a fully online dynamic web portal, Precision Riding Academy will place itself in an excellent position to steadily grow their online support base. A broad online presence gives PRA the opportunity to support their students, staff, and supporters in such a way as to provide top quality services in a open and friendly community environment.

With how closely related Precision Powersports Ltd. is to the Academy, it should come as no surprise that any growth experienced by one party will influence the other. The proposed system will help the Academy to measure and direct the growth they are planning for, allowing it to be more easily passed along to PPS Lethbridge. Cash flow analysis that appears in one of the following pages, provides some financial estimations on what kind of growth Precision Riding Academy could see in the coming years with the implementation of the proposed system.

Web Hosting Platform

The proposed system will provide the most economic benefit to its users in an environment that allows for as much customization of the hardware and supporting software as is possible. The more control that Precision Riding Academy has over their web hosting platform, the easier it will be to deal with issues that arise, develop extensions for, and in general provide a more substantial amount of support to the proposed system. The most important comparison to be considered is between the more common shared server space web hosting and web hosting on a dedicated server.

Shared Server Space vs. Dedicated Server Platform

| Shared Server Space | Dedicated Server |
|---|---|
| • Many restrictions on available resources, few available | • Close to full control over available resources, many available |
| • Very difficult to customize hardware package | • Very easy to customize hardware package |
| • Each website/application must have a rented/purchased IP address. | • Most packages provide more than one IP address allowing for the server to host multiple websites/applications. |
| • Restricted ability to create and preserve data backups/archives | • Full control over data backups/archives |
| • Prices range from $10 to $30 per month per website/application | • Prices range from $150 to $250 per month per dedicated server |

*Table 5.1.2. - 1*

## Cash Flow Analysis

| | Year 0 | Year 1 | Year 2 | Year 3 | Total |
|---|---|---|---|---|---|
| **Benefits:** | | | | | |
| Increased revenue* | | 5,980.00 | 14,950.00 | 20,920.00 | 41,850.00 |
| Efficiency gained** | | 1,500.00 | 2,125.00 | 2,812.50 | 6,437.50 |
| **Total Benefits** | | 7,480.00 | 17,075.00 | 23,732.50 | 48,287.50 |
| **Present Value Total Benefits** | | 5,619.83 | 12,828.70 | 17,830.58 | 36,279.11 |
| **Development Costs:** | | | | | |
| Business Analysis | 2,700.00 | | | | 2,700.00 |
| Project Management | 2,700.00 | | | | 2,700.00 |
| UI/UX Design | 1,800.00 | | | | 1,800.00 |
| Front-end Development | 3,600.00 | | | | 3,600.00 |
| Back-end Development | 5,400.00 | | | | 5,400.00 |
| Implementation and Training | 1,800.00 | | | | 1,800.00 |
| Less: LC Student Project Discount | 18,000.00 | | | | 18,000.00 |
| **Total Development Costs** | 0.00 | | | | 0.00 |
| **Operational Costs:** | | | | | |
| Web Hosting Fee | | 250.00 | 250.00 | 250.00 | 750.00 |
| Online Payment Processing fees | | 418.60 | 1,046.50 | 1,464.40 | 2,929.50 |
| IS maintenance contract*** | | 2,250.00 | 2,250.00 | 2,250.00 | 6,750.00 |
| User training and support | | 500.00 | 525.00 | 551.25 | 1,576.25 |
| **Total Operational Costs** | | 3,418.60 | 4,071.50 | 4,515.65 | 12,005.75 |
| **Total Costs** | **0.00** | **3,418.60** | **4,071.50** | **4,515.65** | **12,005.75** |
| **Total Benefits - Total Costs** | **0.00** | **4,061.40** | **13,003.50** | **19,216.85** | **36,281.75** |
| **Cumulative New Cash Flow** | **0.00** | **4,061.40** | **17,064.90** | **36,281.75** | |
| **Present Value Total Costs** | | **2,568.44** | **3,058.98** | **3,392.67** | **9,020.10** |
| **Return on Investment (ROI)** | **133%** | **(48,287.50/21,839.73)** | | | |
| **Break Even Point** | ***< 1 year from system implementation*** | | | | |
| **NPV (PV Total Benefits - PV Total Costs)** | **$ 27,259.02** | | | | |
| **Intangible Costs and Benefits** | **Intangible Benefit: Enhanced competitive position through expansion of the company's online presence. Intangible Benefit: Increased consumer traffic for parent company, Precision Powersports.** | | | | |

*Table 5.1.2. - 2*

**Year 1 – Classes Offered: 2 Beginner/month*
**Year 2 – Classes Offered: 2 Beginner, 1 Intermediate/month*
**Year 4 – Classes Offered: 2 Beginner, 1 Intermediate, 1 Scooter/month*
**Operating Period: 3rd weekend in May to last weekend in October (weather permitting) = 24 weekends*
**Based on 75% fill rate of available seats*
***$ estimate of employee costs saved through automation of business processes*
****Service contract includes bug fixing, potential system expansion*

### 5.1.3. Organizational

Training Time

Regarding training, all instructors and admin will need to be trained on the new system, and they may not be "computer people". Admins and instructors should be able to spend as little time learning the new system as possible before becoming functionally proficient.

Training Materials

As there may not always be someone available for training who knows the system, there will need to be some sort of training materials available. Whether those materials are available online, within the system, or as hard copies, the training materials must be available, easily accessible, and easily understood.

Familiar Interface

The users should find the interface familiar, either to common programs like MS Office for the staff, or to other registration systems for the users. With this, must come a degree of customizability. If the user interface cannot be customized to suit the needs of the users, then it will not be a perfect fit for PRA.

Intuitive System

Since students are going to be the majority of the user-base, and they won't necessarily have access to any of the training materials for the system, it will need to be fairly self-explanatory. A good navigation system, and pages with obvious purposes will be needed.

Company Culture

Since PRA is not just looking to run their courses, but also to build a community, there will need to be some sort of social features in the system. Whether this is just a way to easily keep in touch with past students, or an actual social page, it will need to be fully integrated and customizable.

"Personality"

There needs to be quite a bit of customization available for the web portion of the system. PRA requires that the website closely mimics the site of it's parent company, Precision Powersports.

## 5.2. Acquisition Strategy

### Solution Alternatives/ Acquisition Strategies

Based on initial feasibility analyses and requirements determination, the most relevant and helpful solutions are to have the application developed by an in-house developer, to outsource the project to a professional development team, or to invest in the integration and licensing of a prepackaged solution. Each approach is outlined in greater specificity below, including the primary candidate identified.

In-house development – Timothy Streibel

This approach will rely on a continuously employed individual developing a web application using primarily personal resources and continual learning, likely to experience longer development time but a greater personal commitment as a result.

Outsourced development – SIT SOFTware

This solution would involve entering a contractual agreement with a professional software development team, who would lay out competitive analysis, development, and support fees, and maintain a guaranteed standard of professional investment.

Prepackaged software – Arlo Training Management System

This option requires similarly variable analysis fees, offering WordPress site development or integration services for preexisting websites, and a recurring subscription fee as well as a per user customer registration fee to deliver a pre-built web application customized to meet certain business specifics.

### Weighting System

Each concern is categorized by whether it is technical, economic, or organizational in nature. Concerns are then weighed in the below matrix of the above discussed alternatives by relative importance to overarching goals of the system in relation to PRA. 100 Points of weight have been distributed across the nine established primary concerns and each is given a score from 1 to 5, with a score of 5 indicating an optimal solution. Each alternatives score is multiplied by its relative weight to produce a weighted score indicating the total benefit provided to the system, which is then totaled to indicate the overall adequacy of the proposed alternative.

## 5.2.1. Alternative Matrix

| Evaluation Criteria | Relative Importance (Weight) | Alternative 1: PRA In-house development | Score (1-5) | Weighted Score | Alternative 2: SIT SOFTware, Project Open Road | Score (1-5) | Weighted Score | Alternative 3: Arlo, Motorcycle Training Software Provider | Score (1-5) | Weighted Score |
|---|---|---|---|---|---|---|---|---|---|---|
| Technical Concerns: | | | | | | | | | | |
| Extensibility | 14 | Custom design, implemented gradually & iteratively, as needed based on continued requirements determination on-site. | 4 | 56 | Custom design, post contract features added per new/ongoing agreements as identified by users. | 5 | 70 | Proprietary design with inaccessible backend and limited customizability. Services are limited to what a chosen package offers. | 1 | 14 |
| Available Support | 12 | Relentless ongoing support from an on-site employee with a growing skill set. Limited by lack of team members. | 4 | 48 | Guaranteed resolution, extensive support, availability & professionalism standards. Limited to agreed upon boundaries. | 4 | 48 | Phone support only available at higher price points, larger customer base results in longer response times. | 3 | 36 |
| Hardware Requirements | 9 | Low overhead web app, developed primarily using limited personal resources with outsourced hosting. | 3 | 27 | Web-based application developed off-site using collective resources with access to further facilities but outsourced hosting. | 4 | 36 | Ready to integrate web application developed and hosted remotely as a comprehensive service package. | 5 | 45 |
| Economic Concerns: | | | | | | | | | | |
| Development Costs | 9 | Regularly paid individual spend paidtime learning, with reduced time consulting due to prior knowledge. | 4 | 36 | Competetive development, analysis & support costs with negotiable ongoing service fees. | 5 | 45 | Scaled analysis, development & integration fees as well as an ongoing subscription fees. | 2 | 18 |
| Operational Costs | 11 | Training/support provided, no final product licensing, extensive dev tools may require licensing, recurring employee wage and site hosting fees will be incurred. | 3 | 33 | No licensing fees, hosting not included in development but extensive training and customer support is provided. | 4 | 44 | Monthly subscription that includes web hosting services and a per user registration fee that hamstirngs the businesses profitability. | 2 | 22 |
| Business Value | 15 | Significant administrative value, labor reduction and improved analytics as well as an employee with a growing suite of skills at the companies disposal moving forward. | 5 | 75 | Immense reduction in menial tasks, and administration work with improve analytics. Improved online brand presence and improved customer service and outreach. | 5 | 75 | All in one service imited in specific utility and training support as well as extinsive & steep perpetual & initial fees that throttle the client business. | 2 | 30 |
| Organizational Concerns: | | | | | | | | | | |
| Ease of learning | 10 | Intuitive UI as a result of implicit familiarity and understanding of business operations and requirements. Coworker relationships will aid in user training time and willingness. | 5 | 50 | Included training and assistance as well as freely available extensive documentation / manuals & custom interface. | 5 | 50 | Foreign UI as a result of predefined composition and dissonance in requirements as a result of physical distance and lack of personal investment. | 2 | 20 |
| Ease of use | 12 | Business familiarity aids once again in the likelihood of developing intuitive and easy to use interfaces that save time spent learning. | 5 | 60 | System designed to mimic familiar forms & follow common processes optimally, with great care being taken to promote simplicity. | 5 | 60 | The solution will not be as intuitively customized, nor will it account for specific operations and user proclivities, but will be backed by extensive design experience. | 3 | 36 |
| Company goals | 8 | While interfaces will be highly customized, design sense is limited to an individuals abilities, who will hold the sole responsibility of creating a clean image for PRA. | 4 | 32 | Custom interfaces will reflect company values and laid back culture while perpetuating existing parent company branding. | 5 | 40 | Experienced professionals with an eye for design will ensure a professional image, but may become disjointed from overarching company goals & priorities. | 3 | 24 |
| Total | | | | 417 | | | 468 | | | 245 |

*Table 5.2.2. - 1*

### 5.2.2. Recommendation

The overall best alternative solution according to the matrix is: Alternative 2 – Outsourcing web application development to SIT SOFTware, with a total feasibility score of 468. This choice clearly outclasses its competitors in that it can remain flexible to change while accurately meeting current requirements, without requiring investment in additional on-site physical infrastructure for the still young business venture. The solution also excels in its ability to provide flexibility in support and development to aid in overall business value without demanding permanent financial investment for basic functionality. The development teams' increased focus on maintaining accessibility through documentation, manuals and one-on-one training will also ensure successful organizational integration.

### 5.2.3. Proposed System Diagram

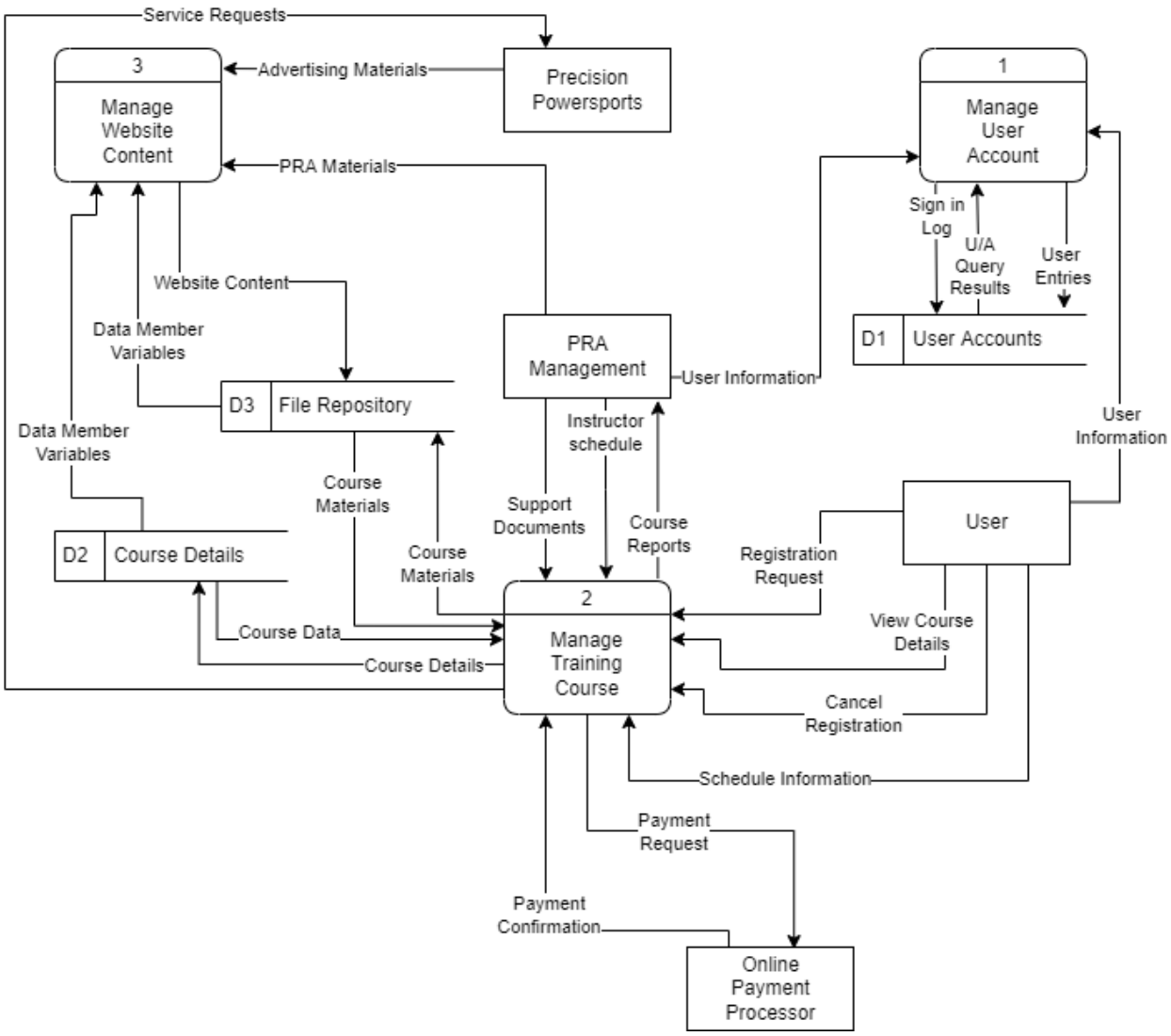



*Figure 5.2.3. - 1*

Based on the additional information gained following a thorough analysis of the business and its needs, the system model proposed in Milestone 3 is still an accurate representation of the proposed system. It details all of the data flow between users, the system, and data stores.

## 5.3. Project Budget

### 5.3.1. Overview

To provide and accurate assessment of the projects estimated cost, the following budget has been developed using the different phases in the Software Development Life Cycle and the project Milestones that measure task progress. Other items that may appear on an information system development budget, like the cost of software licenses required for development, are not present. The reason being that all development is going to be done using open-source integrated development environment and appropriate hardware readily accessible by the development team.

The table below (*see table 5.3.2. – 1*), shows the final estimated budget for the "Open Road" project.

### 5.3.2. Breakdown

# "Open Road"

Project Info

| | |
|---|---|
| **Project Lead:** | **Timothy Streibel** |
| **Start Date:** | **September 7th, 2022** |

## Project Budget

Budget Summary

| | | | Budget | Actual | Under(Over) |
|---|---|---|---|---|---|
| | | Labor | $ 18,000 | $ 18,000 | $ - |
| Tasks | Hrs | Rate | Budget | Actual | Under(Over) |
| **Planning Phase** | | | $ 500 | $ 500 | $ - |
| Initial Project Analysis and Justification | 8 | $50.00 | 400.00 | 400.00 | - |
| System Request | 2 | $50.00 | 100.00 | 100.00 | - |
| **Analysis Phase** | | | $ 2,800 | $ 2,800 | $ - |
| Define Proposed System Requirements and Scope | 8 | $50.00 | 400.00 | 400.00 | - |
| Process Modeling using DFDs and Use Cases | 16 | $50.00 | 800.00 | 800.00 | - |
| Data Modeling - Logical Design | 12 | $50.00 | 600.00 | 600.00 | - |
| Feasibility Analysis of Proposed System | 16 | $50.00 | 800.00 | 800.00 | - |
| Summary Recommendation | 4 | $50.00 | 200.00 | 200.00 | - |
| **Design Phase** | | | $ 3,800 | $ 3,800 | $ - |
| Data Modeling - Physical Design | 16 | $50.00 | 800.00 | 800.00 | - |
| User Interface Design | 36 | $50.00 | 1,800.00 | 1,800.00 | - |
| Program Structure and Specifications | 24 | $50.00 | 1,200.00 | 1,200.00 | - |
| **Implementation Phase** | | | $ 8,200 | $ 8,200 | $ - |
| Program Coding | 64 | $50.00 | 3,200.00 | 3,200.00 | - |
| Program Testing | 64 | $50.00 | 3,200.00 | 3,200.00 | - |
| Develop User Training Guide | 27 | $50.00 | 1,350.00 | 1,350.00 | - |
| Implementation of New System | 8 | $50.00 | 400.00 | 400.00 | - |
| User Training | 1 | $50.00 | 50.00 | 50.00 | - |
| **Other** | | | $ 2,700 | $ 2,700 | $ - |
| Project Planning and Management | 54 | $50.00 | 2,700.00 | 2,700.00 | - |

*Table 5.3.2. - 1*

## 5.4. Project Timeline

### 5.4.1. Overview

Based on the general due dates for Milestones 6 through 10, a general timeline of the design and implementation phases has been planned and is shown in Figure 5.4.1-1.

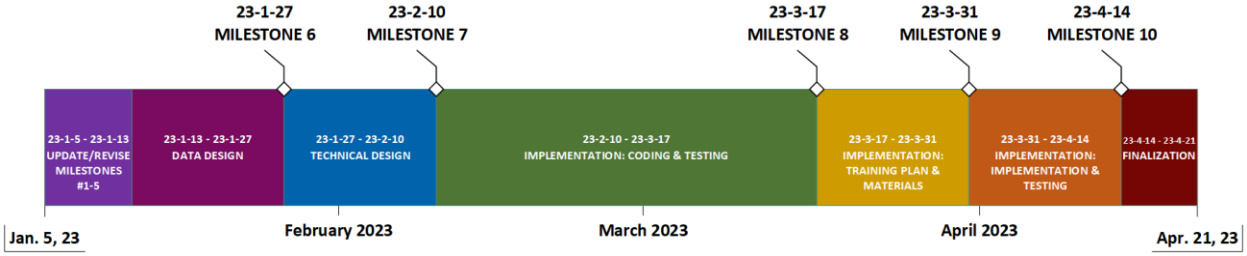



*Figure 5.4.1-1 – General Timeline*

- Revision/Updating of Milestones 1 through 5
  - Due the second week of January (Estimated January 13th)
- Milestone #6: Data Design
  - Due the last week of January (Estimated January 27th)
- Milestone #7: Technical Design
  - Due the second week of February (Estimated February 10)
- Milestone #8: Coding and Testing
  - Due the third week of March (Estimated March 17th)
- Milestone #9: Training Plan and Materials
  - Due the fourth week of March (Estimated March 31st)
- Milestone #10: Implementation and Training
  - Due the second week of April (Estimated April 14th)
- Finalization and Presentation
  - Due end of term (Estimated April 21st)

### 5.4.2. Gantt Chart

A more detailed schedule has been created in Gantt Chart format, showing individual tasks and task flows. (*See Figure 5.4.2. - 1*)

Milestones 6-10

| ID | Task Name | Start | Finish | Duration |
|---|---|---|---|---|
| 1 | **Update/Revise Milestones #1-5** | **2023-01-05** | **2023-01-13** | **8d** |
| 2 | Review/Redefine Scope | 2023-01-05 | 2023-01-09 | 4d |
| 3 | Update Project Timeline | 2023-01-09 | 2023-01-13 | 5d |
| 4 | **MS#6 – Data Design** | **2023-01-14** | **2023-01-27** | **12d** |
| 5 | Physical ERD | 2023-01-14 | 2023-01-20 | 6d |
| 6 | Data Dictionary | 2023-01-21 | 2023-01-23 | 2d |
| 7 | Backup & Archiving Plan | 2023-01-20 | 2023-01-27 | 7d |
| 8 | **MS#7 – Technical Design** | **2023-01-28** | **2023-02-10** | **12d** |
| 9 | User Interface Design | 2023-01-28 | 2023-02-03 | 6d |
| 10 | Process Design | 2023-01-28 | 2023-02-03 | 6d |
| 11 | Physical Architecture | 2023-02-04 | 2023-02-10 | 6d |
| 12 | **MS#8 – Coding & Testing** | **2023-02-11** | **2023-03-17** | **30d** |
| 13 | Coding | 2023-02-11 | 2023-03-06 | 20d |
| 14 | Testing | 2023-03-07 | 2023-03-17 | 10d |
| 15 | **MS#9 – Training Plan and Materials** | **2023-03-18** | **2023-03-31** | **12d** |
| 16 | Write Training Plan | 2023-03-18 | 2023-03-22 | 4d |
| 17 | Compose Training Materials | 2023-03-23 | 2023-03-28 | 5d |
| 18 | Train Another Team | 2023-03-29 | 2023-03-31 | 3d |
| 19 | **MS#10 – Implementation and Testing** | **2023-04-01** | **2023-04-14** | **12d** |
| 20 | Write Implementation Plan | 2023-04-01 | 2023-04-04 | 3d |
| 21 | Implementation | 2023-04-05 | 2023-04-07 | 3d |
| 22 | Training | 2023-04-08 | 2023-04-14 | 6d |
| 23 | **Final Presentation Preparation** | 2023-04-19 | 2023-04-20 | 2d |
| 24 | **Final Presentation** | 2023-04-21 | 2023-04-21 | 1d |

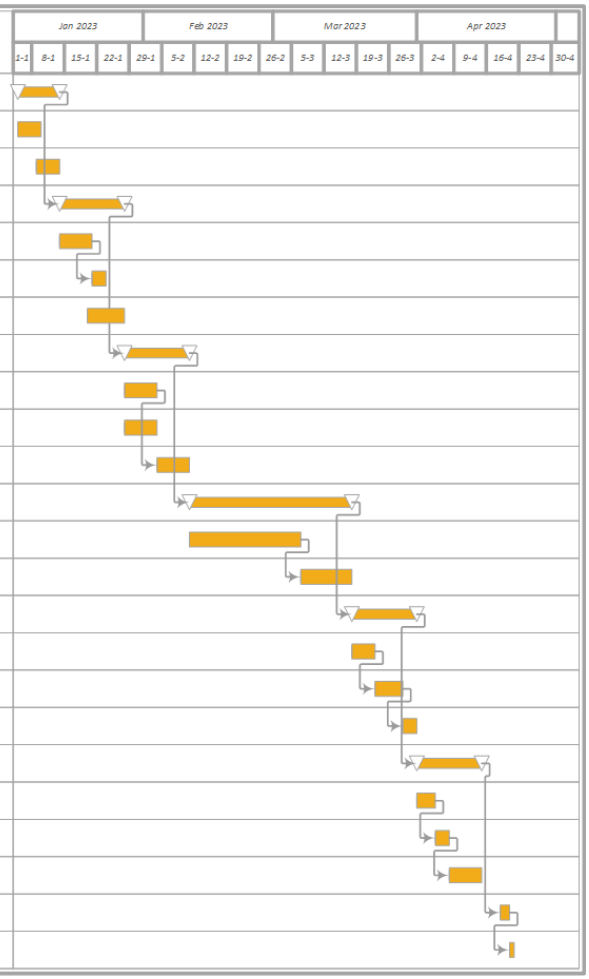



*Figure 5.4.2. - 1*

## 5.5. Conclusion

In all the necessary areas, the proposed system will only provide benefits to and a solid foundation for Precision Riding Academy as they work to expand their influence in the local market of motorcycle safety training courses. While other options to acquire the system do exist, it has been shown that the strategy that best fits the needs of PRA is by outsourcing the development to the SIT SOFTware team. To further support this recommendation, we have provided project budget and timeline estimates that help to define a framework for accomplishing the tasks required in developing the proposed system. As a final closing statement for this milestone, we of SIT SOFTware declare:

The system can be built, it should be built, and if it is built, we are confident that it will be widely accepted and used.

# 6. Milestone Six

# 6.1. Physical Data Design

## 6.1.1. Physical ERD

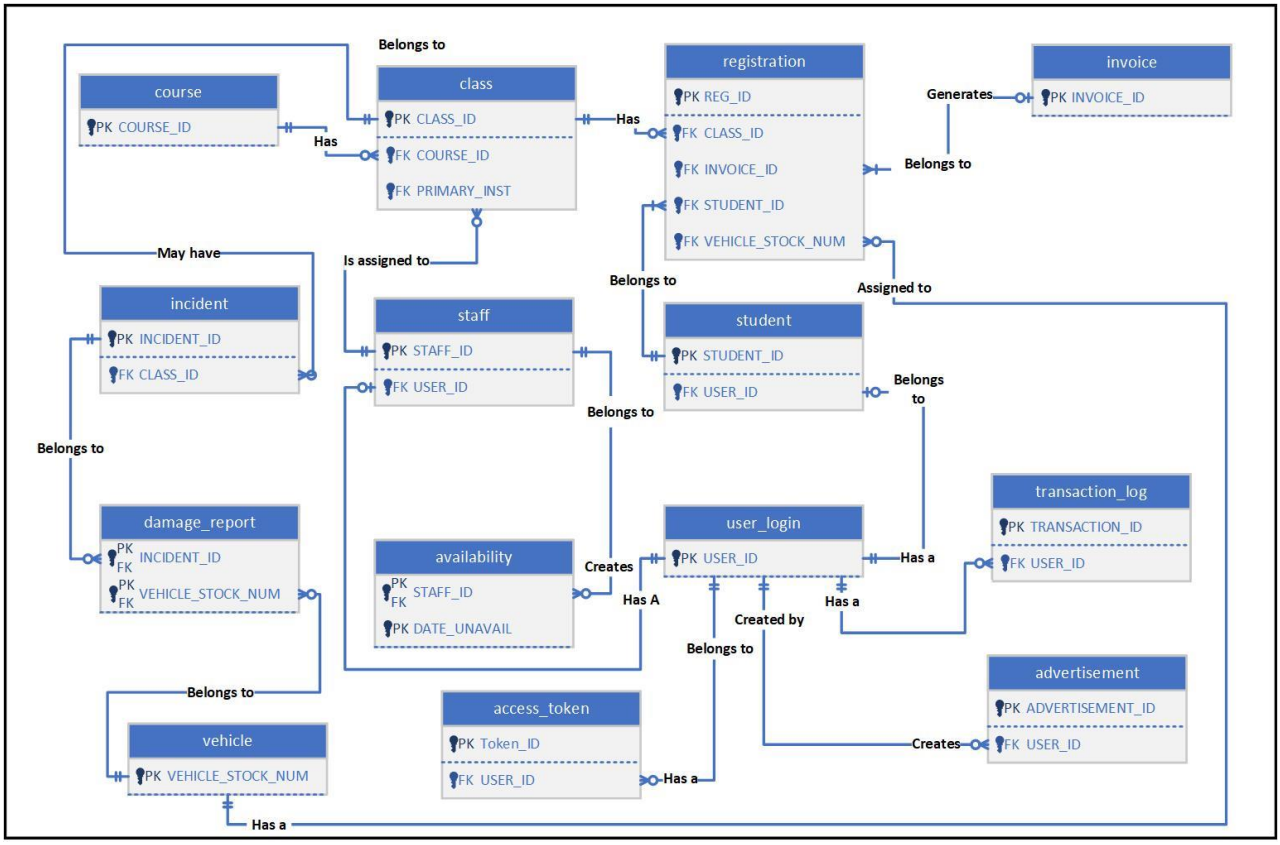



*Figure 6.1.1 - 1*

We made two major changes to the ERD since Milestone Six. A table was added for controlling advertisement banners on the website. This was a change made to help the client control when and where ads can be seen on the website. This will be connected to the rest of the database by a one-to-many relationship with the user_login table.

Two Factor Authentication was something we decided needed to be added because we were going to be storing sensitive information in the form of Driver's License Numbers. The module we are using for this requires another table be added to the database. The access_token table has a one-to-many relationship with the user_login table, as the token is directly connected to individual user IDs.

## 6.1.2. Data Dictionary

| Table Name | Attribute Name | Description | Data Type | Format/Mask | Required | PK or FK | FK Ref Table | Sample Data |
|---|---|---|---|---|---|---|---|---|
| user_login | USER_ID | Unique Code to identify users | UnsignedInt(4) | 9999 | Y | PK | | 1234 |
| | USER_PW | Password chosen by the user | Varchar(20) | | Y | | | Abc$123 |
| | USER_NAME | Username chosen by the user | Varchar(20) | | Y | | | BobSmith |
| | USER_STATUS | Whether the user is Active or Inactive | UnsignedInt(1) | 9 | Y | | | 1 |
| | LAST_ACCESSED | When the account was last accessed | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| access_token | TOKEN_ID | Unique Code to identify the access token | UnsignedInt(10) | 9999999999 | Y | PK | | 5698741320 |
| | USER_ID | Unique Code to identify users | UnsignedInt(4) | 9999 | Y | FK | user_login | 1234 |
| | TOKENABLE_TYPE | The type of access token | Varchar(256) | | Y | | | Google Authenticator |
| | TOKEN | Private key portion of the access token | Varchar(256) | | Y | | | 654sdf81d9re5d264c8s146356431d82c |
| | LAST_USED | When the token was last used to log in | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | CREATED | When the token was originally created | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | LAST_UPDATED | When the token was last updated | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 |
| student | STUDENT_ID | Unique Code to identify students | Char(5) | X9999 | Y | PK | | S4321 |
| | USER_ID | Unique Code to identify users | UnsignedInt(4) | 9999 | Y | FK | user_login | 1234 |
| | STUDENT_PNAME | Preferred name of the Student | Varchar(20) | | | | | Bobby |
| | STUDENT_FNAME | First name of the Student | Varchar(20) | | Y | | | Bob |
| | STUDENT_LNAME | Last name of the Student | Varchar(20) | | Y | | | Smith |
| | STUDENT_PHONE | Phone number of the Student | Char(12) | 999-999-9999 | Y | | | 403-123-4567 |
| | STUDENT_EMAIL | Email address of the Student | Varchar(40) | | Y | | | bobsmith@bob.com |
| | STUDENT_DOB | Students Date of Birth | Date | yyyy-mm-dd | Y | | | 1932-02-29 |
| | STUDENT_DLN | Students Drivers License Number | Char(10) | 999999-999 | Y | | | 123456-789 |
| | STUDENT_ADDR1 | Line one of the Students address | Varchar(30) | | Y | | | 123 Cherry Lane |
| | STUDENT_ADDR2 | Line two of the Students address | Varchar(30) | | | | | Appt 64 |
| | STUDENT_CITY | City the Student lives in | Varchar(30) | | Y | | | Lethbridge |
| | STUDENT_PR_ST | Province or State the Student lives in | Char(2) | XX | Y | | | AB |
| | STUDENT_COUNTRY | Country the Student lives in | Varchar(30) | | Y | | | Canada |
| | STUDENT_POST_ZIP | Postal or Zip Code of the Student | Varchar(7) | | Y | | | A1B C2D |
| | EMERGCONT_NAME | Name of the Students emergency contact | Varchar(40) | | | | | John Doe |
| | EMERGCONT_PHONE | Phone number of the Students emergency contact | Char(12) | 999-999-9999 | | | | 403-987-6543 |
| | STUDENT_WAIVER | Whether the student has a signed waiver | UnsignedInt(1) | 9 | Y | | | 1 |
| | CREATED | When the account was created | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | LAST_UPDATED | When the account was last updated | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | DELETED | When the account was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 |
| staff | STAFF_ID | Unique Code to identify staff members | Char(5) | X9999 | Y | PK | | E4321 |
| | USER_ID | Unique Code to identify users | UnsignedInt(4) | 9999 | Y | FK | user_login | 1234 |
| | STAFF_LEVEL | Level of site access for the staff member | Char(1) | X | Y | | | A |
| | STAFF_PNAME | Preferred name of the Staff member | Varchar(20) | | | | | Bobby |
| | STAFF_FNAME | First name of the Staff member | Varchar(20) | | Y | | | Bob |
| | STAFF_LNAME | Last name of the Staff member | Varchar(20) | | Y | | | Smith |
| | STAFF_PHONE | Phone number of the Staff member | Char(12) | 999-999-9999 | Y | | | 403-123-4567 |
| | STAFF_EMAIL | Email address of the Staff member | Varchar(40) | | Y | | | bobsmith@bob.com |
| | STAFF_DOB | Staff members Date of Birth | Date | yyyy-mm-dd | Y | | | 1932-02-29 |
| | STAFF_DLN | Staff members Drivers License Number | Char(10) | 999999-999 | Y | | | 123456-789 |
| | STAFF_ADDR1 | Line one of the Staff members address | Varchar(30) | | Y | | | 123 Cherry Lane |
| | STAFF_ADDR2 | Line two of the Staff members address | Varchar(30) | | | | | Appt 64 |
| | STAFF_CITY | City the Staff member lives in | Varchar(30) | | Y | | | Lethbridge |
| | STAFF_PR_ST | Province or State the Staff member lives in | Char(2) | XX | Y | | | AB |
| | STAFF_COUNTRY | Country the Staff member lives in | Varchar(30) | | Y | | | Canada |
| | STAFF_POST_ZIP | Postal or Zip Code of the Staff member | Varchar(7) | | Y | | | A1B C2D |
| | EMERGCONT_NAME | Name of the Staff members emergency contact | Varchar(40) | | | | | John Doe |
| | EMERGCONT_PHONE | Phone number of the Staff members emergency contact | Char(12) | 999-999-9999 | | | | 403-987-6543 |
| | CREATED | When the account was created | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | LAST_UPDATED | When the account was last updated | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | DELETED | When the account was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 |

*Figure 6.1.2 – 1*

| Table Name | Attribute Name | Description | Data Type | Format/Mask | Required | PK or FK | FK Ref Table | Sample Data |
|---|---|---|---|---|---|---|---|---|
| availability | STAFF_ID | Unique Code to identify staff members | Char(5) | X9999 | Y | PK/FK | staff | E4321 |
| | DATE_UNAVAIL | Date the staff member is unavailable to work | Date | yyyy-mm-dd | Y | PK | | 1932-02-29 |
| | APPROVED | Whether the request has been approved | UnsignedInt(1) | 9 | Y | | | 1900-01-00 |
| transaction_log | TRANSACTION_ID | Autogenerated number | UnsignedInt(5) | 99999 | Y | PK | | 54321 |
| | USER_ID | ID of the user initiating transaction | UnsignedInt(4) | 9999 | Y | FK | user_login | 1234 |
| | TRANSACTION_DATE | Date the transaction occurred | Date | yyyy-mm-dd | Y | | | 1932-02-29 |
| | TRANSACTION_TIME | Time the user logged in to their account | Time | 99:99:99 | Y | | | 15:26:50 |
| | TRANSACTION_IP_ADDR | IP address of the user initiating transaction | Char(15) | 999.999.999.999 | Y | | | 172.022.002.029 |
| course | COURSE_ID | ID number for the course | Char(3) | 999 | Y | PK | | 101 |
| | COURSE_NAME | Name associated with the course | Varchar(15) | | Y | | | Beginner |
| | COURSE_DOCS | File path to docs related to the course | Varchar(256) | | | | | \Beginner\Docs\ |
| | COURSE_MAX_SEATS | Maximum number of seats for the course | UnsignedInt(1) | 9 | Y | | | 5 |
| | COURSE_FEE | Cost to register for the course | Double(5,2) | 999.99 | Y | | | 259.99 |
| | CREATED | When the account was created | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | LAST_UPDATED | When the account was last updated | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | DELETED | When the account was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 |
| class | CLASS_ID | ID number for the class | Char(6) | yy-999 | Y | PK | | 23-101 |
| | COURSE_ID | Course type of the class | Char(3) | 999 | Y | FK | course | 101 |
| | PRIMARY_INST | Staff ID of the main instructor | Char(5) | X9999 | Y | FK | staff | E4321 |
| | SECONDARY_INST | Staff ID of the assistant instructror | Char(5) | X9999 | | | | E4321 |
| | CLASS_START | Start date of the class | Date | yyyy-mm-dd | Y | | | 2023-05-20 |
| | CLASS_END | End date of the class | Date | yyyy-mm-dd | Y | | | 2023-05-23 |
| | CREATED | When the account was created | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | LAST_UPDATED | When the account was last updated | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | DELETED | When the account was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 |
| registration | REG_ID | Autogenerated number | Char(4) | 9999 | Y | PK | | 1917-12-04 |
| | CLASS_ID | ID number for the class | Char(6) | yy-999 | Y | FK | class | 23-101 |
| | INVOICE_ID | ID number of the invoice | UnsignedInt(4) | 9999 | Y | FK | invoice | 5678 |
| | STUDENT_ID | ID number for the student | Char(5) | X9999 | Y | FK | student | S1234 |
| | VEHICLE_STOCK_NUM | Vehicle the student is initially assigned | Varchar(17) | XXX-999 | Y | FK | vehicle | YAM-123 |
| | REGISTRATION_CANCELLED | Whether the registration has been cancelled | UnsignedInt(1) | 9 | Y | | | 0 |
| | REGISTRATION_WAITLIST | Whether the registration has been waitlisted | UnsignedInt(1) | 9 | Y | | | 0 |
| | CREATED | When the account was created | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | LAST_UPDATED | When the account was last updated | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | DELETED | When the account was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 |
| invoice | INVOICE_ID | ID number of the invoice | UnsignedInt(4) | 9999 | Y | PK | | 5678 |
| | PAYMENT_SUBTOTAL | Subtotal before tax | Double(5,2) | 999.99 | Y | | | 259.99 |
| | PAYMENT_TAX_AMT | Percentage of Sales Tax to be applied | Double(3,2) | 0.99 | Y | | | 0.05 |
| | PAYMENT_TYPE | Method of payment | Varchar(15) | | Y | | | PayPal |
| vehicle | VEHICLE_STOCK_NUM | Stock Number of the vehicle | Varchar(17) | XXX-999 | Y | PK | | YAM-123 |
| | VEHICLE_VIN | VIN of the vehicle | Varchar(17) | 9XXXX99XXXX999999 | Y | | | 5YJSA1DG9DFP14705 |
| | VEHICLE_YEAR | Year the vehicle was manufactured | Char(4) | 9999 | Y | | | 2002 |
| | VEHICLE_MAKE | Make of the vehicle | Varchar(20) | | Y | | | Kawasaki |
| | VEHICLE_MODEL | Model of the vehicle | Varchar(40) | | Y | | | Ninja |
| | VEHICLE_ODO | Odemeter reading of the vehicle | Char(10) | 9999999999 | Y | | | 0002564821 |
| | VEHICLE_TYPE | Type of vehicle | UnsignedInt(1) | 9 | Y | | | 3 |
| | VEHICLE_COLOR | Color of the vehicle | Varchar(10) | | | | | Red |
| | VEHICLE_SIZE | Engine displacement of the vehicle | Varchar(4) | 9999 | Y | | | 750 |
| | AVAIL_STATUS | Availability status of the vehicle | UnsignedInt(1) | 9 | Y | | | 1 |
| | NOTES | Any notes regarding the vehicle | Varchar(256) | | | | | Last maintained 02-29 |
| | CREATED | When the account was created | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | LAST_UPDATED | When the account was last updated | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | DELETED | When the account was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 |

*Figure 6.1.2 - 2*

| Table Name | Attribute Name | Description | Data Type | Format/Mask | Required | PK or FK | FK Ref Table | Sample Data |
|---|---|---|---|---|---|---|---|---|
| incident | INCIDENT_ID | ID# of the incident | UnsignedInt(4) | 9999 | Y | PK | | 1346 |
| | INCIDENT_DATE | Date the incident happened | Date | yyyy-mm-dd | Y | | | 2023-06-17 |
| | CLASS_ID | Registration ID of the student involved | Char(6) | yy-999 | Y | FK | class | 23-101 |
| | INCIDENT_DETAILS | Details of the incident | LongText | | Y | | | They crashed |
| | EMERG_SERV_CALLED | Whether or not an emergency services were called | UnsignedInt(1) | 9 | Y | | | 0 |
| | EMERG_CONTACT_CALLED | Whether or not an emergency contact was called | UnsignedInt(1) | 9 | Y | | | 1 |
| damage_report | INCIDENT_ID | ID# of the incident causing damage | UnsignedInt(4) | 9999 | Y | PK/FK | incident | 1346 |
| | VEHICLE_STOCK_NUM | Stock Number of the vehicle | Varchar(17) | XXX-999 | Y | PK/FK | | YAM-123 |
| advertisement | ADVERTISEMENT_ID | ID# of the advertisement | UnsignedInt(4) | 9999 | Y | PK | | 1355 |
| | USER_ID | Unique Code to identify users | UnsignedInt(4) | 9999 | Y | FK | user_login | 1234 |
| | AD_TITLE | Title of the advertisement | Varchar(20) | | Y | | | X-Mas Sale |
| | AD_DESCRIPTION | Description of the advertisement | Varchar(256) | | Y | | | Sale for x-mas 2023 |
| | AD_URL | URL that the advertisement would link to | Varchar(256) | | | | | https://precisionpowersportsltd.com/ |
| | AD_PATH | Path that the advertisement image is linked from | Varchar(256) | | Y | | | Pictures/Ads/Xmas |
| | AD_SPACE_TYPE | Type of ad (horizontal, vertical, or corner square) | UnsignedInt(1) | 9 | Y | | | 2 |
| | AD_LOCATION | Specific pages where the ads can appear, if any | Varchar(256) | | | | | Student |
| | START_DATE | Date the ad should start running (if left blank, default to current date) | Date | yyyy-mm-dd | Y | | | 1932-02-29 |
| | END_DATE | Date the ad should stop running | Date | yyyy-mm-dd | | | | 1932-02-29 |
| | CREATED | When the advertisement was created | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | LAST_UPDATED | When the advertisement was last updated | Date | yyyy-mm-dd hh:mm:ss | Y | | | 1932-02-29 16:23:50 |
| | DELETED | When the advertisement was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 |

*Figure 6.1.2 - 3*

The most important changes to the Database Dictionary since Milestone 4 are the addition of the access_token and advertisement tables. The access_token table, which is a part of the authentication module we are using, allows for two-factor authentication, easy password recovery, and a remember me feature that will cause the system to query a cookie left on the users device for login credentials, instead of asking for a password.

The advertisement table will enable the system to display in-house advertisements from the Precision Riding Academy and Precision Powersports. It will be connected to the rest of the database via the USER_ID foreign key, allowing administrators to see who uploaded which ads. The most important thing to note for the advertisement table is we will only be storing a relative file path to the ad images, not the images themselves. This is to save on database storage space, as well as to aid in database backup and recovery.

## 6.2. Data Backup & Archiving

### 6.2.1. Data Backup Plan

To protect against bad actors and system errors while preserving valuable data, we will combine several approaches to backing up the web applications code, assets such as files, and the database structure/contents. The different backup process are as follows:

- **Backup 1:** The projects initial layer of redundancy and the primary point of restoration for all source code is our GitHub repository, which will hold the most recent version of the application, relevant project documentation, and official change logs. The initial implementation of the system will include distributing several physical drives with the most recent version of the source code for use in the case of an emergency.
- **Backup 2:** The second point of restoration is the web hosts automatic backup service. While all professional hosting services offer some form of automated backup, the PRA and PPS sites are currently hosted by GoDaddy who offer free daily backups to all subscription levels, with paid options for 7, 14, and 30 day restore points. Should PRA choose to change hosting services, the accompanying free backup service will be used in place of GoDaddy's 24 hour restore point.
- **Backup 3:** The final backup process is an automatic Cron job created within the web host control panel and scheduled to run monthly, storing within the web servers file system csv and SQL files representing the contents and structure of the database tables. These files can be manually moved or copied from the web hosts file system to an external drive by developers & administrators as needed. This manual aspect of the process will be emphasized during training as a way of preserving the state of the site before implementing major changes.

### 6.2.2. Data Archiving Plan

To estimate the amount of data that will need to be supported by our systems hardware we used the volumetric technique. Our calculations account for the average raw data (in characters) as well as an average overhead of 30%, multiplied by the number of rows, or entries, after our expected growth rate per month over 3 years of operation (with a 7-month operating period as a result of weather) which gives us the volume of data in characters which we then multiply by an average amount of 8 bits per character. Performing this calculation leaves us with just 0.244 mb of raw data after 3 years of successful operation. This data is primarily generated by our transaction_log table, while our staff table which has the densest fields generates very few entries.

Due to the miniscule volume of data expected to be processed at a base level, even including source files and backups the application will not become inundated with large amounts of information, nor will the information being stored lose its relevance. As such, data stored by the application will remain as "live" data with archiving not being considered as a current concern for the project. Planning to remove and archive table entries at this point for the businesses would provide little to no benefit, likely only limiting PRAs ability to generate accurate and relevant reports over the long term.

| staff Table | |
|---|---|
| **Field** | **Average Size (Characters)** |
| STAFF_ID | 5 |
| USER_ID | 4 |
| STAFF_LEVEL | 1 |
| STAFF_PNAME | 6 |
| STAFF_FNAME | 6 |
| STAFF_LNAME | 8 |
| STAFF_PHONE | 12 |
| STAFF_EMAIL | 22 |
| STAFF_DOB | 9 |
| STAFF_DLN | 10 |
| STAFF_ADDR1 | 13 |
| STAFF_ADDR2 | 6 |
| STAFF_CITY | 10 |
| STAFF_PR_ST | 2 |
| STAFF_COUNTRY | 6 |
| STAFF_POST_ZIP | 6 |
| EMERGCONT_NAME | 14 |
| EMERGCONT_PHONE | 12 |
| CREATED | 18 |
| LAST_UPDATED | 18 |
| DELETED | 18 |
| **Record Size:** | 206 |
| Overhead | 30% |
| **Total Record Size:** | 267.8 |
| Initial Table Size | 5 |
| **Growth Rate/Month:** | 0.1 |
| **Table Size @ 3 Years:** | 7 |
| **Stored Characters @ 3 Years:** | 1874.6 |
| **Required Storage (bits) @ 3 Years:** | 14996.8 |

*Figure 6.2.2 - 1*

| transaction_log Table | |
|---|---|
| **Field** | **Average Size (Characters)** |
| TRANSACTION_ID | 4 |
| USER_ID | 4 |
| TRANSACTION_DATE | 9 |
| TRANSACTION_TIME | 8 |
| TRANSACTION_IP_ADDR | 15 |
| **Record Size:** | 40 |
| Overhead | 30% |
| **Total Record Size:** | 52 |
| Initial Table Size | 0 |
| **Growth Rate/Month:** | 80 |
| **Table Size @ 3 Years:** | 1680 |
| **Stored Characters @ 3 Years:** | 87360 |
| **Required Storage (bits) @ 3 Years:** | 698880 |

*Figure 6.2.2 -2*

## 6.3. Conclusion

During the course of Milestone Six we have revaluated the tables in the database, and added a couple more to increase system functionality. The first table is going to keep access tokens to add two-factor authentication, easy password recovery, and streamline the login process. Our second table is our local Advertisement, which will only advertise for Precision Powersports and the Precision Riding Academy. The Advertisement table has fields to help us upload, modify and delete, as well as include running dates for when the ad should start/stop running the banner.

For our back up we have a three-tier solution, first the GitHub Repository which holds our source code, and database schema. It can be backed up as needed but maintained by developers and is the primary control method. Second we have the web host that will do a back up automatically. It will back up all files, folders and database data. Third, we will be using an automated cron job to backup all database data into csv files. It will also back up the files and folders in the system. Thorough training will be needed for all admins in the backup processes.

# 7. Milestone Seven

Table of Contents

# 7.1 User Interface Design

## 7.1.1. Site Map

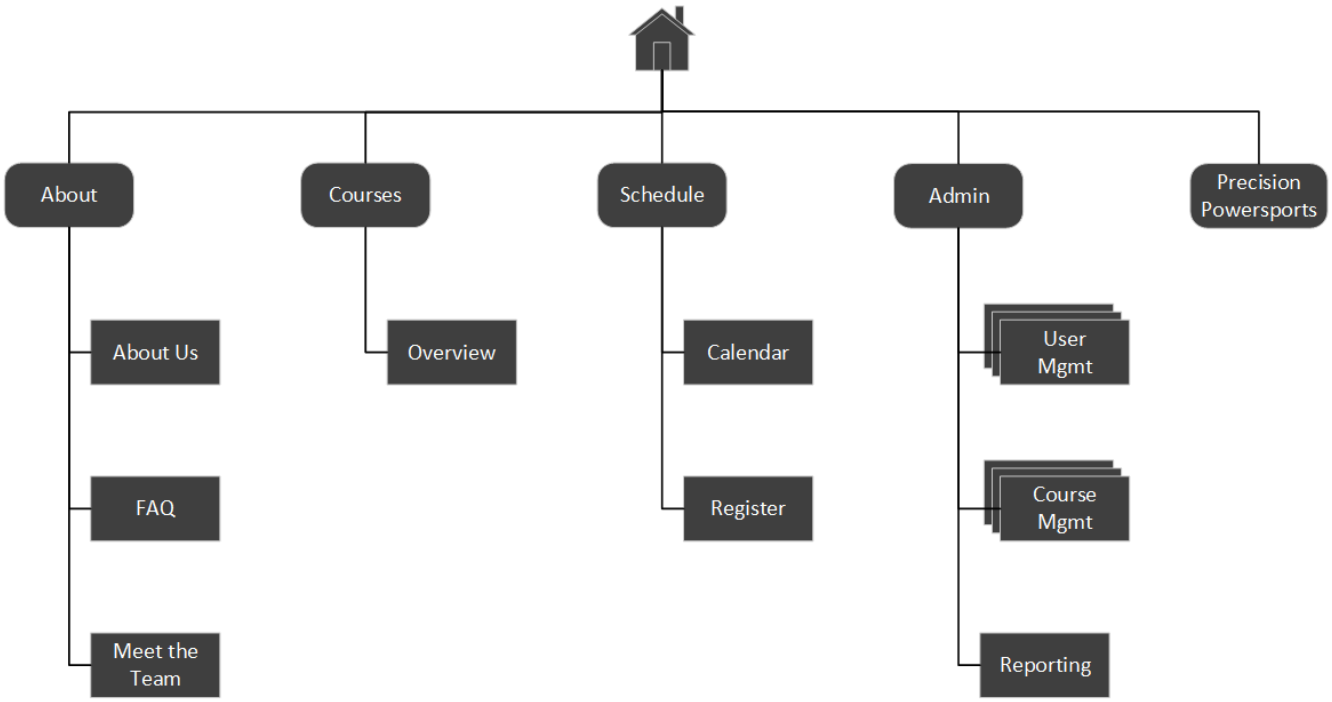



*Figure 7.1.1 - 1 Site Map for Project Open Road*

### 7.1.2 Figma Examples of Views

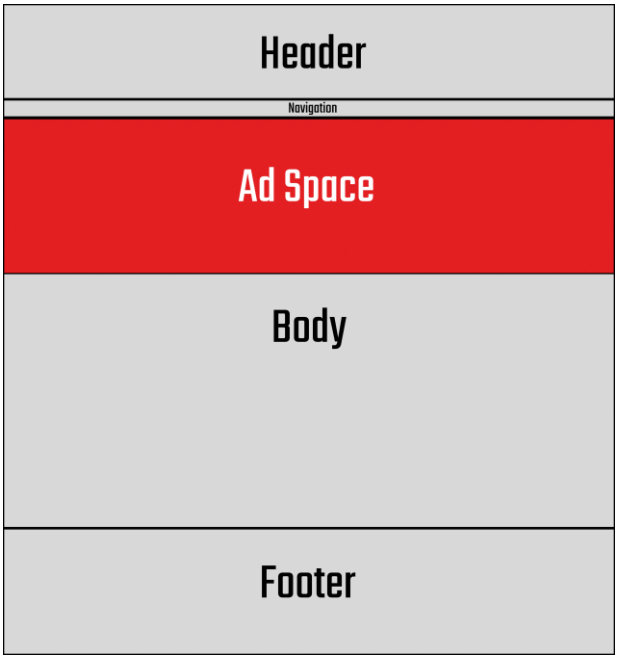



*Figure 7.1.2– 1 User Wireframe*

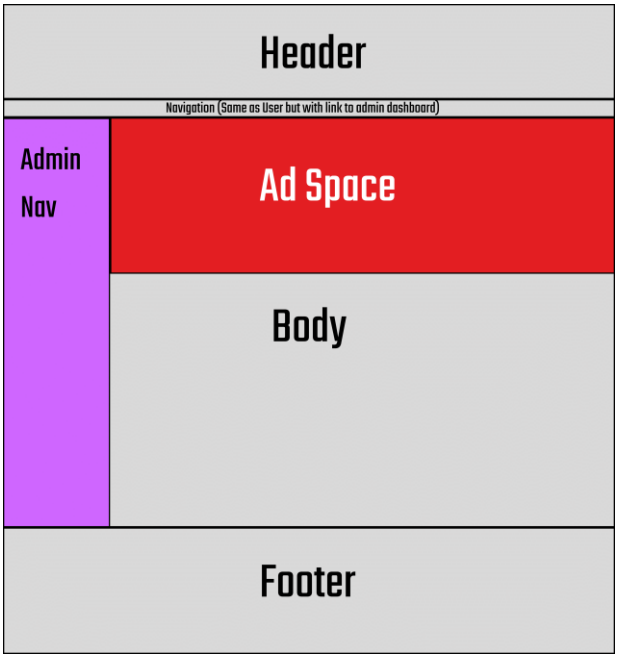



*Figure 7.1.2 – 2 Admin Wireframe*

Figures 7.1.2 – 1 and 7.1.2 – 2 depict the wireframe models for the general page layout. The purple Admin Nav will be visible and collapsible depending on the user type of the currently logged in user.

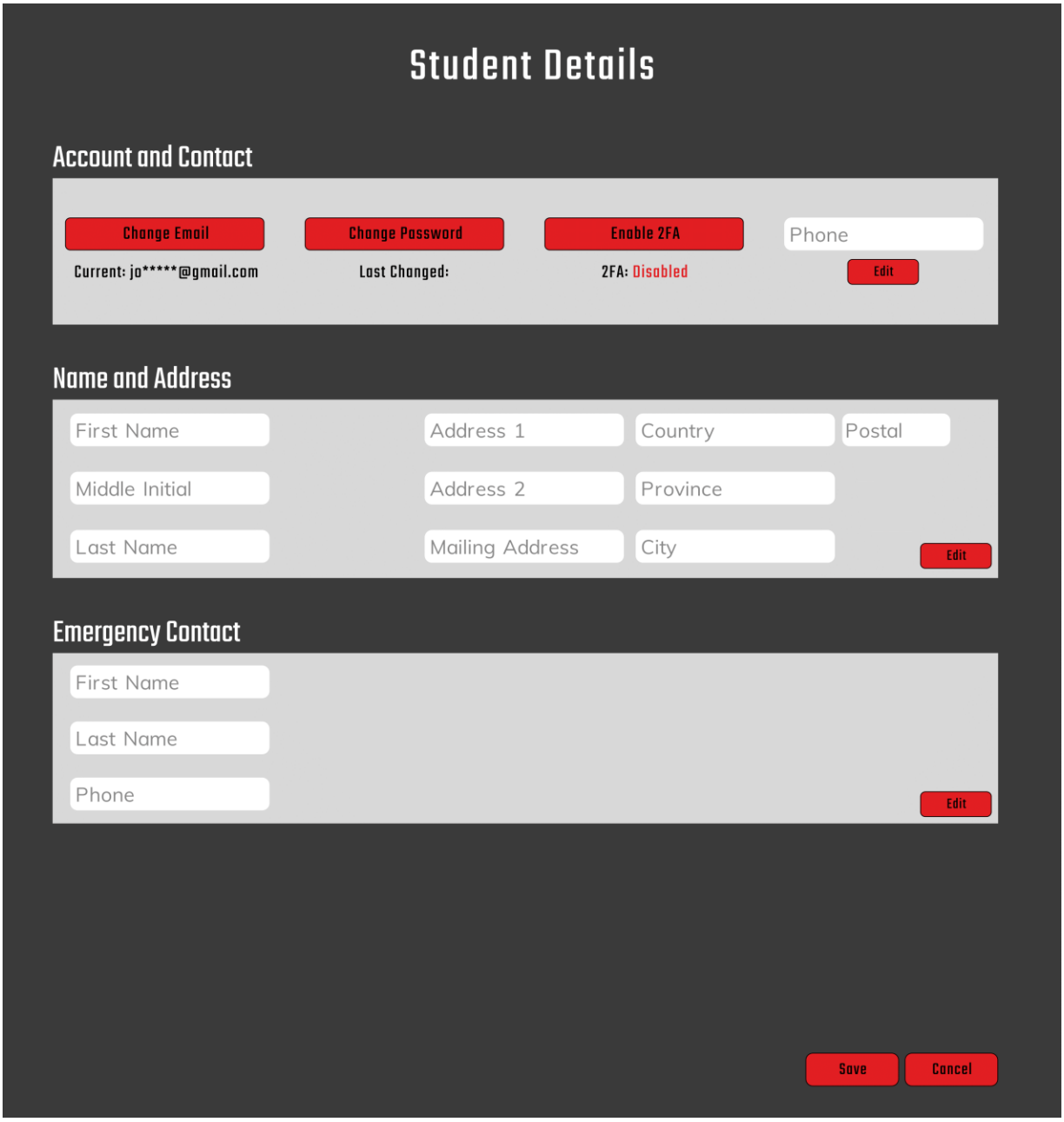



*Figure 7.1.2 – 3 Student Details Form*

Figure 7.1.2 – 3 depicts the Student Details Form. This page will display as inactive fillable fields containing the student's current record until they decide to edit their contact information.

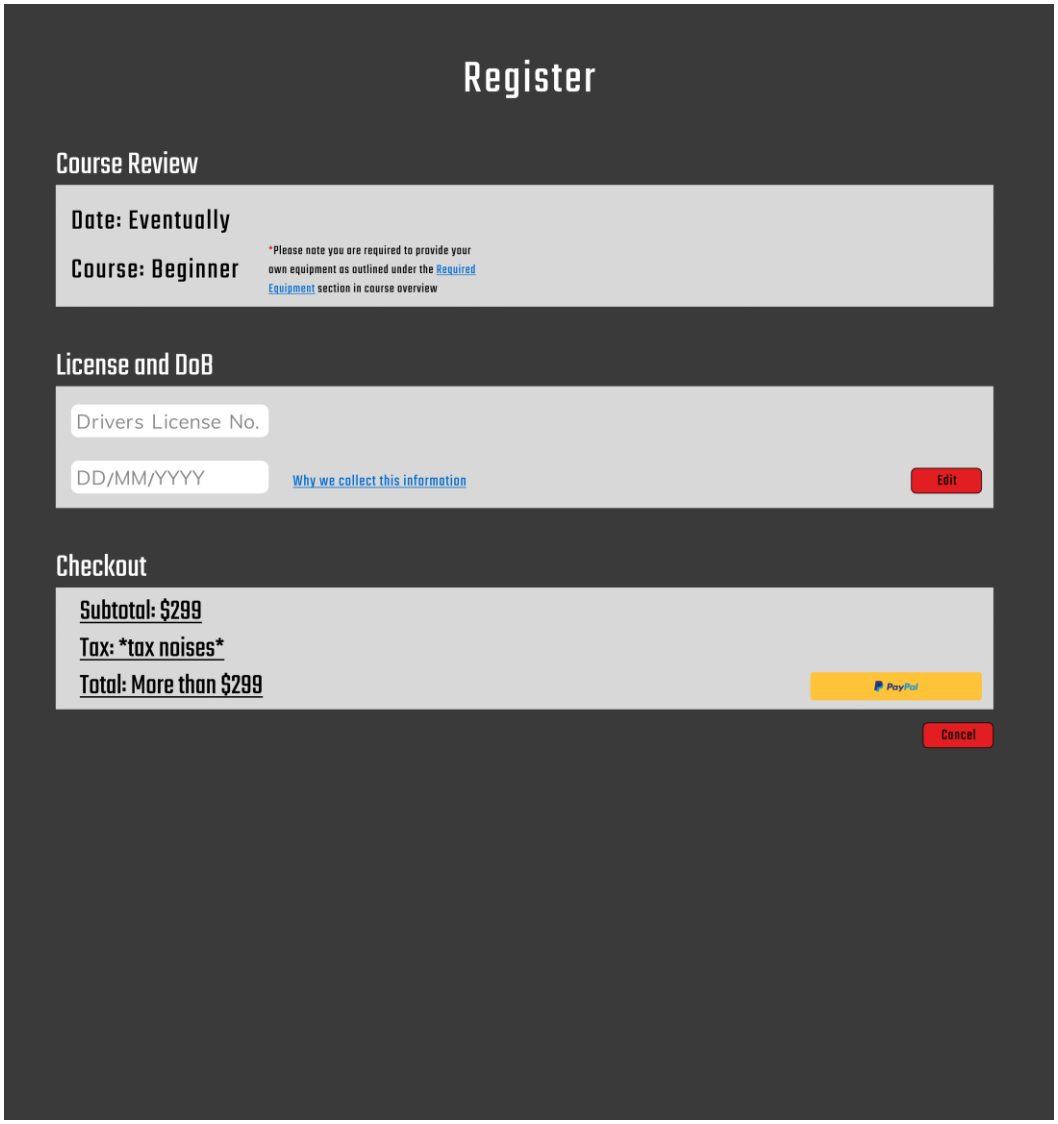



*Figure 7.1.2– 4 Registration Page*

Figure 7.1.2 – 4 depicts the final page in the registration process, the payment page. This outputs the date and type of the class being registered for, as well as the subtotal, tax, and total to be paid. There are also fields on this page for the user to input their Drivers License Number and date of birth. Finally, there is a button that will direct them to the PayPal API where payment will be processed.

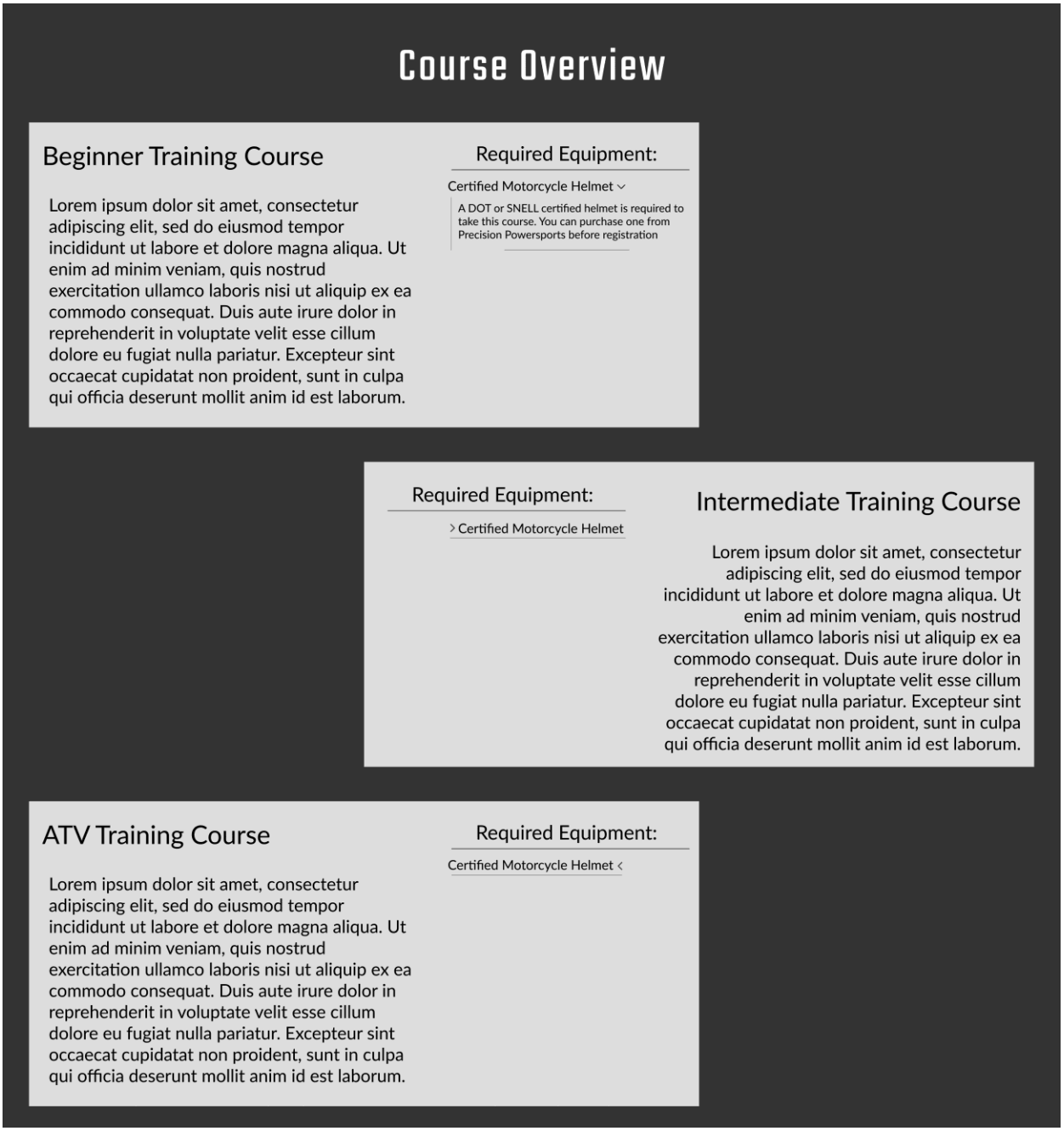



*Figure 7.1.2– 5 Course Overview Page*

Figure 7.1.2 – 5 depicts the Course Overview page. This page displays course information and required equipment that has been pulled from a JSON file and formatted for easy viewing.

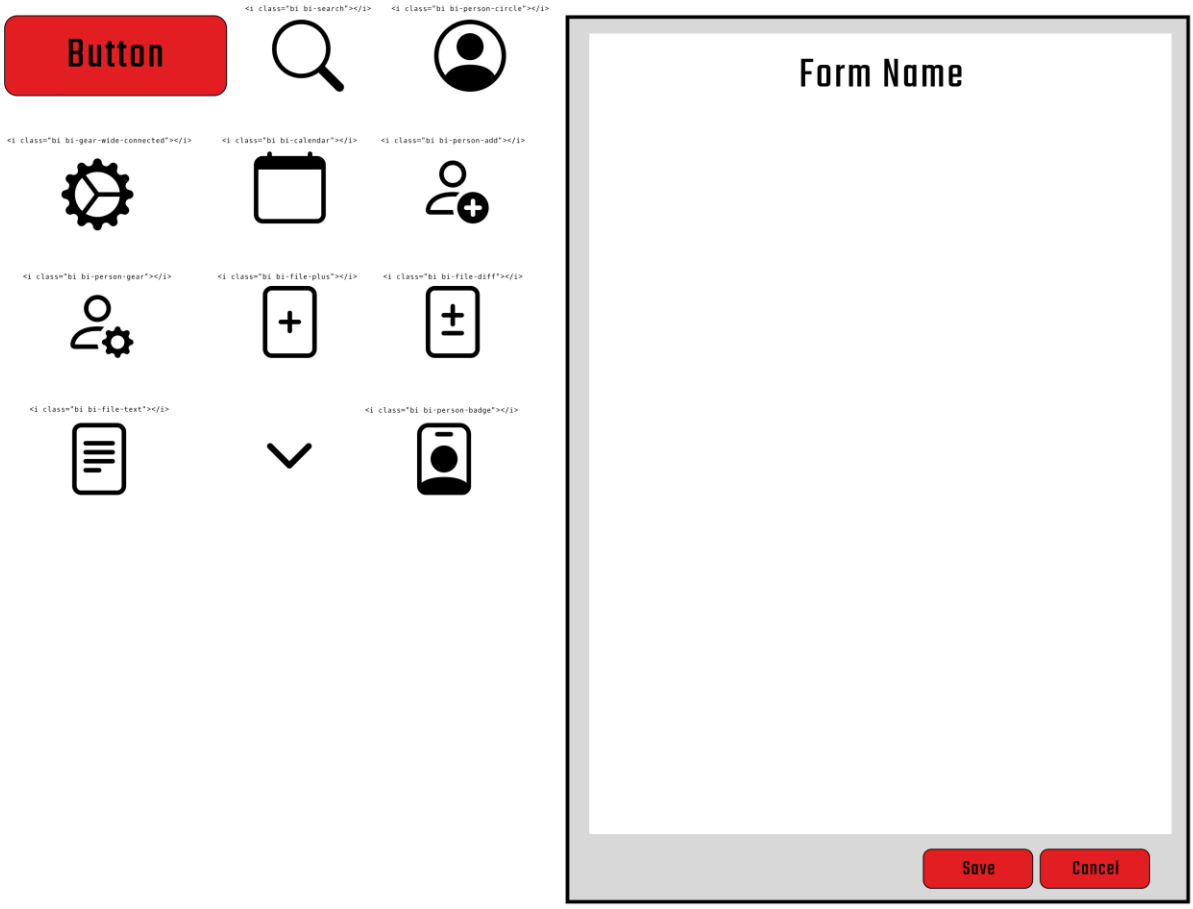



*Figure 7.1.2 – 6 UI Standards*

Figure 7.1.2 – 6 Depicts the UI standards that will be used for the system. Standard icons and buttons are displayed, as well as the default frame for forms.

# 7.2 Process Design

## 7.2.1. Physical Data Flow Diagrams

### 7.2.1.1. P 0 – Open Road Application

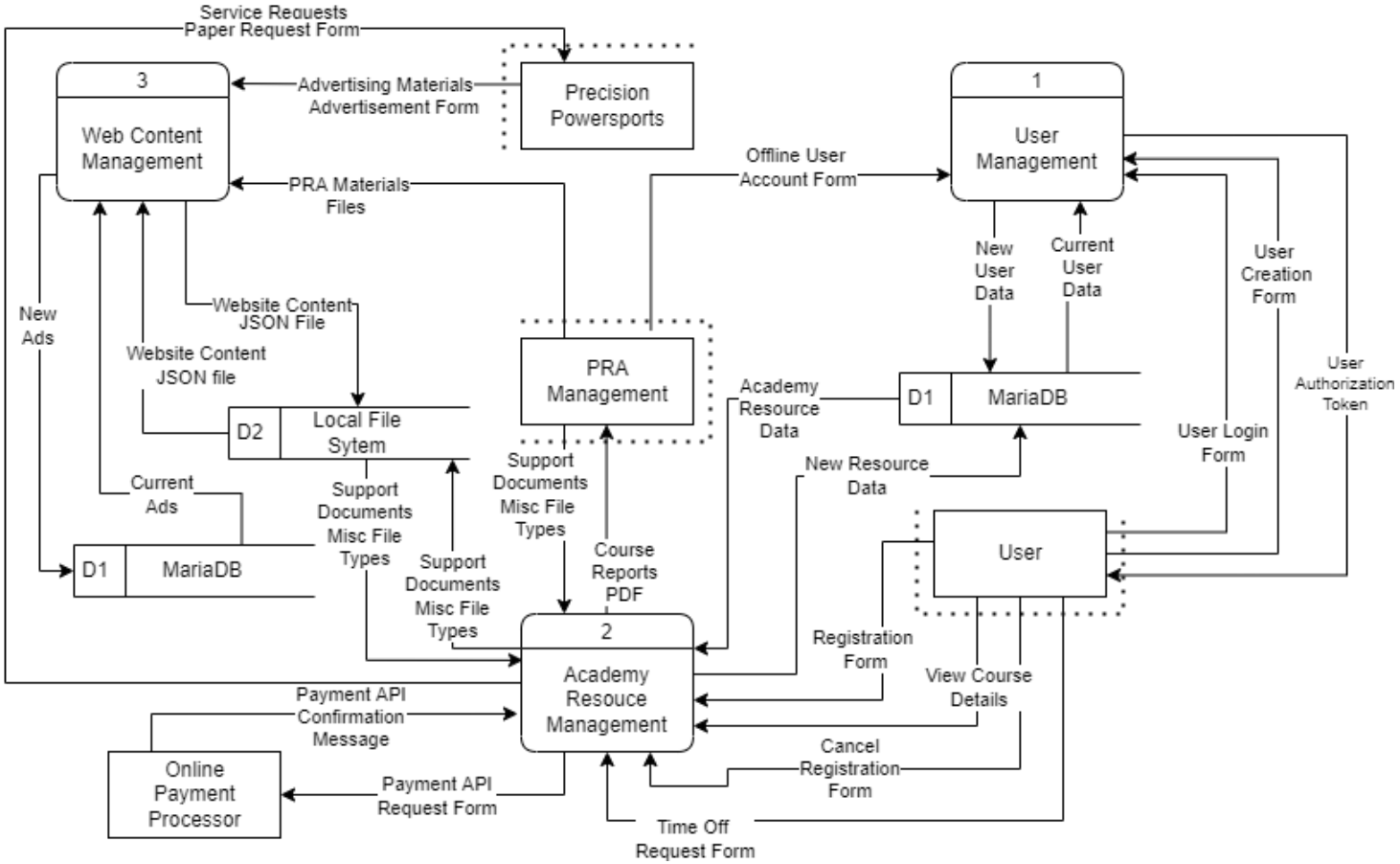



*Figure 7.2.1.1 – 1 Level 0 DFD of the system for Project Open Road*

### 7.2.1.2. P 1 – User Management

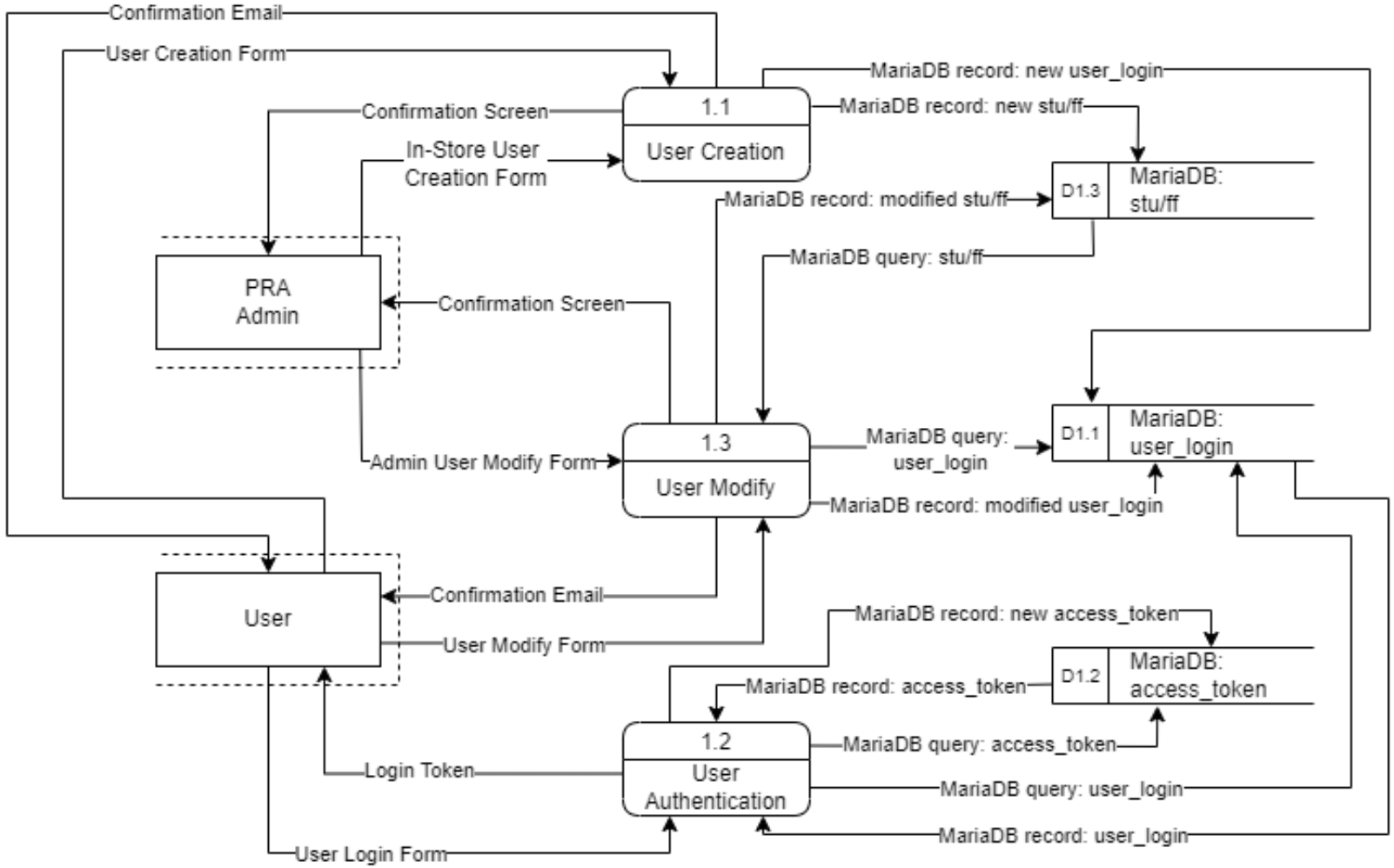



*Figure 7.2.1.2 – 1 Level 1 of Process 1 User Management*

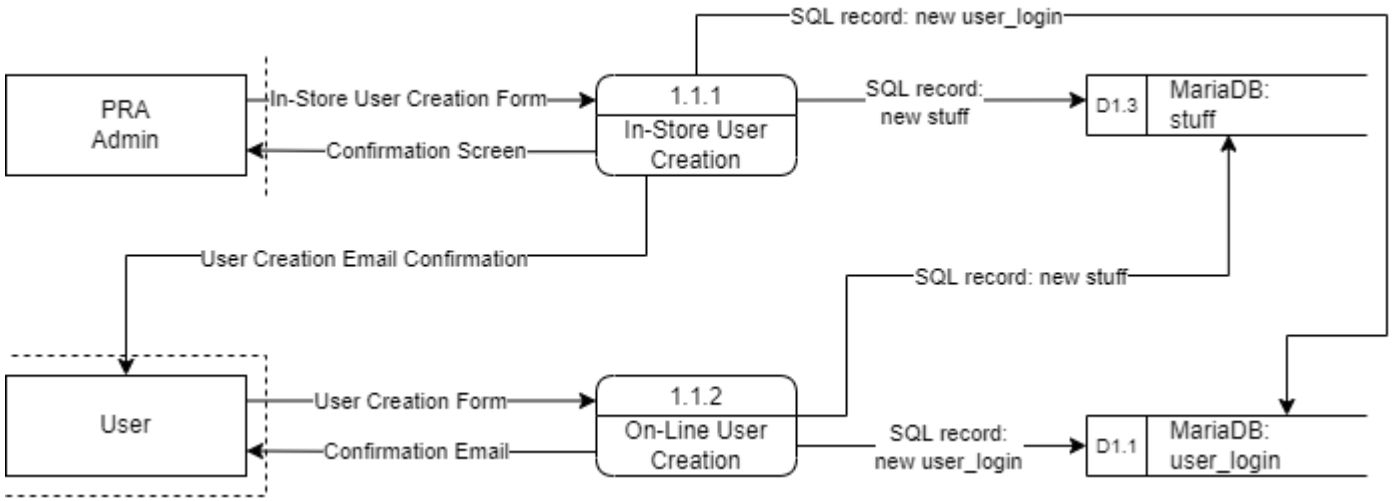



*Figure 7.2.1.2 -2 Level 2 of Process 1.1. User Creation*

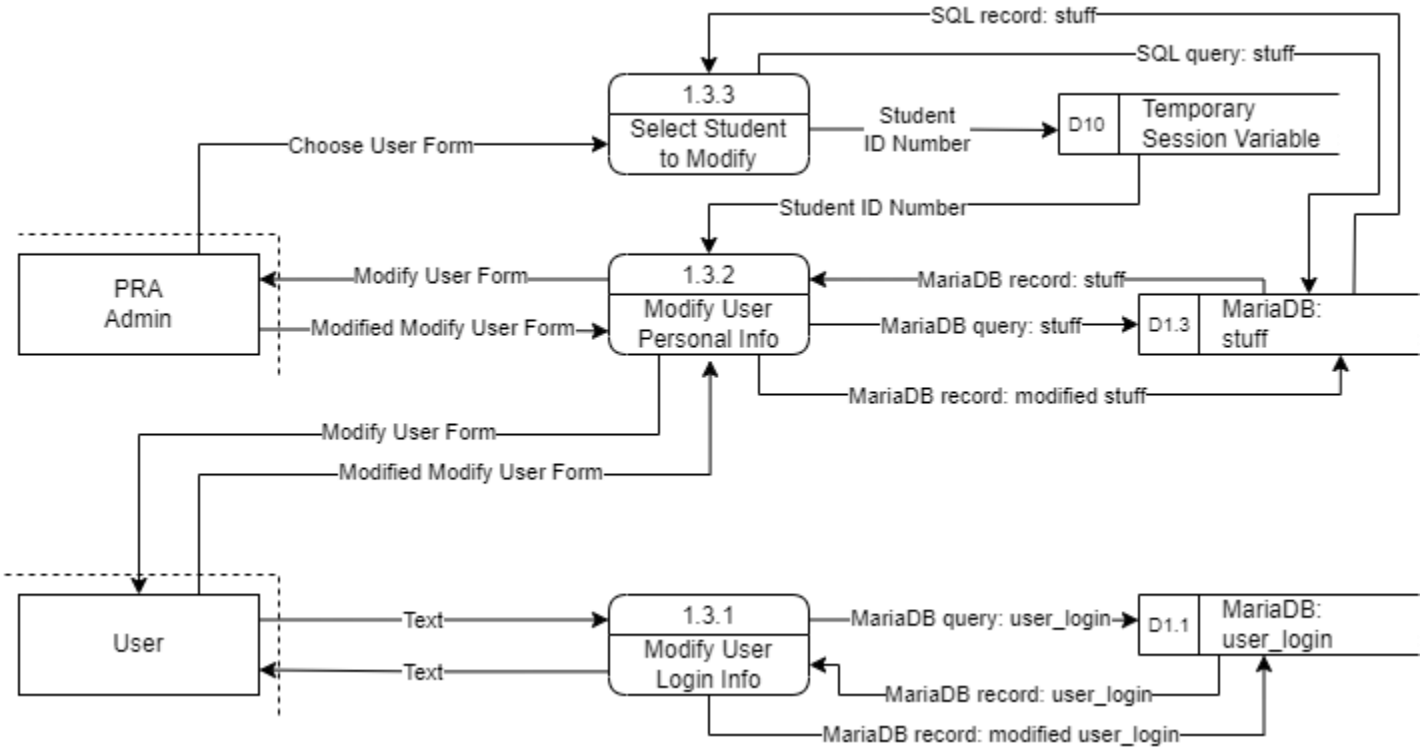



*Figure 7.2.1.3. – 4* *Level 2 of Process 1.3: User Modify*

### 7.2.1.3. P 2 – Academy Resource Management

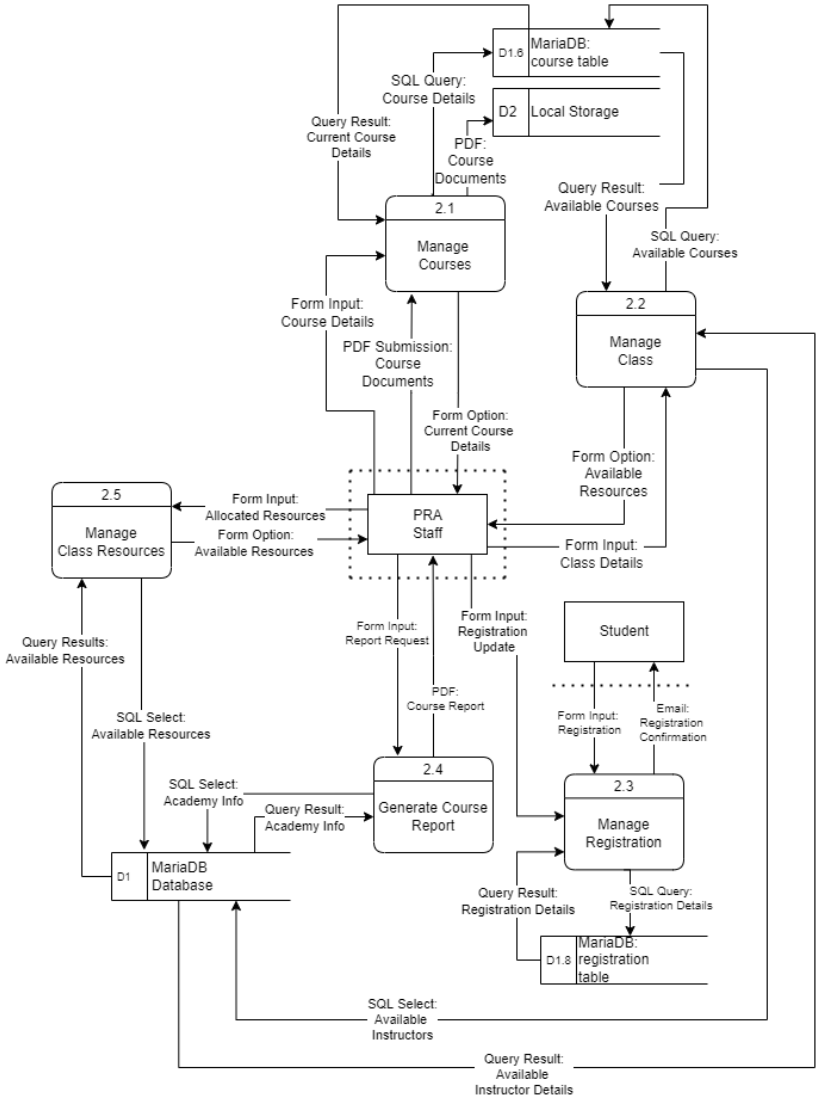



*Figure 7.2.1.3. – 1 Level 1 of Process 2: Academy Resource Management*

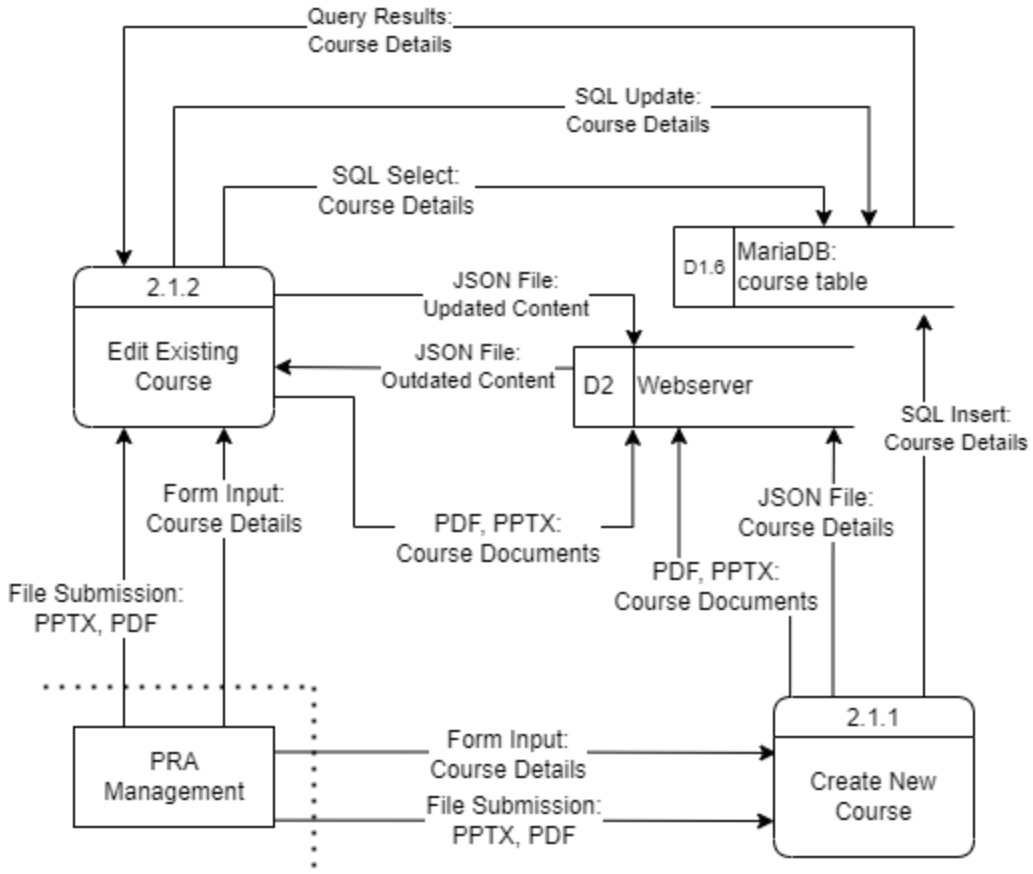



*Figure 7.2.1.3. – 2 Level 2 of Process 2.1: Manage Courses*

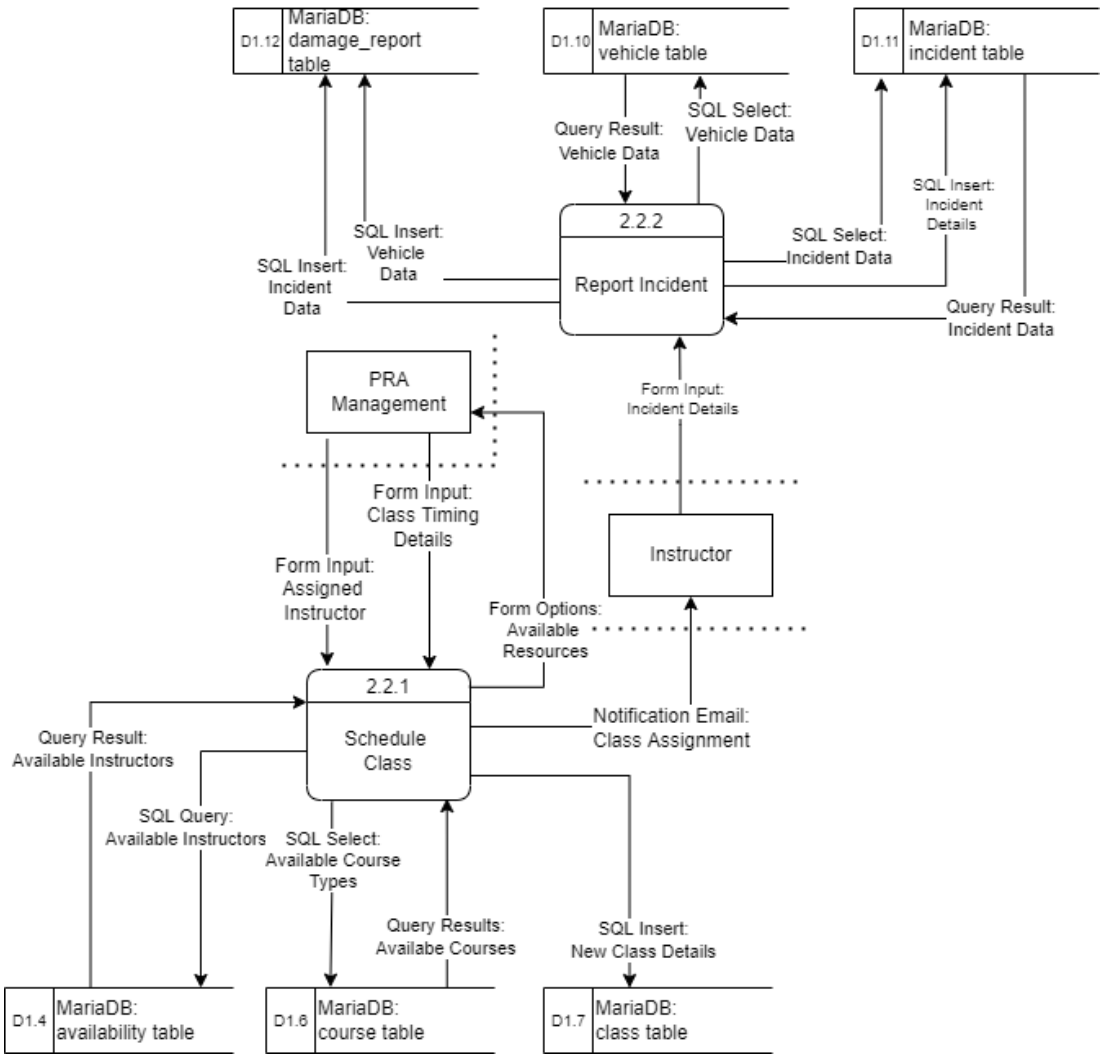



*Figure 7.2.1.3. – 3 Level 2 of Process 2.2: Manage Class*

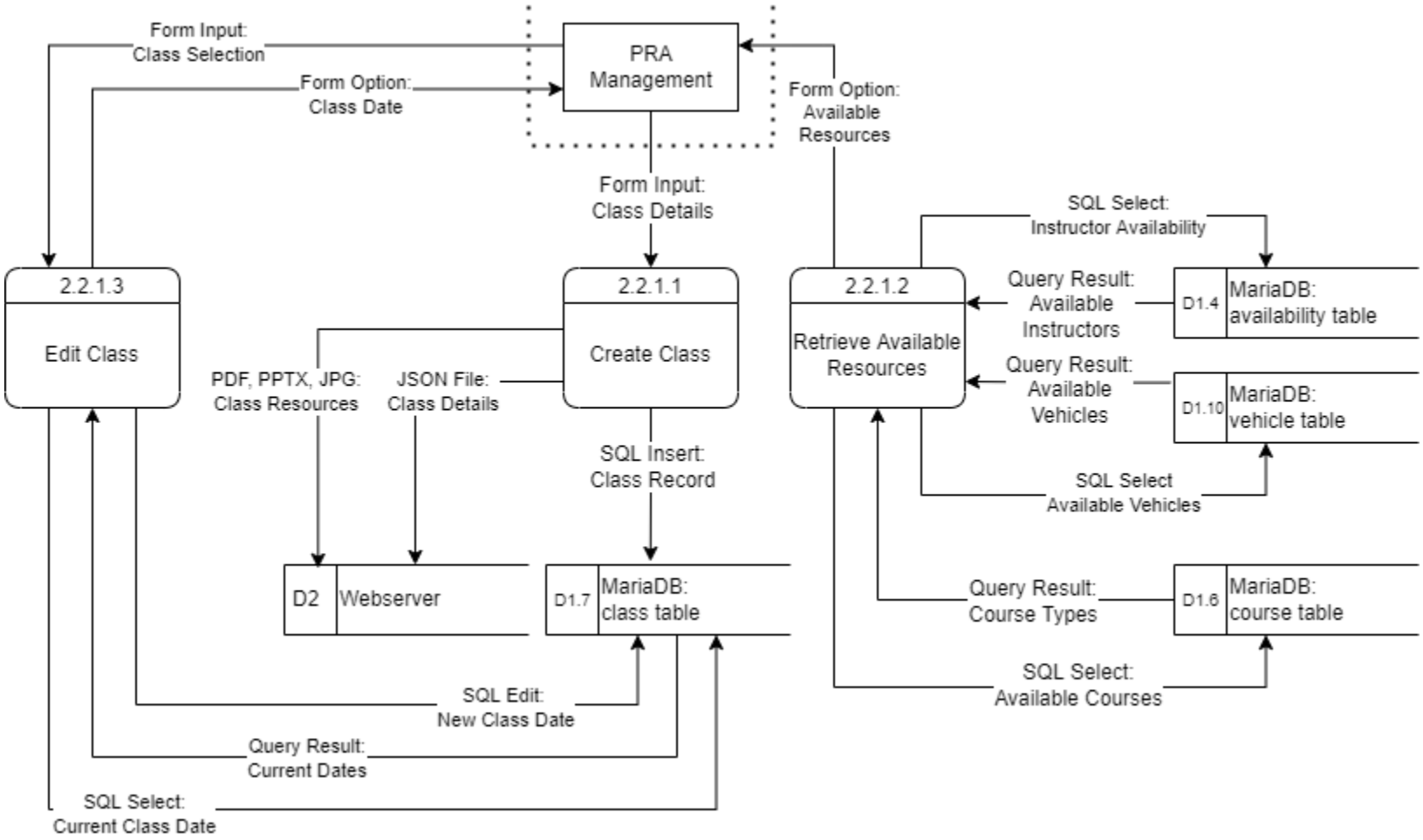



*Figure 7.2.1.3. – 4 Level 3 of Process 2.2.1: Schedule Class*

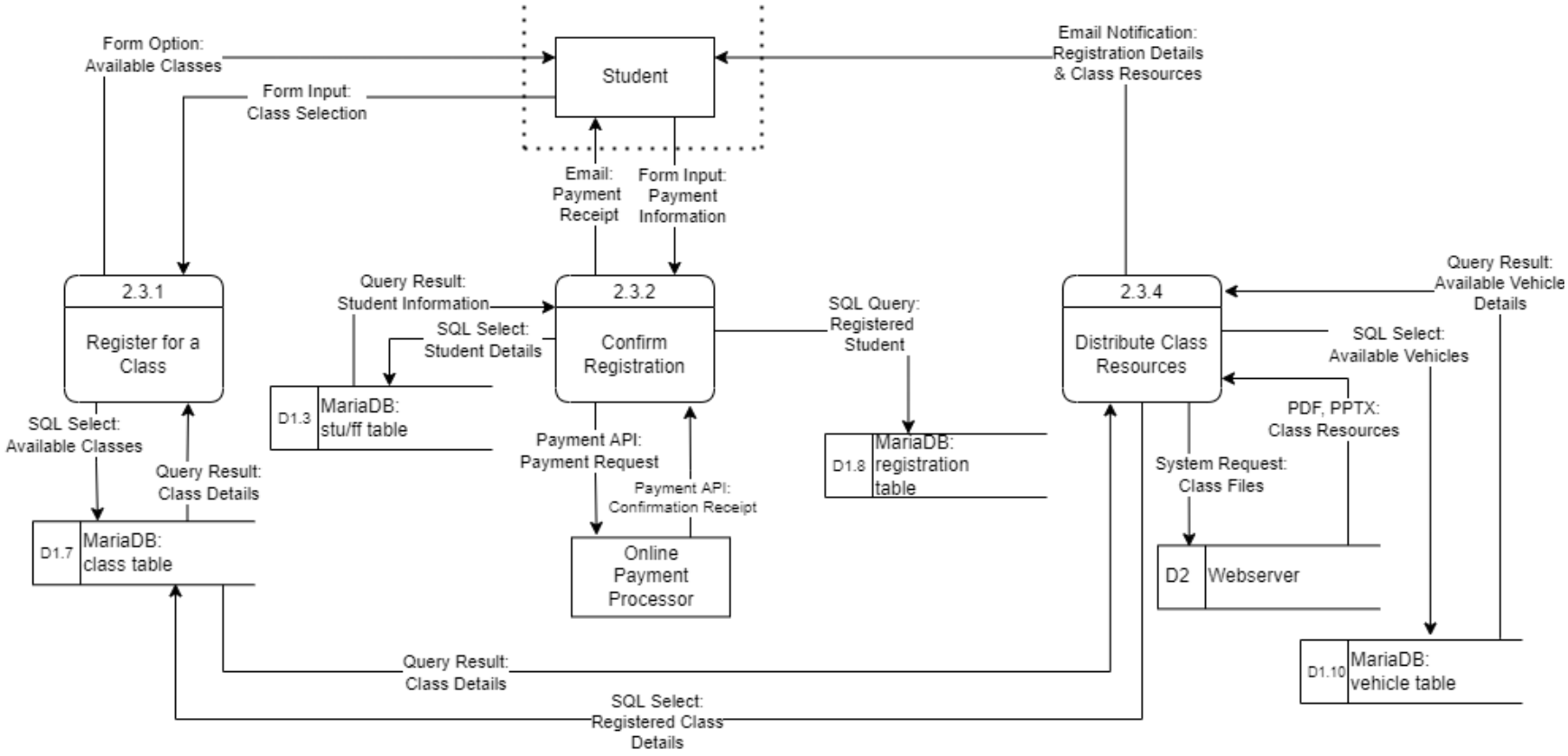



*Figure 7.2.1.3. – 5 Level 2 of Process 2.3: Manage Registration*

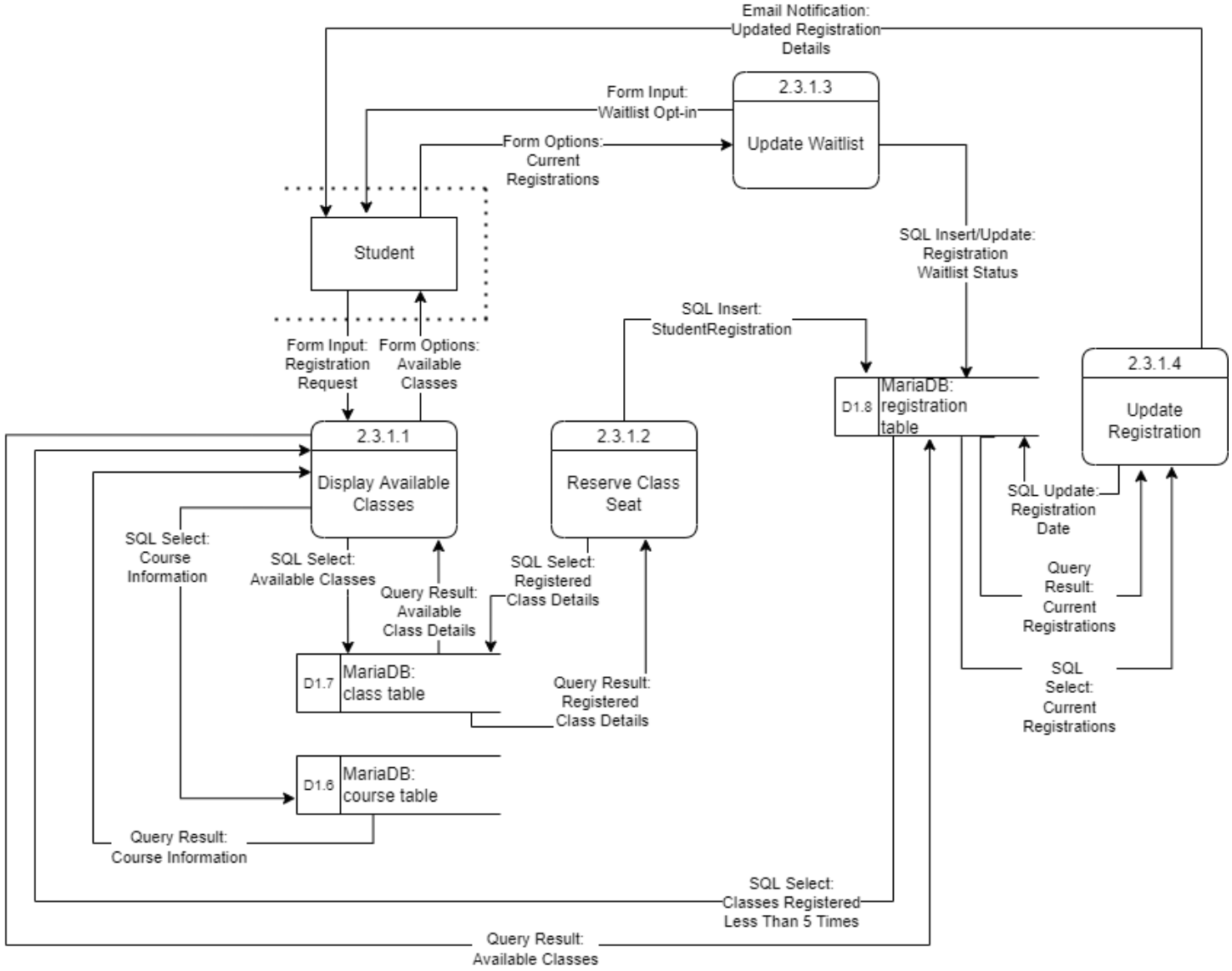



*Figure 7.2.1.3. – 6 Level 3 of Process 2.3.1: Register for a Class*

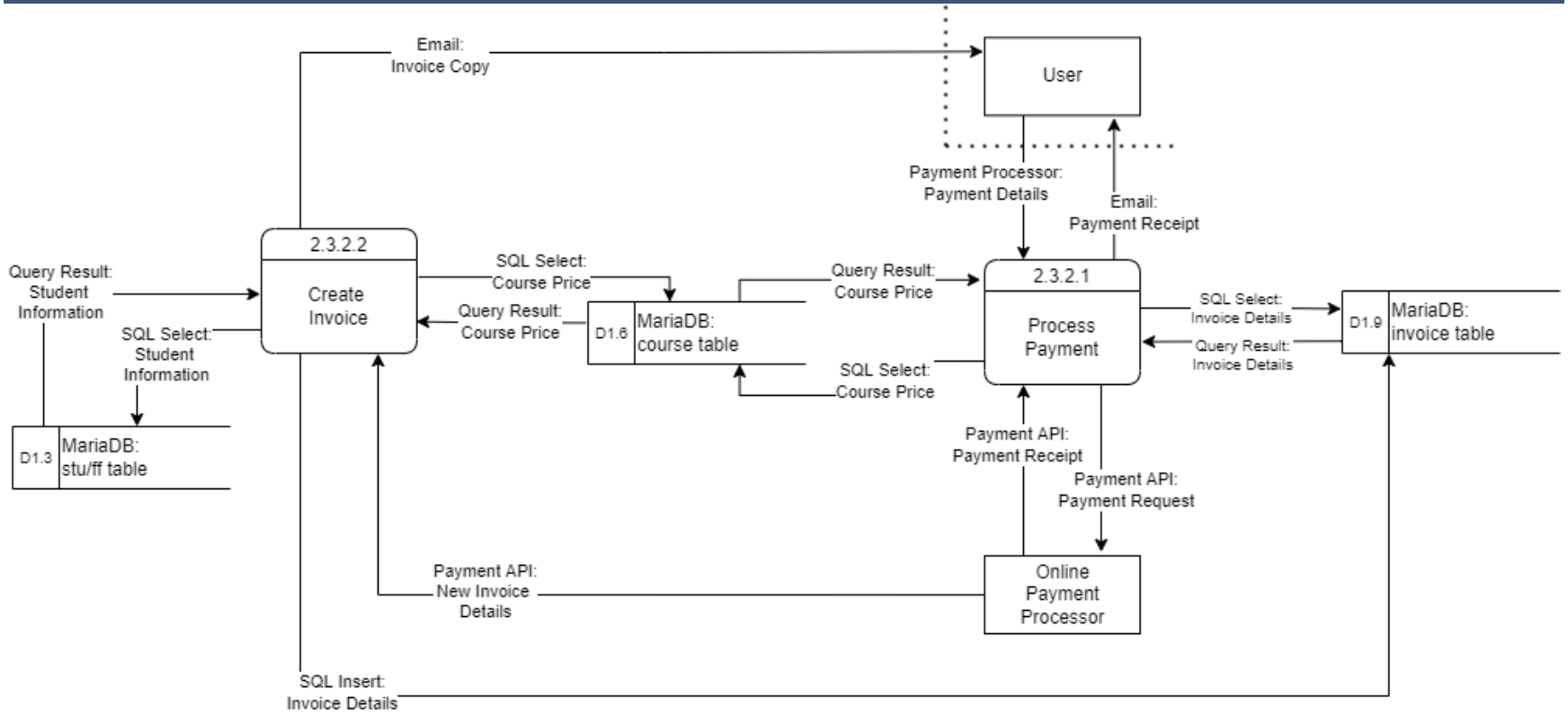



*Figure 7.2.1.3 – 7 Level 3 of Process 2.3.2: Confirm Registration*

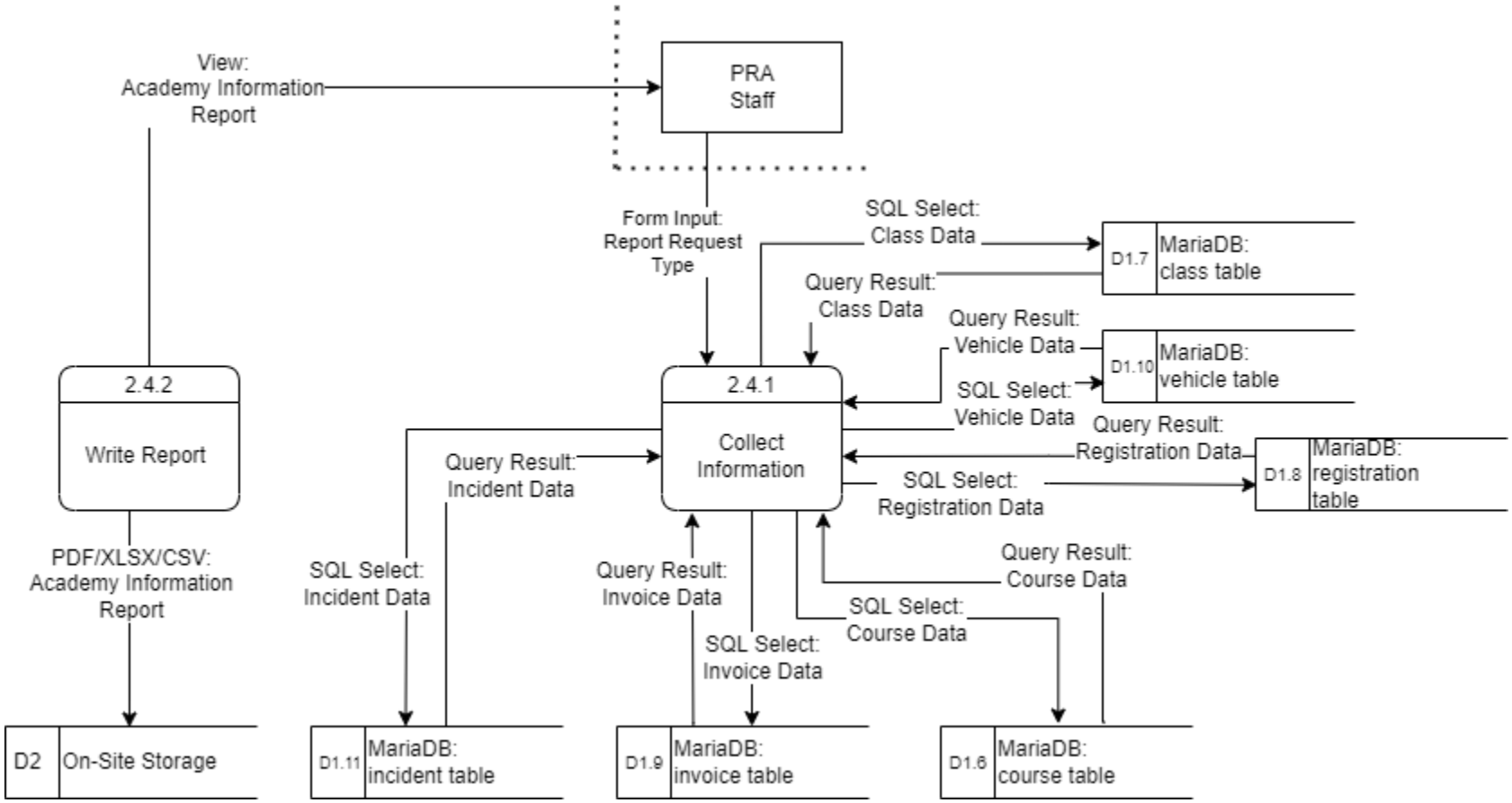



*Figure 7.2.1.3 – 8 Level 2 of Process 2.4: Generate Course Report*

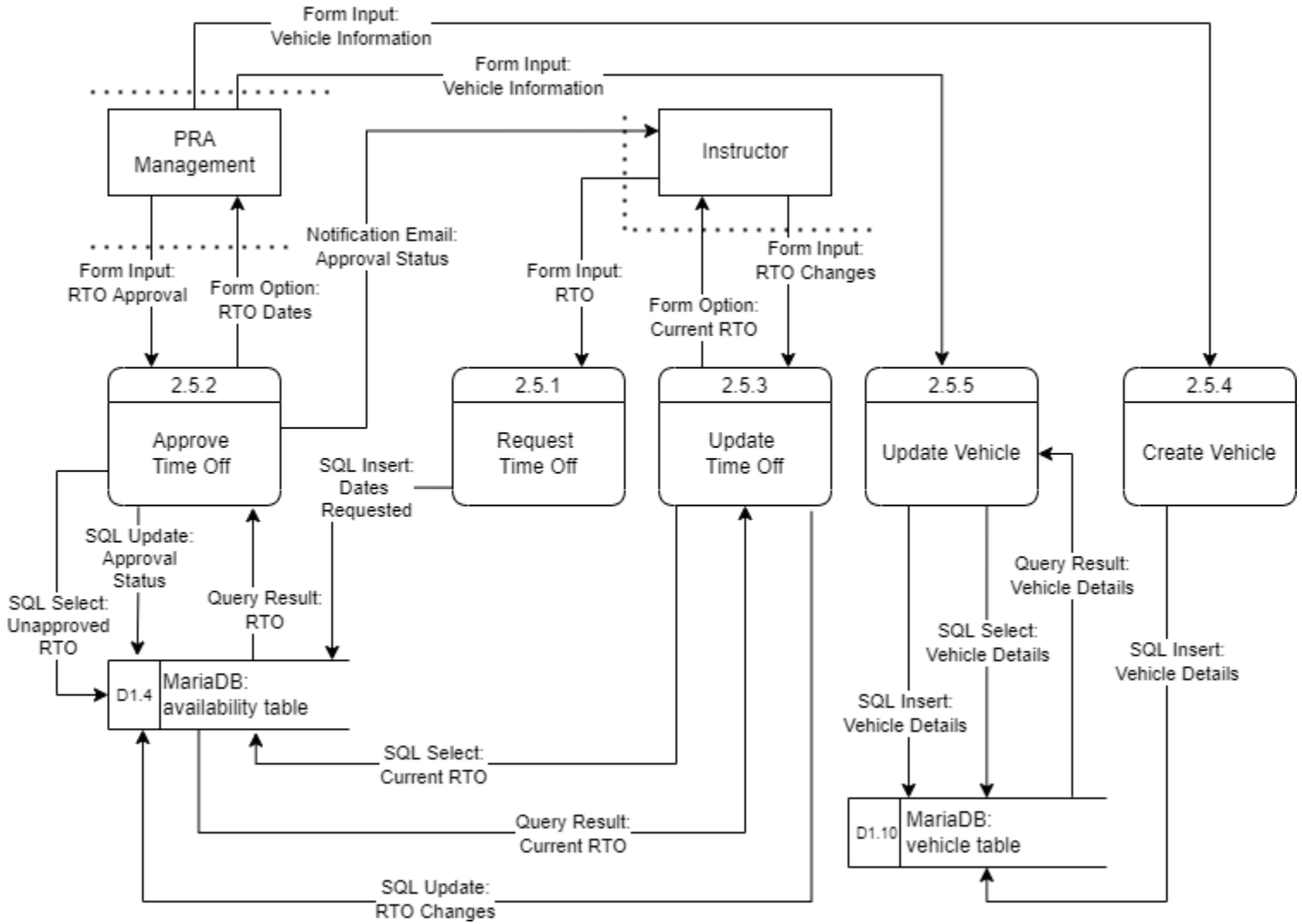



*Figure 7.2.1.3. – 9 Level 2 of Process 2.5: Manage Resources*

### 7.2.1.4. P 3 – Web Content Management

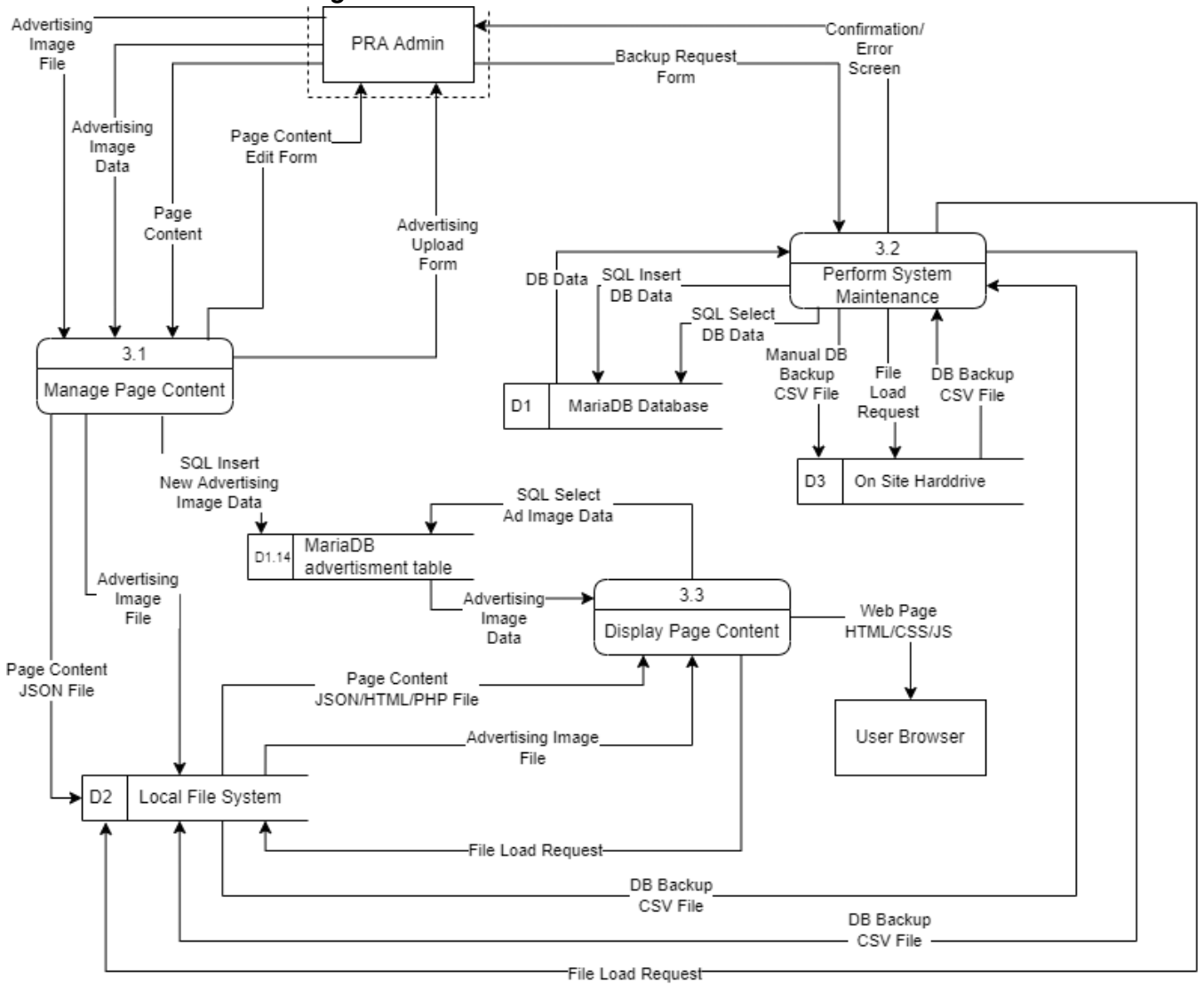



*Figure 7.2.1.4. – 1 Level 1 of Process 3: Web Content Management*

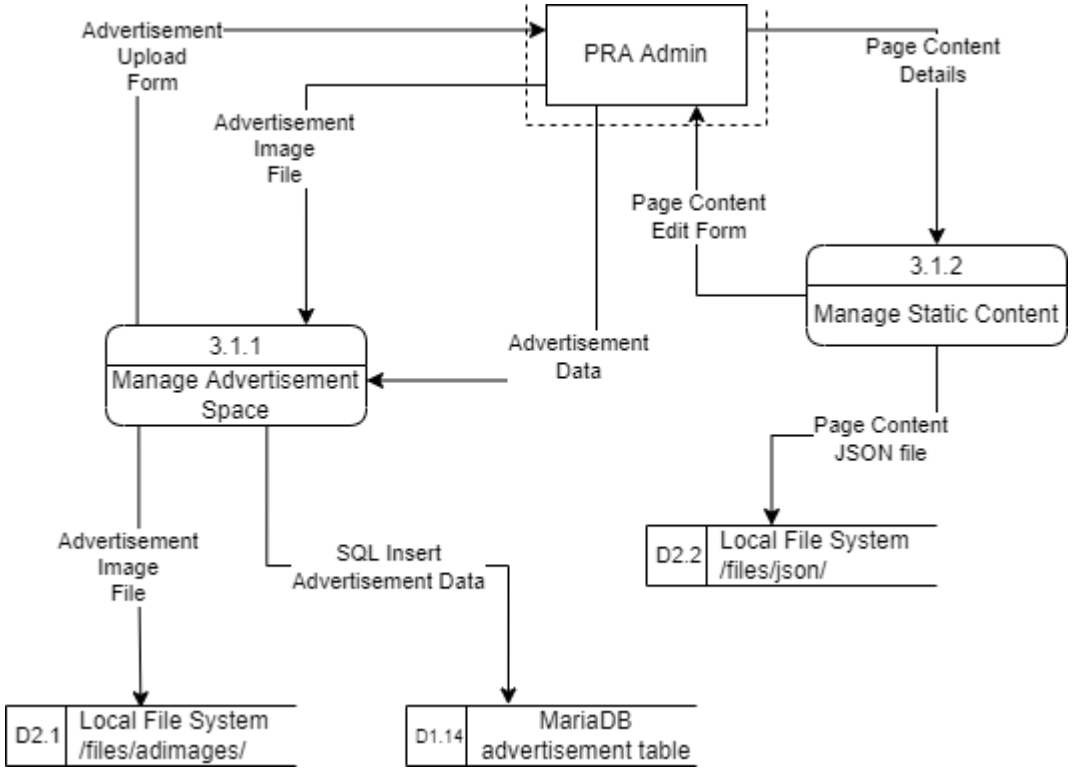



*Figure 7.2.1.4. – 2 Level 2 of Process 3.1: Manage Page Content*

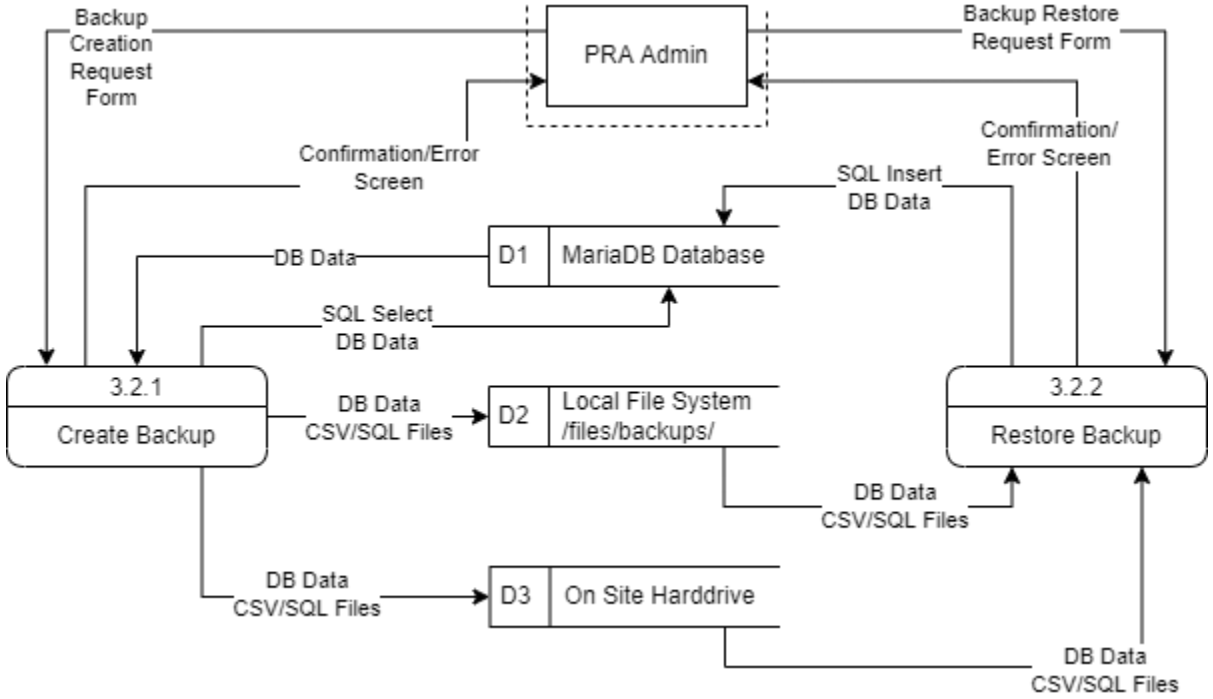



*Figure 7.2.1.4. – 3 Level 2 of Process 3.2: Perform System Maintenance*

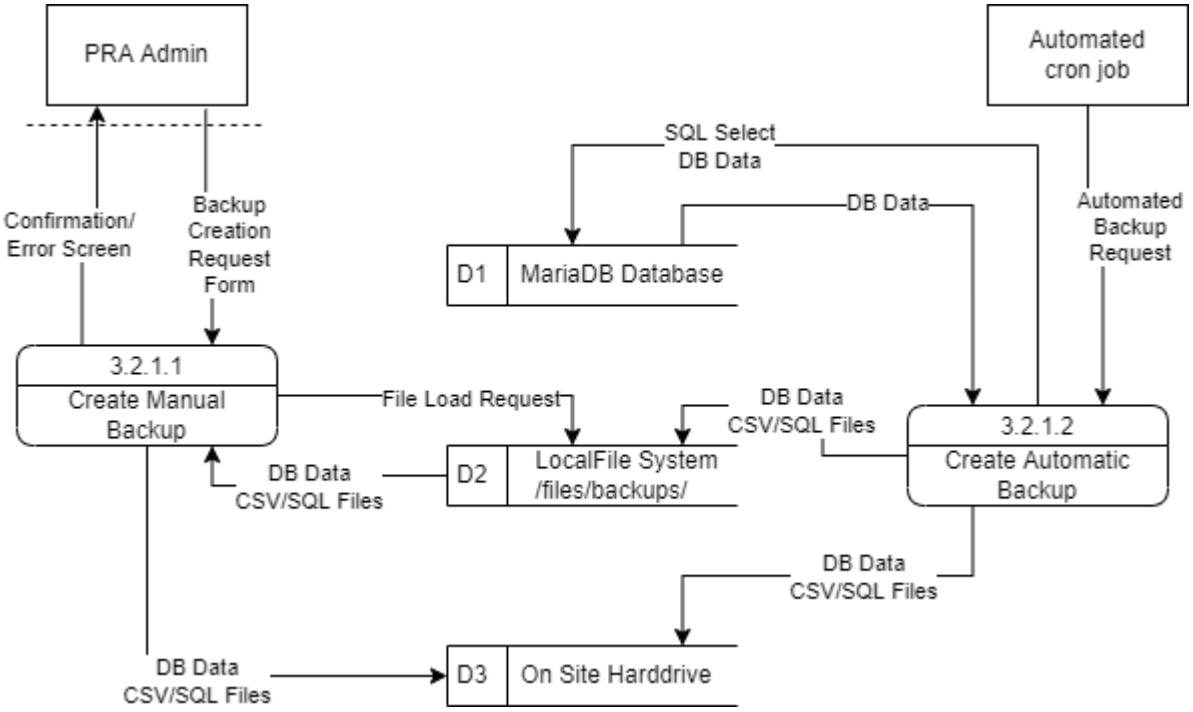



*Figure 7.2.1.4. – 4 Level 3 of Process 3.2.1: Create Backup*

### 7.2.2. Program Structure & Specifications

#### 7.2.2.1. P 1 – User Management

Process 1 User Management

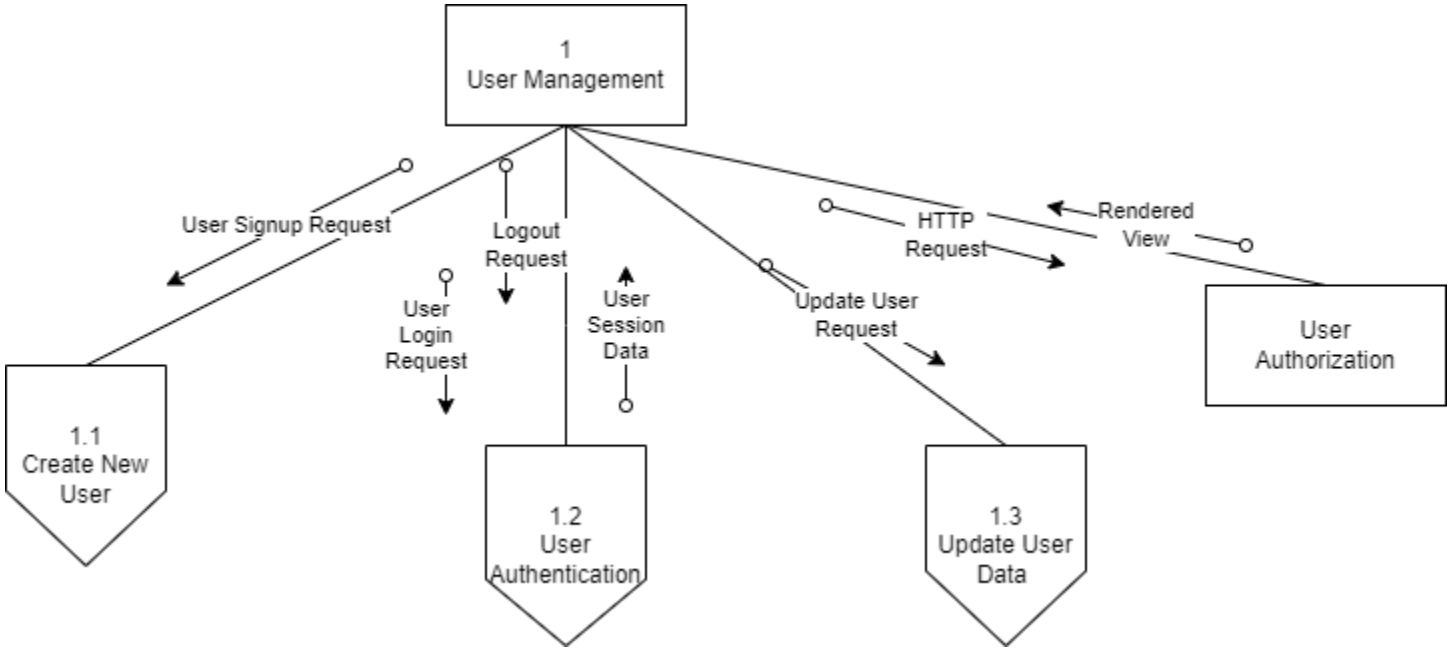



*Figure 7.2.2.1. – 1 Process 1 Structure Chart*

## Program Specifications for Process 1

| | | | |
|---|---|---|---|
| | **Program Specifications for** | "Open Road" | |
| **Module Name, number:** | 1 User Management | | |
| **Purpose:** | Provides the various methods associate with User management | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| User Signup, User Authentication/Authorization, Update User information | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| User Session Data | Server Object | Process 1.2 | |
| Rendered View | HTTP packet | Controller | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| User Signup Request | Server Object | Process 1.1 | |
| User Login/Logout Request | Server Object | Process 1.2 | |
| Update User Request | Server Object | Process 1.3 | |
| View/Process Request | HTTP Request | User Authorization | |
| | | | |
| **Pseudocode** | | | |
| if ($user is new) then send user to signup<br>if ($user is authenticated) then create new HTTP session<br>if ($user has authorization) return requested process | | | |

*Figure 7.2.2.1. - 2*

Process 1.1 Create New User

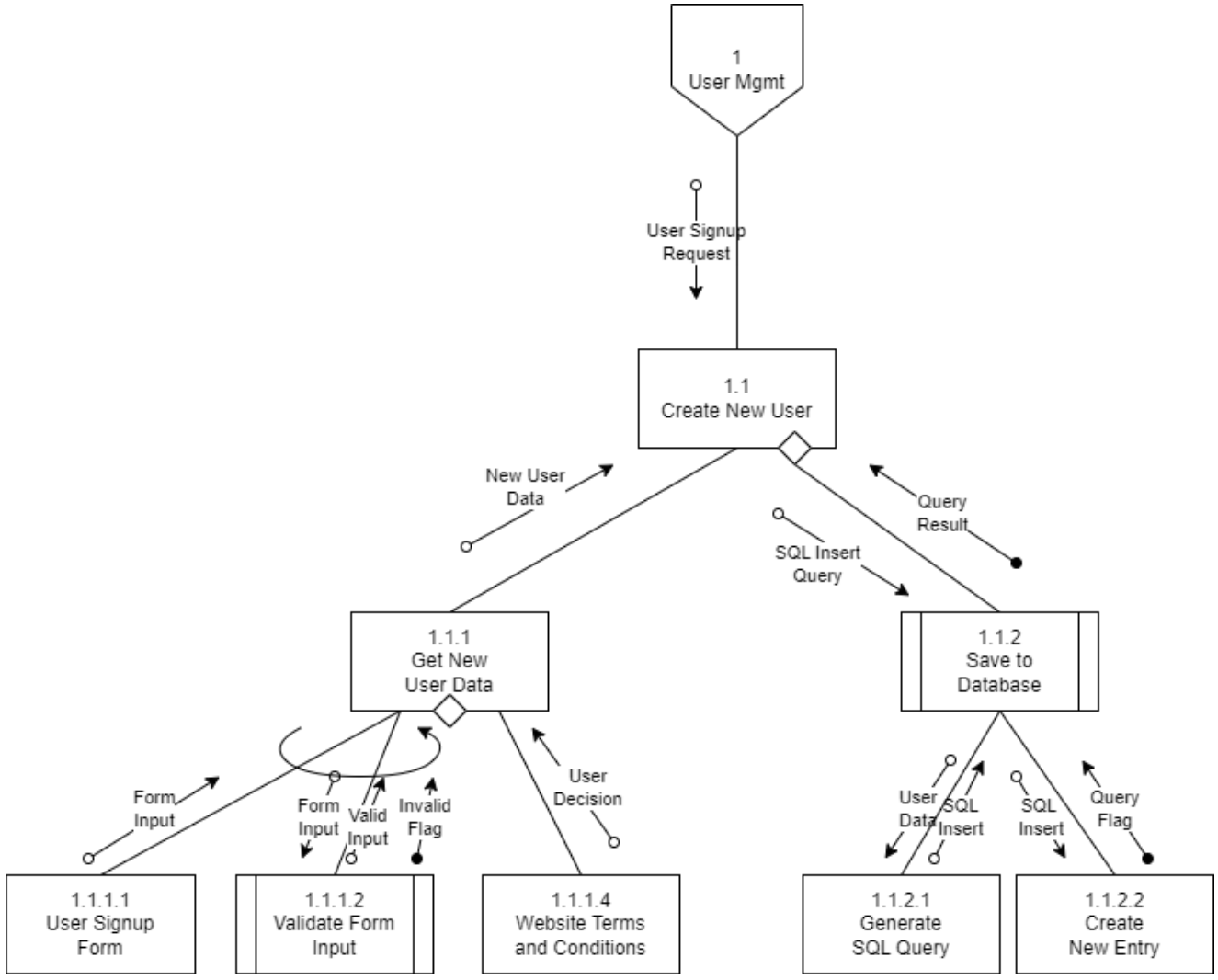



*Figure 7.2.2.1. – 3*

## Program Specifications for Process 1.1

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.1 Create New User | | |
| **Purpose:** | Handles the creation of new user accounts | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | HTML/CSS/PHP/SQL | | |
| **Events:** | | | |
| returns user signup form to client device | | | |
| collects and validates new user data | | | |
| creates and sends SQL query to database | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| New User Data | Various | Process 1.1.1 | |
| Query Result | sysFlag | Process 1.1.2 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| New User Data | Various | Process 1.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| return user_signup view<br>get and validate form data<br>save new user to database | | | |

*Figure 7.2.2.1. – 4*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.1.1 Get New User Data | | |
| **Purpose:** | Accepts and validates new user data | | |
| **Programmer:** | | | |
| **Date due:** | | | |
| **Language/platform:** | | | |
| **Events:** | | | |
| User presses 'Signup' or 'Register' button | | | |
| User submits information using signup form | | | |
| User accepts/rejects site Terms and Conditions | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Form Input | Varchar | Process 1.1.1.1 | |
| Valid Input | Varchar | Process 1.1.1.2 | optional sysFlag |
| Yes/No Flag | Boolean | Process 1.1.1.3 | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| New User Data | Various | Process 1.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| return('user/signup', $user_signup)<br>$request.validate($user_signup)<br>return('user/terms_conds', $user_signup) | | | |

*Figure 7.2.2.1. -5*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.1.1.1 - User Signup Form | | |
| **Purpose:** | Input Form to collect new User Data | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | Laravel View (PHP/HTML/CSS) | | |
| **Events:** | | | |
| Trigger - User presses "Signup" or "Register" button | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| User Information | Various | User | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Form Input | Varchar | 1.1.2 - Save to Database | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
<input type="text" name="f_name">
<input type="text" name="l_name">
...
<input type="submit">
```

*Figure 7.2.2.1. – 6*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.1.1.2 - Validate Form Input | | |
| **Purpose:** | Ensure data entered in form inputs is correct format and length | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | Javascript/PHP | | |
| **Events:** | | | |
| Trigger - user types information into form inputs | | | |
| OR | | | |
| Trigger - user presses Submit button on input form | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Form Input | Varchar | 1.1.1.1 - User Signup Form | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Valid Input | Varchar | 1.1.2 - Save to Database | |
| Invalid Flag | System Flag | 1.1.1.3 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
If $request is valid
$user = /**...*/
return to_route('user.profile', ['user' => $user->id]);
```

*Figure 7.2.2.1. -7*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.1.1.3 - Website Terms and Conditions Form | | |
| **Purpose:** | Informs user of sites terms and conditions and requires user acceptance/decline | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | HTML/CSS/PHP | | |
| **Events:** | | | |
| System identified invalid input during form validation | | | |
| Offers retry/cancel Signup options to user | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| User decision | Boolean | User | If false, no user account created |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Yes/No flag | Boolean | Process 1.1.1.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
|  Terms and conditions… <br><input type='radio' name='termsAccept' value='accept' /><br><input type='submit' name='submit' value='submit' /><br><input type='button' name='cancel'>Cancel</input> | | | |
| **Other** | | | |
| User will not be allowed to proceed with signup if they do not accept site terms and conditions. No user account wil be created if they cancel the signup process as a result of not accepting terms and conditions. | | | |

*Figure 7.2.2.1. – 8*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.1.2 Save to Database | | |
| **Purpose:** | Creates SQL INSERT query and sends transaction to DB | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | PHP/SQL | | |
| **Events:** | | | |
| Receives new user data to be input into database as new user entry | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| New User Data | Various | Process 1.1.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Query Result | SysFlag | Process 1.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Generate SQL query (New User Data)<br>Send to Database<br>if (!query)<br>  noitfy user | | | |
| **Other** | | | |
| | | | |

*Figure 7.2.2.1.- 9*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.1.2.1 Generate SQL Query | | |
| **Purpose:** | Builds SQL query for database transaction | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | PHP/SQL | | |
| **Events:** | | | |
| Receives new $user data to send to database | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| New User Data | Various | Process 1.1.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| SQL INSERT Query | SQL Query | Process 1.1.2.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| QueryBuilder.insert($user).into($database->user) | | | |

*Figure 7.2.2.1. – 10*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.1.2.2 Create New Entry | | |
| **Purpose:** | Submits SQL Query to write new entry in DB | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | PHP/SQL | | |
| **Events:** | | | |
| Sends SQL query to database cache and logs database transaction | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| SQL Query | Query | Process 1.1.2.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Query Flag | sysFlag | Process 1.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| send($query).to($database) | | | |

*Figure 7.2.2.1. – 11*

Process 1.2 User Authentication

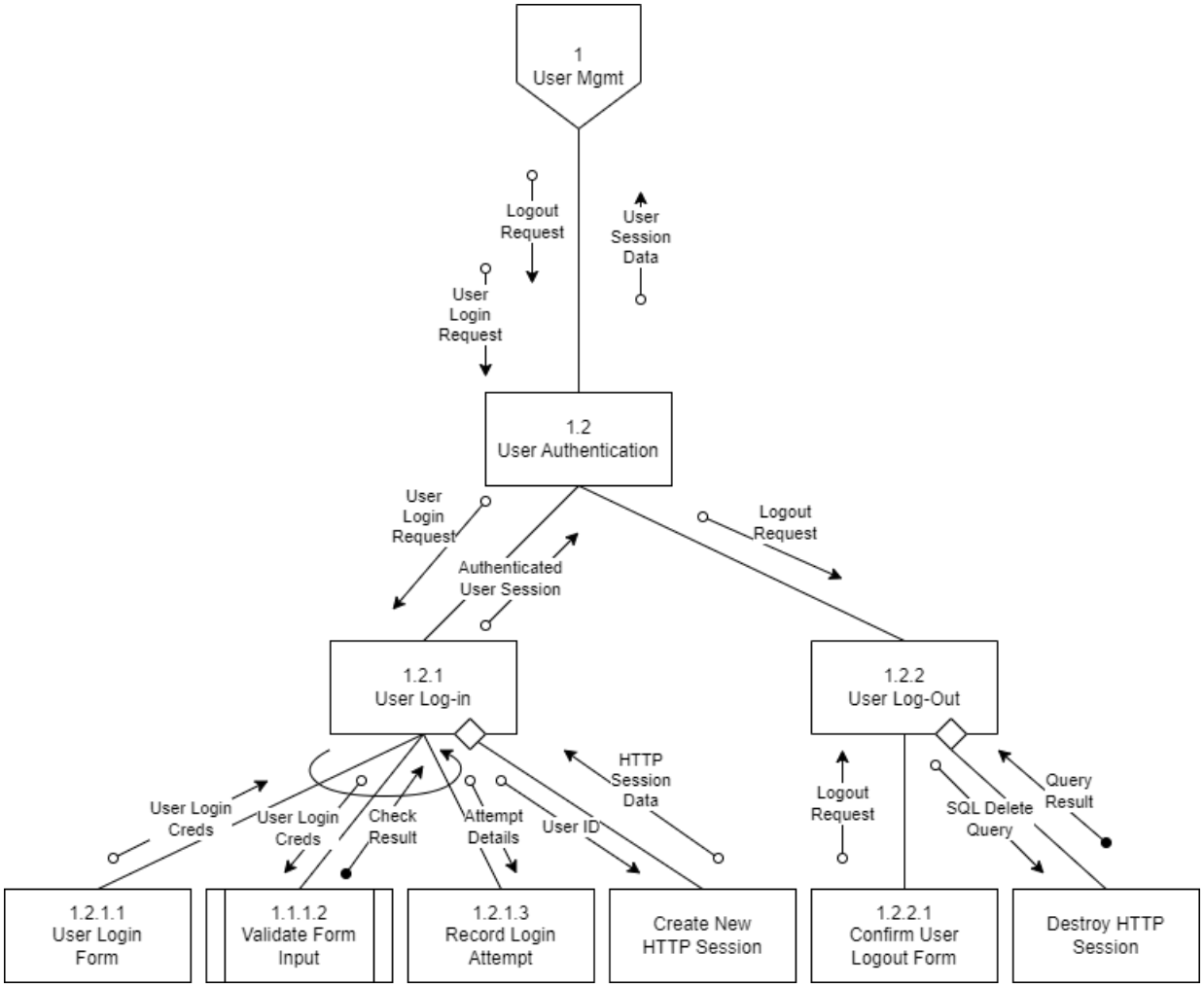



*Figure 7.2.2.1. - 12 Process 1.2 Structure Chart*

## Process 1.2 Program Specifications

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.2 User Authentication | | |
| **Purpose:** | Verifies user identity and creates/destroys HTTP sessions | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| authenticates or denies a user login request | | | |
| performs proccesses to properly logout users from active sessions | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| User Login Request | HTTP request | User | |
| User Logout Request | HTTP request | User | |
| Authd User Session | Session Object | System | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| User Login Request | HTTP request | Process 1.2.1 | |
| User Logout Request | HTTP request | Process 1.2.2 | |
| HTTP Session Data | Cookie | User/Process 1 | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
return('user/login') view
if user is authenticated
   create new session
return('user/logout') view
    destroy user session
```

*Figure 7.2.2.1. – 13*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.2.1 User Login | | |
| **Purpose:** | collects and verifies user identity to allow entry into web app | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Collects login credentials and verifies them | | | |
| creates new HTTP session | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| User Login Request | HTTP request | Process 1.2 | |
| User Login Creds | Various | Process 1.2.1.1 | |
| Check Flag | sysFlag/Boolean | Process 1.1.1.2 | |
| HTTP Session Data | Various | HTTP Session module | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| User Login Creds | Various | Process 1.1.1.2 | |
| Login Attempt Details | StringText | Process 1.2.1.3 | |
| User ID | Varchar | HTTP Session module | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
get user $creds
if (auth($creds))
    create new session
else
    return authError($creds)
log($attempt)
```

*Figure 7.2.2.1. – 14*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.2.1.1 User Login Form | | |
| **Purpose:** | Collect user login creds, has links for resetting user login creds or user signup | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | HTML/CSS/JavaScript/PHP | | |
| **Events:** | | | |
| User pressed login button | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| User Login Creds | varchar | Process 1.1.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
<input type="text" name="user_email" />
<input type="password" name="user_passwd" />
<input type="submit" value="Login" />
<a href="/reset_user_pass">Forgot your password?</a>
```

| **Other** |
|---|
| View comes as a resources with Laravel/Jetstream module, just needs to be restyled to match our UI. |

*Figure 7.2.2.1. – 15*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.2.1.3 Record Login Attempt | | |
| **Purpose:** | Logs successful/unsuccessful login attempts for error checking/security purposes | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| User attempts to login and is either successful or unsuccessful | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Login Attempt details | String/Text | Process 1.2.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Login Attempt log | String/Text | System | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
log($user, $request->ip, $request->timestamp)
```

| **Other** |
|---|
| This log will be stored as a text file on the server's local file system. |

*Figure 7.2.2.1. – 16*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.2.2 User Logout | | |
| **Purpose:** | Provides the user method of invoking logout feature, and confirms user intent | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| User presses logout button | | | |
| System returns logout_confirm form | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Logout Request | sysFlag | UI | |
| Redis Result | sysFlag | Destroy HTTP Session | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Logout Confirm View | partial view | Process 1.2.2.1 | |
| Redis Query | Server Object | Destroy HTTP Session | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
<input type="button" value="Logout" />
```

| **Other** |
|---|
| Logout proccess is a feature included in the Laravel/Jetstream module. |

*Figure 7.2.2.1. – 17*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.2.2.1 Confirm User Logout Form | | |
| **Purpose:** | confirms whether user intends to logout or not | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | HTML/CSS/JavaScript/PHP | | |
| **Events:** | | | |
| User confirm logout | | | |
| User cancels logout | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Logout Request | Server Object | Process 1.2.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
<p>Are you sure you want to logout?</p>
<input type="submit" value="Logout" />
<input type="submit" value="Cancel" />
```

| **Other** |
|---|
| View is included as a resource with the Laravel/Jetstream module. Should only need re-styling to match UI. |

*Figure 7.2.2.1. – 18*

Process 1.3 Update User Data

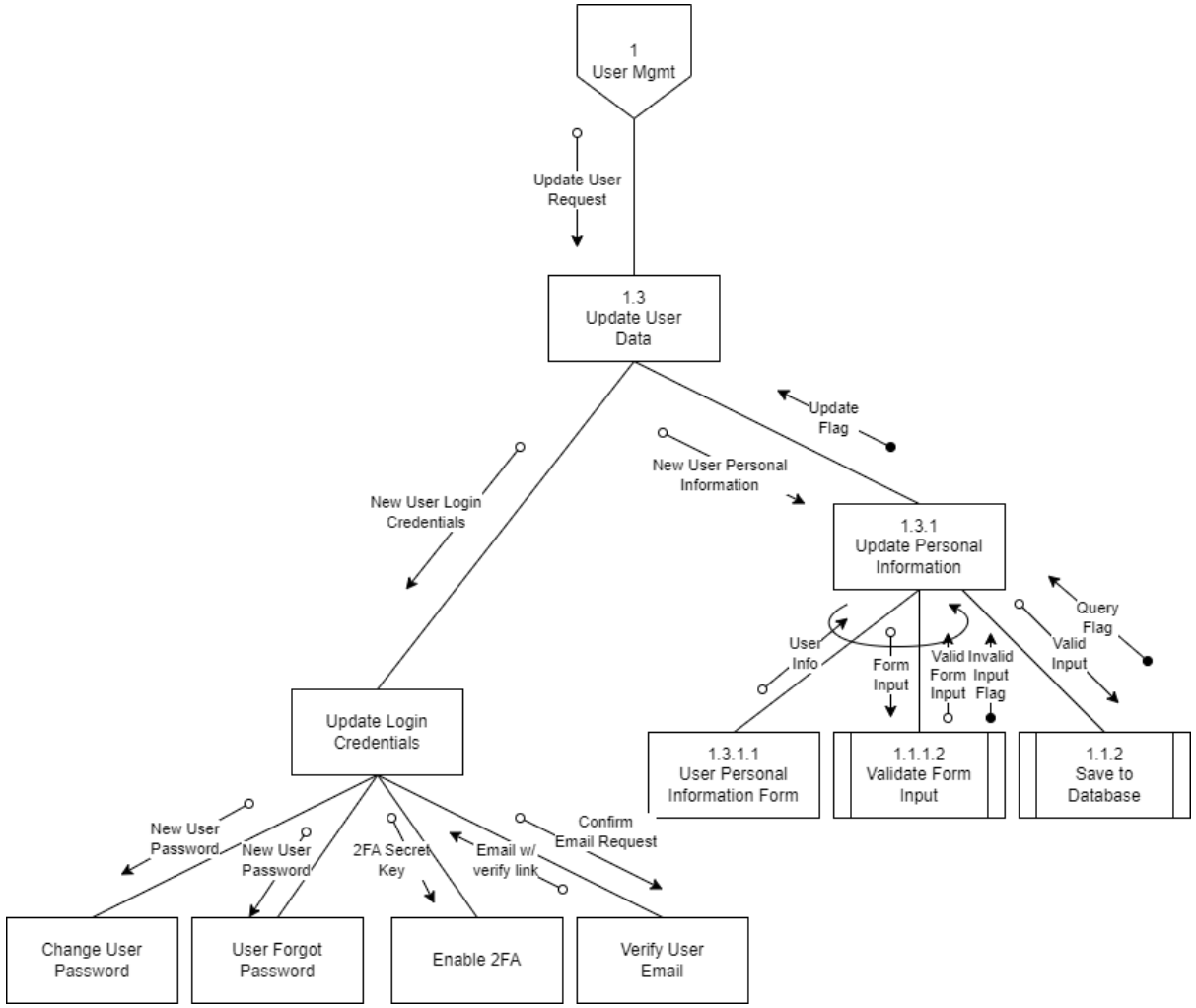



*Figure 7.2.2.1. – 19*

## Program Specifications for Process 1.3

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.3 Update User Data | | |
| **Purpose:** | lets user invoke methods to update auth credentials or personal information | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | HTML/CSS/JavaScript/PHP | | |
| **Events:** | | | |
| User invokes method to update password | | | |
| User invokes method to reset forgotten password | | | |
| User invokes method to change email associated with account | | | |
| User invokes method to enable 2FA | | | |
| User invokes method to verify email associated with account | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Update User Request | Server Object | Process 1 | |
| Update Flag | sysFlag | Process 1.3.1 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Update Login Creds Request | Server Object | Update Login Creds | |
| Update User Info Request | Server Object | Process 1.3.1 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
if($user is athorized)return update login credentials process
if($ user is authorized) return account details process
```

*Figure 7.2.2.1. – 20*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number:** | 1.3.1 | | |
| **Purpose:** | Provides methods for user to change personal information | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | HTML/CSS/JavaScript/PHP | | |
| **Events:** | | | |
| User access personal information form in account details area of application | | | |
| input format alert | | | |
| Successful save alert | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| New User Information | Various | User | |
| Valid form input | Server Object | Process 1.1.1.2 | |
| Query Flag | sysFlag | Process 1.1.2 | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Update Flag | sysFlag | Process 1.3 | |
| Form Input | Server Object | Process 1.1.1.2 | |
| Valid Input | Server Object | Process 1.1.2 | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
<input type="button" value="Account Details" href="/user_pers_info" />
```

*Figure 7.2.2.1. – 21*

| | | | |
|---|---|---|---|
| | **Program Specifications for** | "Open Road" | |
| **Module Name, number:** | 1.3.1.1 User Personal Information Form | | |
| **Purpose:** | Shows currently stored data and collects data to be update, if any | | |
| **Programmer:** | Tim Streibel | | |
| **Date due:** | | | |
| **Language/platform:** | HTML/CSS/JavaScript/PHP | | |
| **Events:** | | | |
| User accesses account details | | | |
| User selects section of form to edit data | | | |
| User submits/cancels form | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| New User Info | various | Process 1.1.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
return($user as $key->value)
button to edit info
button to submit changes
button to cancel changes
```

*Figure 7.2.2.1. - 22*

## 7.2.2.2. P 2 – Academy Resource Management

### Process 2.1 Edit Course Types

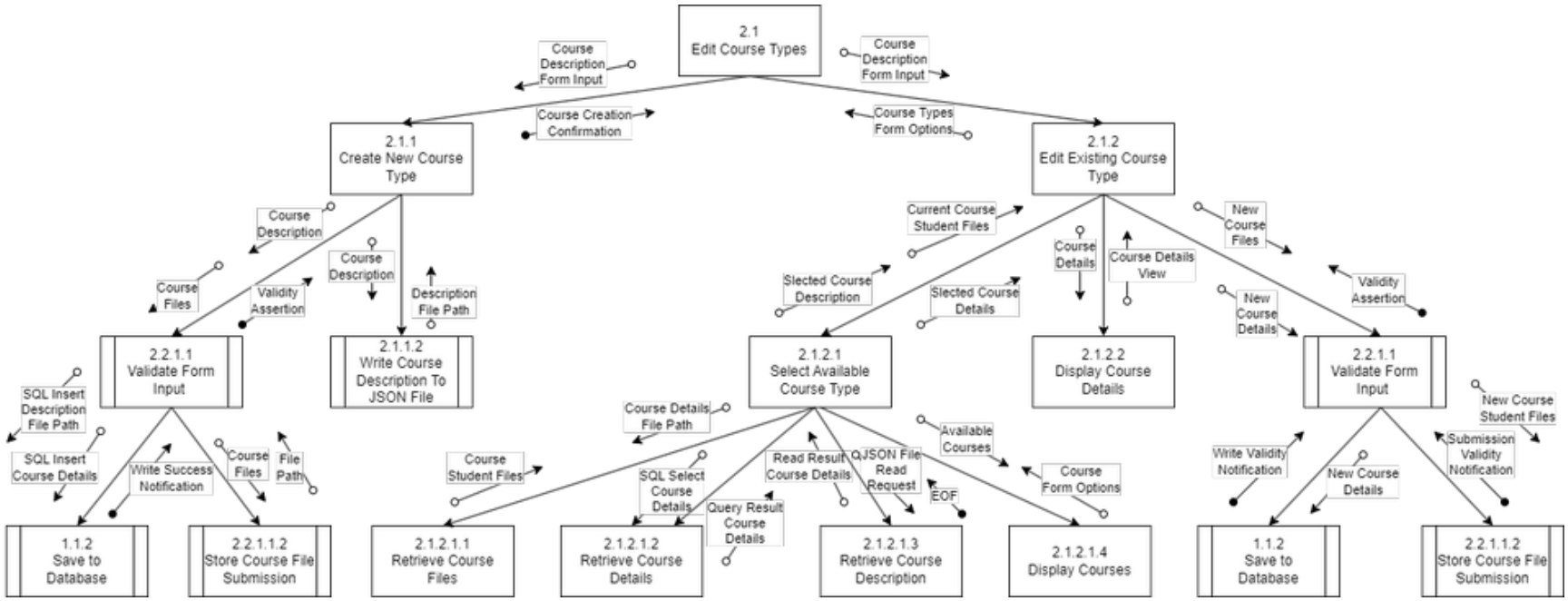



*Figure 3.2.2 – 1*

## Program Specifications for Process 2.1

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.1 | | |
| **Purpose:** | Create the route that returns the view for editing courses. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| User navigates to edit course page. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Web URL path | String | 2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Web view | Blade template | 2.1.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| //Add Route | | | |
| Route::get('/', function() | | | |
| return view('newCourseView', ['$coursePrice' => 'coursePrice']) | | | |
| | | | |
| | | | |
| | | | |

*Figure 7.2.2.2 – 2*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.1.1 Create New Course Type | | |
| **Purpose:** | Display form for creating a new course type. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | HTML | | |
| **Events:** | | | |
| User types new course details. | | | |
| Form input button keystroke. | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Course details | Varchar | User | |
| Course files | PDF, JPG, PNG, XSLX | User | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Form fields | Blade template | 2.2.1.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| <form> | | | |
| <label for "fieldName">Field Name: </label> | | | |
| <input type="text" name="fieldName"> | | | |
| <input type="file" name="courseFile"> | | | |
| <input type="button" name="submit"> | | | |
| <form> | | | |

*Figure 7.2.2.2 - 3*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.2.1.1.2 Process Course File Submission | | |
| **Purpose:** | Store course file materials | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| User file submission. | | | |
| Form input button keystroke. | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Course Documentation | PDF | 2.2.1.1 | |
| Course Presentations | XLSX | 2.2.1.1 | |
| Course Images | PNG/JPG | 2.2.1.1 | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Destination Path | String | 1.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| In CLI: | | | |
| php artisan make:controller UploadFileController | | | |
| PHP inside the new controller file: | | | |
| //Retrieve stipulated file | | | |
| $file = $request->file('image'); | | | |
| //Move file to destination | | | |
| $destination = file path | | | |
| $file->move($destination,$file->getClientOriginalName()) | | | |

*Figure 7.2.2.2 – 4*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.1.1.2 Write Course Description to JSON File | | |
| **Purpose:** | Store previously processed user input in system file. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Processed user input received from control module. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Course Description | String | 2.1.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Write result | System Flag | 2.1.1 | |
| File path | String | 2.1.1.3 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| //Install prerequisites | | | |
| use Illuminate\Support\Facades\Storage; | | | |
| PHP inside the UploadDetailController file: | | | |
| $data=array(input fields) | | | |
| Storage::disk(local storage folder)->put('json file name', json_encode($data)); | | | |
| $destination = file path | | | |

*Figure 7.2.2.2 - 5*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.1.2.1 Select Available Course Type | | |
| **Purpose:** | Display form for editing an existing course type. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, HTML | | |
| **Events:** | | | |
| Course data and files received from subordinate modules. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Available course description | String | 2.1.2.1.3 | |
| Current course details | String | 2.1.2.1.2 | |
| Current course documents | PPTX, PNG, PDF | 2.1.2.1.1 | |
| Current courses form option | String | 2.1.2.1.4 | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Selected course type | String | 2.1.2.2 | |
| | | | |
| | | | |
| | | | |
| | | | |

**Pseudocode**

```
<form>
  //Populate form options
  <select name="available courses">
    <option>Course Name variable</option>
  </select>
  <p> Current Course Details </p>
  <label for "fieldName">Field Name: </label>
  <input type="text" name="fieldName">
  <input type="button" name="submit">
<form>
```

*Figure 7.2.2.2 – 6*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.1.2.1.1 Retrieve Course Files | | |
| **Purpose:** | Pulling course file details for display to the user. | | |
| **Programmer:** | | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | Josh | | |
| Submit form input button keystroke. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| File path | String | User input form | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Course File Details | String | 2.1.2.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |

**Pseudocode**

```
In CLI:
php artisan make:controller EditDetailController
PHP inside the new controller file:
$files = Storage::disk(local storage folder)->allFiles($directory);
```

*Figure 7.2.2.2 – 7*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.1.2.1.1.2 Retrieve Course Details | | |
| **Purpose:** | Retrieves course information to display to the user. | | |
| **Programmer:** | | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, SQL | | |
| **Events:** | Josh | | |
| Submit form input button keystroke. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Course type name | String | User input | The course being reviewed |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Course details | Varchar | 2.1.2.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| PHP inside the EditDetailController file: | | | |
| $details = DB::select('select * from table name'); //Repeat for each necessary field | | | |
| | | | |
| | | | |
| | | | |
| | | | |

*Figure 7.2.2.2 – 8*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.1.2.1.3 Retrieve Course Description | | |
| **Purpose:** | Retrieves course descriptions from a json file. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Submit form input button keystroke. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Course Type Name | String | User Input | |
| Course Type File Paths | String | 2.1.2.1 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Course Description | String | 2.1.2.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| PHP inside the EditDetailController file: | | | |
| Install prerequisites: use Storage | | | |
| $json = Storage::disk(local storage folder)->get(json file) | | | |
| $json = json_decode($json, true) | | | |
| | | | |

*Figure 7.2.2.2 – 9*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc:** | 2.1.2.2 Display Course Details | | |
| **Purpose:** | Display previously collected data to user | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, HTML | | |
| **Events:** | | | |
| System sends formatted data | | | |
| Route is invoked | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Course Details | String | 2.1.2.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Current Course Detail View | Blade View | PRA Admin | |
| | | | |
| | | | |
| | | | |
| | | | |

**Pseudocode**

```
//Populate view
<p>Course price: {{ $coursePrice}}<p>
//Add Route
Route::get('/courseDetails', function()
   return view('editDetailsView', ['$coursePrice' => 'coursePrice'])
```

*Figure 7.2.2.2 – 10*

## Process 2.2 Manage Classes

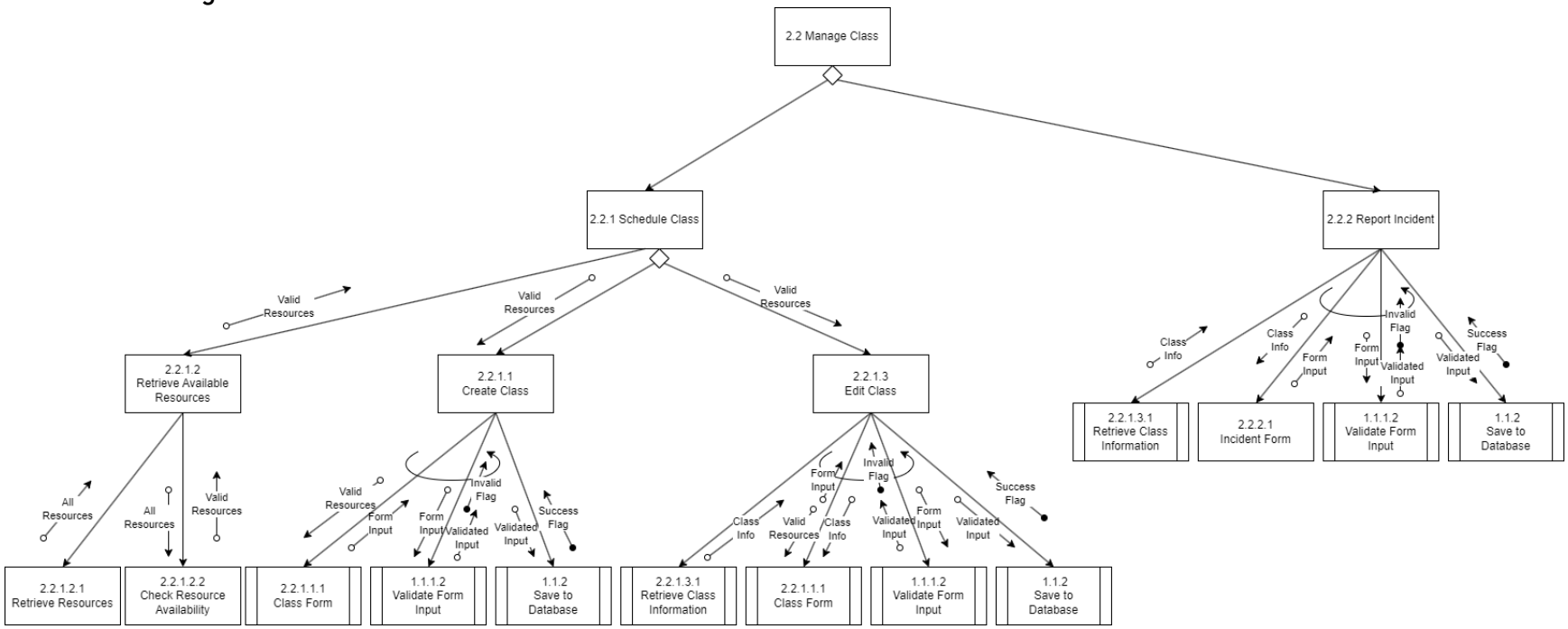



*Figure 7.2.2.2. - 11*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.2.1 Schedule Class | | |
| **Purpose:** | Class Creation and Management | | |
| **Programmer:** | Bryan Towpich | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Administrator Presses Class Management Button | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Valid Resources | String Array | 2.2.1.2 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Valid Resources | String Array | 2.2.1.1 | |
| Valid Resources | String Array | 2.2.1.3 | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
validResources = RetrieveAvailableResources();

<button label="Create Class" onPress={CreateClass(validResources)}>
<button label="Edit Class" onPress={EditClass(validResources)}>
```

*Figure 7.2.2.2. – 12*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.2.1.1 Create Class | | |
| **Purpose:** | Allow an Admin to create a class | | |
| **Programmer:** | Bryan Towpich | | |
| **Date due:** | | | |
| **Language/platform:** | HTML, PHP | | |
| **Events:** | | | |
| Using Create a class and save it to the Database | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Valid Resources | String Array | Program 2.2.1 | Selectable Resources |
| Form Input | String Array | Program 2.2.1.1.1 | |
| Invalid Flag | Boolean | Program 1.1.1.2 | Form Input is Invalid |
| Validated Input | String Array | Program 1.1.1.2 | |
| Success Flag | boolean | Program 1.1.2 | Completed Successfully |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Valid Resources | String Array | Program 2.2.1.1.1 | Selectable Resources |
| Form Input | String Array | Program 1.1.1.2 | |
| Validated Input | String Array | Program 1.1.2 | |
| **Pseudocode** | | | |

```
validResources passed by control module
validatedInput = false
while(!validatedInput){
  formInput = classCreationForm(validResources);
  validatedInput = ValidateFormInput(validatedInput);
}
SaveToDatabase(validatedInput);
```

*Figure 7.2.2.2. - 13*

| Program Specifications for "Open Road" | | | |
|---|---|---|---|
| **Module Name, number, et** | 2.2.1.1.1 Class Creation Form | | |
| **Purpose:** | Retrieve Class Information from Admin | | |
| **Programmer:** | | | |
| **Date due:** | | | |
| **Language/platform:** | HTML, PHP | | |
| **Events:** | | | |
| Load selectable resources into form, Button press to send form information | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Valid Resources | String Array | Program 2.2.1.1 | Selectable Resources |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Form Input | String Array | Program 2.2.1.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| for(i = 0; i<validResources.length; i++){<br>put validResources[i] into dropdown for staff<br>}<br>if module recieves data from previous entry Populate Input Fields with Data<br>form to fill out<br>return(formInput) | | | |

*Figure 7.2.2.2. – 14*

| Program Specifications for "Open Road" | | | |
|---|---|---|---|
| **Module Name, number, et** | 2.2.1.2 Rerieve Available Resources | | |
| **Purpose:** | Retrieve Resources for Class Creation Form | | |
| **Programmer:** | | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Retrieve list of Resources for the class form | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| All Resources | String Array | Program 2.2.1.2.1 | |
| Valid Resources | String Array | Program 2.2.1.2.2 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| All Resources | String Array | Program 2.2.1.2.2 | |
| Valid Resources | String Array | Program 2.2.1 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| allResources = RetrieveResources();<br>validResources = checkResourceAvailability(allResources);<br>return (validResources) | | | |

*Figure 7.2.2.2. - 15*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.2.1.2.1 Retrieve Resources | | |
| **Purpose:** | Retrieve all resources from database | | |
| **Programmer:** | | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Retrieve all Resources Return List of all resources | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| All Resources | String Array | Program 2.2.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Establish Connection to DB<br>execute query "SELECT * FROM stuff WHERE stuff_level = 1;"<br>$resources = array();<br>while ($row = $result->fetch_assoc()) {<br>$resources[] = $row;<br>}<br>return ($resources);<br>close DB connection | | | |

*Figure 7.2.2.2. – 16*

| Program Specifications for | "Open Road" | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.2.1.2.2 Check Resource Availability | | |
| **Purpose:** | Check Resource Availability | | |
| **Programmer:** | Bryan Towpich | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Recieves list of all resources and checks them against who is available for a specific date | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| All Resources | String Array | Program 2.2.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Valid Resources | String Array | Program 2.2.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Establish Connection to DB<br>for each instructor in array {<br>execute query "SELECT * FROM stuff WHERE stuff_id = {instructor};"<br>if instructor is available add to validResources array<br>}<br>return(validResources) | | | |

*Figure 7.2.2.2. – 17*

| Program Specifications for "Open Road" | | | |
|---|---|---|---|
| **Module Name, number, etc:** | 2.2.1.3 Edit Class | | |
| **Purpose:** | Allows Editing a class | | |
| **Programmer:** | Bryan | | |
| **Date due:** | | | |
| **Language/platform:** | HTML, PHP | | |
| **Events:** | | | |
| Retrieve class info, Edit class info, Save class info | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Class Info | String Array | Program 2.2.1.3.1 | Selectable Resources |
| Valid Resources | String Array | Program 2.2.1 | Selectable Resources |
| Form Input | String Array | Program 2.2.1.1.1 | |
| Invalid Flag | Boolean | Program 1.1.1.2 | Form Input is Invalid |
| Validated Input | String Array | Program 1.1.1.2 | |
| Success Flag | boolean | Program 1.1.2 | Completed Successfully |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Class Info | String Array | Program 2.2.1.3.1 | Selectable Resources |
| Valid Resources | String Array | Program 2.2.1.3.1 | Selectable Resources |
| Form Input | String Array | Program 1.1.1.2 | |
| Validated Input | String Array | Program 1.1.2 | |
| **Pseudocode** | | | |

```
validResources passed by control module
validatedInput = false
classInfo = RetrieveClassInformation
while(!validatedInput){
   formInput = classCreationForm(validResources, classInfo);
   validatedInput = ValidateFormInput(validatedInput);
}
SaveToDatabase(validatedInput);
```

*Figure 7.2.2.2. – 18*

| Program Specifications for "Open Road" | | | |
|---|---|---|---|
| **Module Name, number, etc:** | 2.2.1.3 Edit Class | | |
| **Purpose:** | Gathers Class Info | | |
| **Programmer:** | Bryan | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, SQL | | |
| **Events:** | | | |
| Retrieve class info, Returns to control Module | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Class Info | String Array | Program 2.2.1.3.1 | Selectable Resources |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
Establish DB Connection
Execute Query for class Info
return classInfo
```

*Figure 7.2.2.2. - 19*

| Program Specifications for "Open Road" | | | |
|---|---|---|---|
| **Module Name, number, etc:** | 2.2.2 Report Incident | | |
| **Purpose:** | Report an Incident that happens during a class | | |
| **Programmer:** | Bryan | | |
| **Date due:** | | | |
| **Language/platform:** | HTML, PHP | | |
| **Events:** | | | |
| Retrieves Class Info, Creates Incident form for completion, validates form input, saves to database | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Class Info | String Array | Program 2.2.1.3.1 | |
| Form Input | String Array | Program 2.2.2.1 | |
| Invalid Flag | Boolean | Program 1.1.1.2 | |
| Validated Input | String Array | Program 1.1.1.2 | |
| Success Flag | Boolean | Program 1.1.2 | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Class Info | String Array | Program 2.2.2.1 | |
| Form Input | String Array | Program 1.1.1.2 | |
| Validated Input | String Array | Program 1.1.2 | |
| **Pseudocode** | | | |
| classInfo = RetrieveClassInformation();<br>validatedInput = false;<br>while(!validatedInput){<br>formInput = IncidentForm(classInfo);<br>validatedInput = ValidateFormInput(formInput);<br>}<br>SaveToDatabase(validatedInput); | | | |

*Figure 7.2.2.2. – 20*

| Program Specifications for "Open Road" | | | |
|---|---|---|---|
| **Module Name, number, etc:** | 2.2.2.1 | | |
| **Purpose:** | Incident Form | | |
| **Programmer:** | Bryan | | |
| **Date due:** | | | |
| **Language/platform:** | HTML, PHP | | |
| **Events:** | | | |
| Generate form Return Inputs | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Class Info | String Array | Program 2.2.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Form Input | String Array | Program 2.2.2 | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| load class_id from class into class_id field<br><br>Generate Form<br><br><button label="Submit" onPress={return (formInfo)}> | | | |

*Figure 7.2.2.2. – 21*

## Process 2.3 Manage Registration

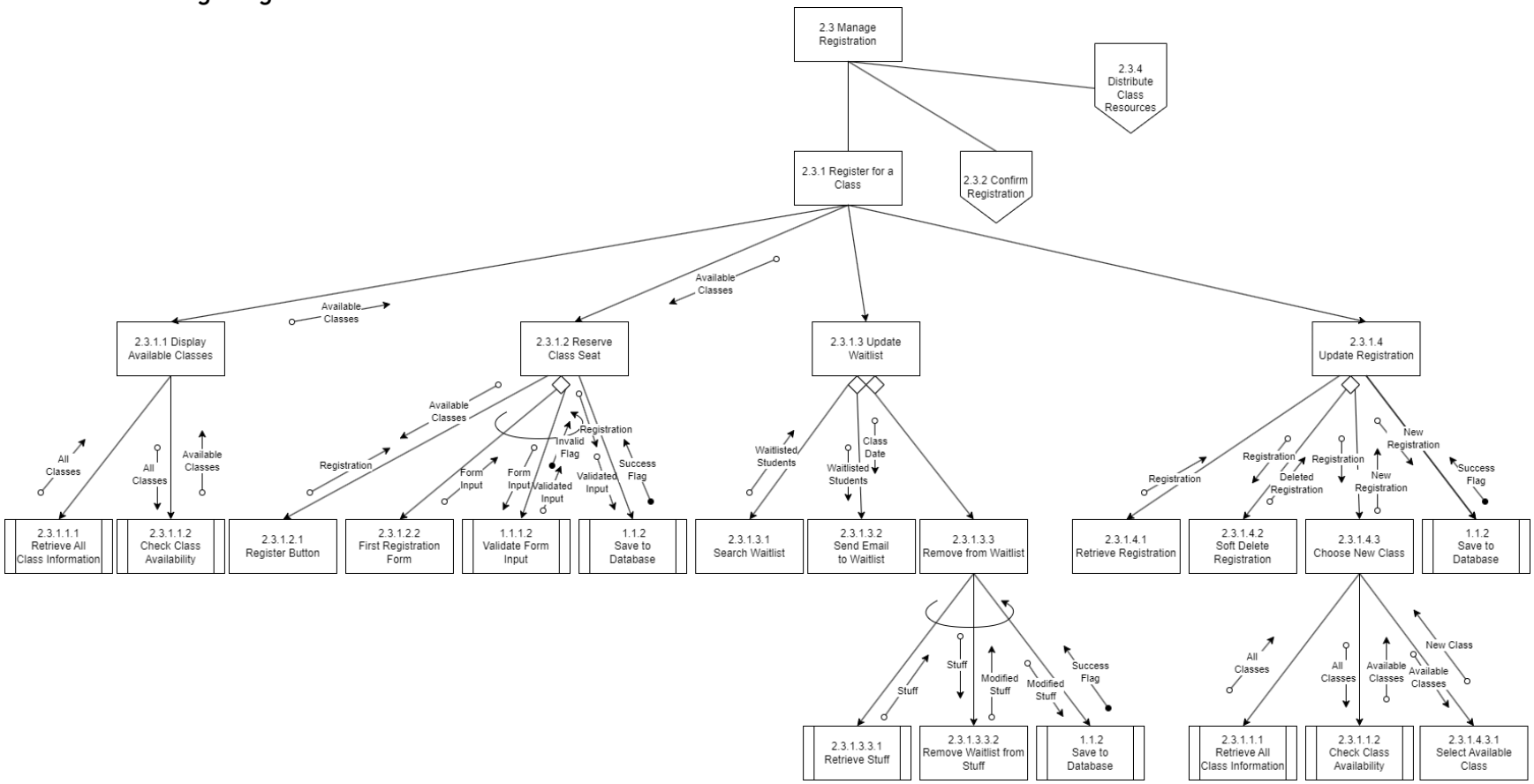



*Figure 7.2.2.2. – 22*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc** | Register for a Class 2.3.1.1 | | |
| **Purpose:** | Get a list of Available Classes | | |
| **Programmer:** | Bryan Towpich | | |
| **Date due:** | | | |
| **Language/platform:** | HTML,PHP, SQL, BootStrap | | |
| **Events:** | | | |
| Retrieve available classes and return them | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| All Classes | String Array | Program 2.3.1.1.1 | |
| Available Classes | String Array | Program 2.3.1.1.2 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| All Classes | String Array | Program 2.3.1.1.2 | |
| Available Classes | String Array | Program 2.3.1 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| allClasses = RetrieveAllClassInformation();<br>availableClasses = CheckClassAvailability(allClasses); | | | |

*Figure 7.2.2.2. – 23*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.3.1.1.1 Retrieve Classes | | |
| **Purpose:** | To Retrieve All Classes | | |
| **Programmer:** | Bryan Towpich | | |
| **Date due:** | | | |
| **Language/platform:** | PhP, HTML. CSS,SQL | | |
| **Events:** | | | |
| When the link is clicked all classed should appear in a form | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Establish Connection to DB<br>execute query "SELECT * FROM class WHERE {class is not passed}"<br>allClasses= array();<br>while (row = allClasses->fetch_assoc()) {<br>  allClasses[] = row;<br>} | | | |

*Figure 7.2.2.2. - 24*

| | | | |
|---|---|---|---|
| **Module Name, number, e** | 2.3.1.1.2 Check Class Availability | | |
| **Purpose:** | To run an Availability check on classes | | |
| **Programmer:** | Bryan Towpich | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, HTML, SQL | | |
| **Events:** | recieves all classes returns only available classes | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
Establish Connection to DB
for each class in class table
execute query "SELECT * FROM registration WHERE class id = {this.class id}" check for 5 instances
if less then 5 instances add to availableClasses array

close DB connection
return (availableClasses);
```

*Figure 7.2.2.2. - 25*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, et** | 2.3.1.2 Reserve Class Seat | | |
| **Purpose:** | To Display all Avaailable classes | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-05 | | |
| **Language/platform:** | PHP,HTML,CSS,SQL | | |
| **Events:** | | | |
| recieve Available Classes, Create Button, Create first time form, Save registration to database | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Available Classes | String Array | Program 2.3.1 | |
| Registration | String Array | Program 2.3.1.2.1 | |
| Form Input | String Array | Program 2.3.1.2.2 | |
| Validated Input | String Array | Program 1.1.1.2 | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Available Classes | String Array | Program 2.3.1.2.1 | |
| Form Input | String Array | Program 1.1.1.2 | |
| Registration | String Array | Program 1.1.2 | |
| Validated Input | String Array | Program 1.1.2 | |
| | | | |
| **Pseudocode** | | | |

```
registration = RegisterButton();
if user has never registered before {
  while(!validatedInput){
    formInput = FirstRegistrationForm();
    validatetedInput = ValidateFormInput(formInput);
  }
  SaveToDatabase(registration)
}
SaveToDatabase(validatedInput)
```

*Figure 7.2.2.2. – 26*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, et** | 2.3.1.2.1 Register Button | | |
| **Purpose:** | Chose class from buttons generated | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | SQL, PHP | | |
| **Events:** | | | |
| Creates buttons for classes available | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Available Classes | String Array | Program 2.3.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Registration | String Array | Program 2.3.1.2.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
for each class in availableClasses{
  <Button label={class.date} onPress={return(class)}>
}
```

*Figure 7.2.2.2. - 27*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, e** | 2.3.1.2.2 First Registration Form | | |
| **Purpose:** | Form Used to fill Stuff table on first registration | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | SQL, PHP | | |
| **Events:** | | | |
| Creates form with submit button | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Form Input | String Array | 2.3.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Create form required to fill stuff table with all relevant information<br><br>submit button returns formInput | | | |

*Figure 7.2.2.2. - 28*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, et** | 2.3.1.3 Update Waitlist | | |
| **Purpose:** | To Allow the waitlist to be updated and Emails to be sent to those waiting. | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, SQL, HTML | | |
| **Events:** | | | |
| Search Waitlist, Send Emails, Remove student from waitlist | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Waitlisted Students | String Array | Program 2.3.1.3.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Waitlisted Students | String Array | Program 2.3.1.3.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
If class is not full before class begins{
   waitlistedStudents = SearchWaitlist();
   SendEmailToWaitlist();
}
if student on waitlist registers for class{
   RemoveFromWaitlist();
}
```

*Figure 7.2.2.2. – 29*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, et** | 2.3.1.3.1 Search Waitlist | | |
| **Purpose:** | Shows all of a user's Student Registrations | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | SQL, PHP | | |
| **Events:** | | | |
| Seach Stuff For all waitlisted students, Return All Waitlisted Students | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Waitlisted Students | String Array | Program 2.3.1.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
Connect to DB
Execute SQL query to search all stuff on waitlist insert data into waitlistedStudents string array
return (waitlistedStudents)
```

*Figure 7.2.2.2. - 30*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, et** | 2.3.1.3.2 Send Email to Waitlist | | |
| **Purpose:** | To send an Email to the waitlist students | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | SQL, PHP | | |
| **Events:** | | | |
| Send Email to each Student on waitlist showing new class opportunity | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Waitlisted Students | String Array | Program 2.3.1.3 | |
| Class Date | String | Program 2.3.1.3 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
for each Waitlisted Student{
  Send Email with Class Date
}
```

*Figure 7.2.2.2. – 31*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, et** | 2.3.1.3.3 Remove from Waitlist | | |
| **Purpose:** | removed from the waitlist after class is taken | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | SQl, PHP | | |
| **Events:** | | | |
| Class is complete Waitlisted students who took part are removed | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Stuff | String Array | Program 2.3.1.3.3.1 | |
| Modified Stuff | String Array | Program 2.3.1.3.3.2 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Stuff | String Array | Program 2.3.1.3.3.1 | |
| Modified Stuff | String Array | Program 1.1.2 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
For each student who completed class and is on Waitlist{
  stuff = RetrieveStuff();
  modifiedStuff = RemoveWaitlistFromStuff(stuff);
  SaveToDatabase(modifiedStuff);
}
```

*Figure 7.2.2.2. - 32*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, et** | 2.3.1.3.3.1 Retrieve Registration | | |
| **Purpose:** | Stuff is retrieved | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | SQL, PHP | | |
| **Events:** | | | |
| Retrieve stuff from database | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Stuff | String Array | Program 2.3.1.3.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Connect to DB<br>execute query to retrieve stuff put in stuff string array<br>return (stuff) | | | |

*Figure 7.2.2.2. – 33*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, et** | 2.3.1.3.3.2 Generate SQL Query | | |
| **Purpose:** | Removes Waitlist flag from Stuff | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | SQL, PHP | | |
| **Events:** | | | |
| Removes waitlist from from | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Stuff | String Array | Program 2.3.1.3.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Modified Stuff | String Array | Program 2.3.1.3.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Remove Waitlist flag from Stuff array string<br>return (modifiedStuff); | | | |

*Figure 7.2.2.2. - 34*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, e** | 2.3.1.4 Update Registration | | |
| **Purpose:** | Update Registration Either cancel or change date | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | PHP,SQL,HTML | | |
| **Events:** | | | |
| Retrieve Registration, Chose to cancel or change registration date, Save registration to database | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Registration | String Array | Program 2.3.1.4.1 | |
| Deleted Registration | String Array | Program 2.3.1.4.2 | becomes new registration |
| New Registration | String Array | Program 2.3.1.4.3 | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Registration | String Array | Program 2.3.1.4.2 | |
| Registration | String Array | Program 2.3.1.4.3 | |
| New Registration | String Array | Program 1.1.2 | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| registration = RetrieveRegistration();<br><button label="Cancel Registration" onPress={newRegistration = SoftDeleteRegistration(registration)}><br><button label="Change Registration" onPress={newRegistration = ChooseNewClass(registration)}><br>SaveToDatabase(registration) | | | |

*Figure 7.2.2.2. – 35*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, e** | 2.3.1.4.1 Retrieve Registration | | |
| **Purpose:** | To retrieve Registration from the Database | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | PHP, SQL, | | |
| **Events:** | | | |
| Creates a button for each registration assigned to the user | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Registration | String Array | Program 2.3.1.4 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Create DB Connection<br>Execute Query to find all valid registrations for User as validRegistrations<br>for each validRegistration as registration{<br><button label={registration.date} onPress{return(registration)}><br>} | | | |

*Figure 7.2.2.2. 36*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, et** | 2.3.1.4.2 Soft Delete Registration | | |
| **Purpose:** | marks a registration as deleted returns | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | PHP, SQL, HTML, CSS | | |
| **Events:** | | | |
| Marks a registration as deleted and returns string | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Registration | String Array | Program 2.3.1.4 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Deleted Registration | String Array | Program 2.3.1.4 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Mark Deleted field in registration as true<br>return (deletedRegistration) | | | |

*Figure 7.2.2.2. – 37*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, et** | 2.3.1.4.3 Choose New Class | | |
| **Purpose:** | Take the Registration and choose a new class and return a Class ID | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | PHP, SQL, HTML, CSS | | |
| **Events:** | | | |
| Recieve Registration, Retrieve All Classes, Check Availability, Present New Choices to students, | | | |
| Return New Registration, | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Registration | String Array | Program 2.3.1.4 | |
| All Classes | String Array | Program 2.3.1.4 | |
| Available Classes | String Array | Program 2.3.1.4 | |
| New Registration | String Array | Program 2.3.1.4 | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| All Classes | String Array | Program 2.3.1.4 | |
| Available Classes | String Array | Program 2.3.1.4 | |
| New Class | String Array | Program 2.3.1.4 | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| allClasses = RetrieveAllClassInformation();<br>availableClasses = CheckClassAvailability(allClasses);<br>newRegistration = SelectAvailableClass(availableClasses);<br><br>change Class ID of registration to Class ID<br>return(newRegistration); | | | |

*Figure 7.2.2.2. - 38*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, e** | 2.3.1.4.3 Choose New Class | | |
| **Purpose:** | Take the Registration and choose a new class and return a Class ID | | |
| **Programmer:** | Bryan Topwich | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | PHP, SQL, HTML, CSS | | |
| **Events:** | | | |
| Create Button for each registration, return selected registration | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Available Classes | String Array | Program 2.3.1.4.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| **New** | | | |
| New Class | String Array | Program 2.3.1.4.3 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
for each class in availableClasses{
  <Button label={class.date} onPress={return(class)}
}
```

*Figure 7.2.2.2. - 39*

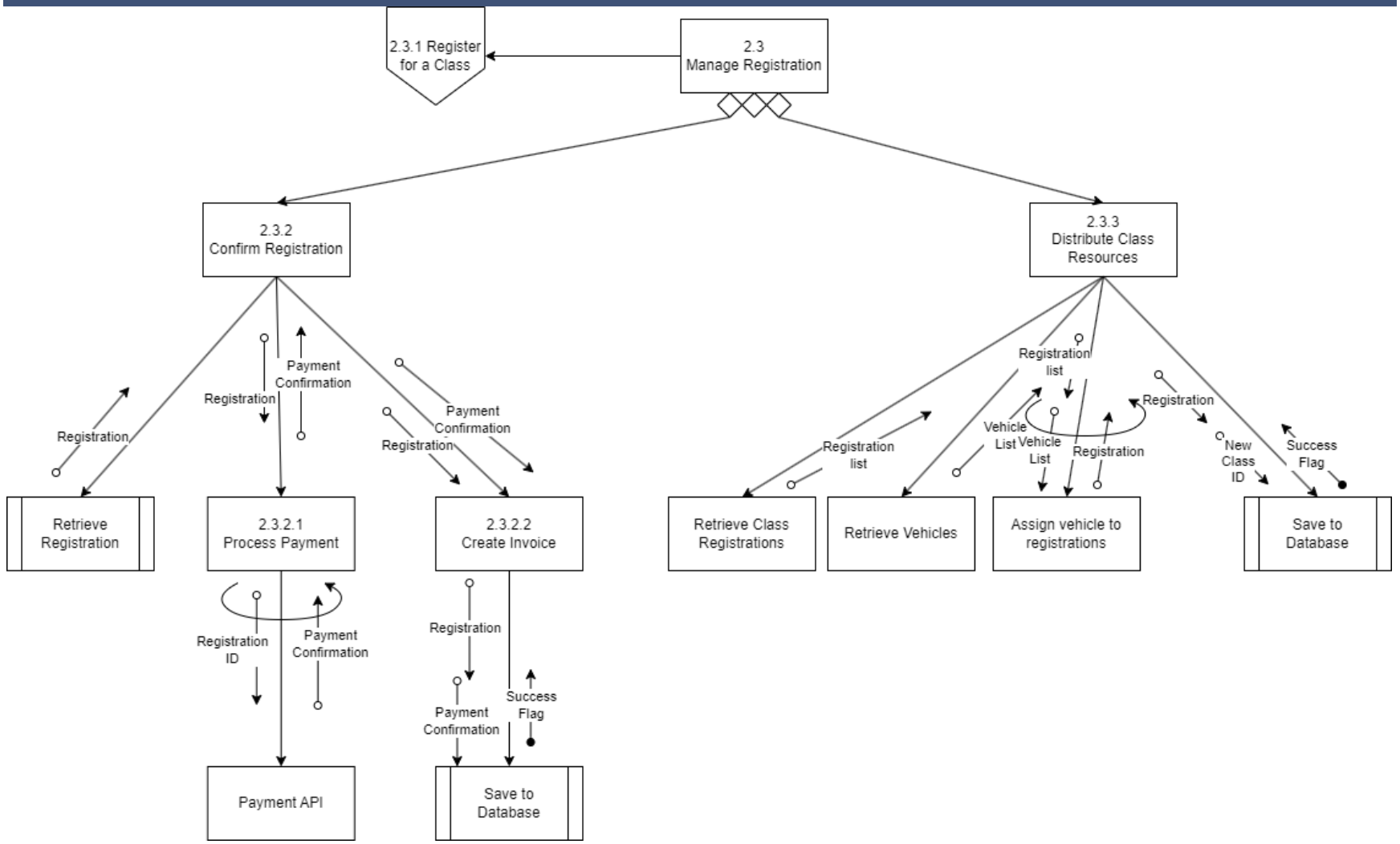



*Figure 7.2.2.2. - 40*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, et** | 2.3.2 Confirm Registration | | |
| **Purpose:** | Confirm the Registration thought payment and processes | | |
| **Programmer:** | Nathaniel Meyer | | |
| **Date due:** | | | |
| **Language/platform:** | PHP,SQL | | |
| **Events:** | | | |
| Retrieve Registration, Process Payment, Create Invoice | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Registration | String Array | Retrieve Registration | |
| Payment Confirmation | String Array | Program 2.3.2.1 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Registration | String Array | Program 2.3.2.1 | |
| Registration | String Array | Program 2.3.2.2 | |
| Payment Confirmation | String Array | Program 2.3.2.2 | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
regtosend = getregistration()
confirmationid = sendregistration(regtosend, payment)
createinvoice(regtosend, confirmationid)
```

*Figure 7.2.2.2. – 41*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, et** | 2.3.2.1 Process Payment | | |
| **Purpose:** | Take the Registration ID and allow Payment API and wait for Payment Confirmati | | |
| **Programmer:** | Nathaniel Meyer | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | PHP,SQL | | |
| **Events:** | | | |
| Send Registration to payment API, return confirmation of payment | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Registration | String Array | Program 2.3.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Payment Confirmation | Boolean | Program 2.3.2.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
sendPaymentRequest(price)
ifPaymentSuccess (sendPaymentRequest) {
   paymentConfirmation = true;
} else {
   paymentConfirmation = false;
}
```

*Figure 7.2.2.2. - 42*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, e** | 2.3.2.2 Create Invoice | | |
| **Purpose:** | An invoice will be created with the registration info and payment confirmation | | |
| **Programmer:** | Nathaniel Meyer | | |
| **Date due:** | | | |
| **Language/platform:** | PHP,SQL | | |
| **Events:** | | | |
| Receive Payment Confirmation, Receive registration Info, Save Registration to DB, Save Invoice to DB | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Payment Confirmation | Boolean | Program 2.3.2 | |
| Registration | String Array | Program 2.3.2 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Registration | String Array | Save to Database | |
| Payment Confirmation | Boolean | Save to Database | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| saveToDB(registration)<br>saveToDB(registration, invoice) | | | |

*Figure 7.2.2.2. - 43*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, e** | 2.3.3 Distribute Class Resources | | |
| **Purpose:** | Distribute class resources between registrations | | |
| **Programmer:** | Nathaniel Meyer | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, SQL | | |
| **Events:** | | | |
| Get Registrations, Get Vehicles, Assign Vehicles to Registrations, Update Registrations | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Registration List | String Array | Program 2.3.3.1 | |
| Vehicle List | String Array | Program 2.3.3.2 | |
| Registration | String Array | Program 2.3.3.3 | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Registration List | String Array | Program 2.3.3.3 | |
| Vehicle List | String Array | Program 2.3.3.3 | |
| Registration | String Array | Program 2.3.3.4 | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| regToSend = getRegList()<br>vehToSend = getVehList()<br><br>foreach registration in class {<br>newReg[x] = assignNewReg(regToSend[x], vehToSend[x])<br>saveToDB(newReg[x])<br>} | | | |

*Figure 7.2.2.2. – 44*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, e** | 2.3.3.1 Retrieve Class Registrations | | |
| **Purpose:** | Call the registration database and retrieve Class Registrations | | |
| **Programmer:** | Nathaniel Meyer | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | PHP, SQL | | |
| **Events:** | | | |
| Get class registrations from db | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Registration List | String Array | 2.3.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Return getRegList() | | | |

*Figure 7.2.2.2. – 45*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, e** | 2.3.3.2 Retrieve Vehicles | | |
| **Purpose:** | call to database to Retrieve Vehicles information | | |
| **Programmer:** | Nathaniel Meyer | | |
| **Date due:** | 2023-02-15 | | |
| **Language/platform:** | PHP, SQL | | |
| **Events:** | | | |
| Retrieve Vehicle list | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Vehicle List | String Array | 2.3.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Return getVehList() | | | |

*Figure 7.2.2.2. – 46*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, e** | 2.3.3.3 Assign vehicle to registrations | | |
| **Purpose:** | Find the vehicle and assign it to registrations is the Distribute Class Resource | | |
| **Programmer:** | Nathaniel Meyer | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, SQL | | |
| **Events:** | | | |
| Receive registration and vehicle list, assign vehicle to registration, return new registration | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Registration List | String Array | 2.3.3 | |
| Vehicle List | String Array | 2.3.3 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Registration | String Array | 2.3.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| newReg = regToSend + vehToSend<br><br>Return newReg | | | |

*Figure 7.2.2.2. – 47*

## Process 2.4 Generate Report

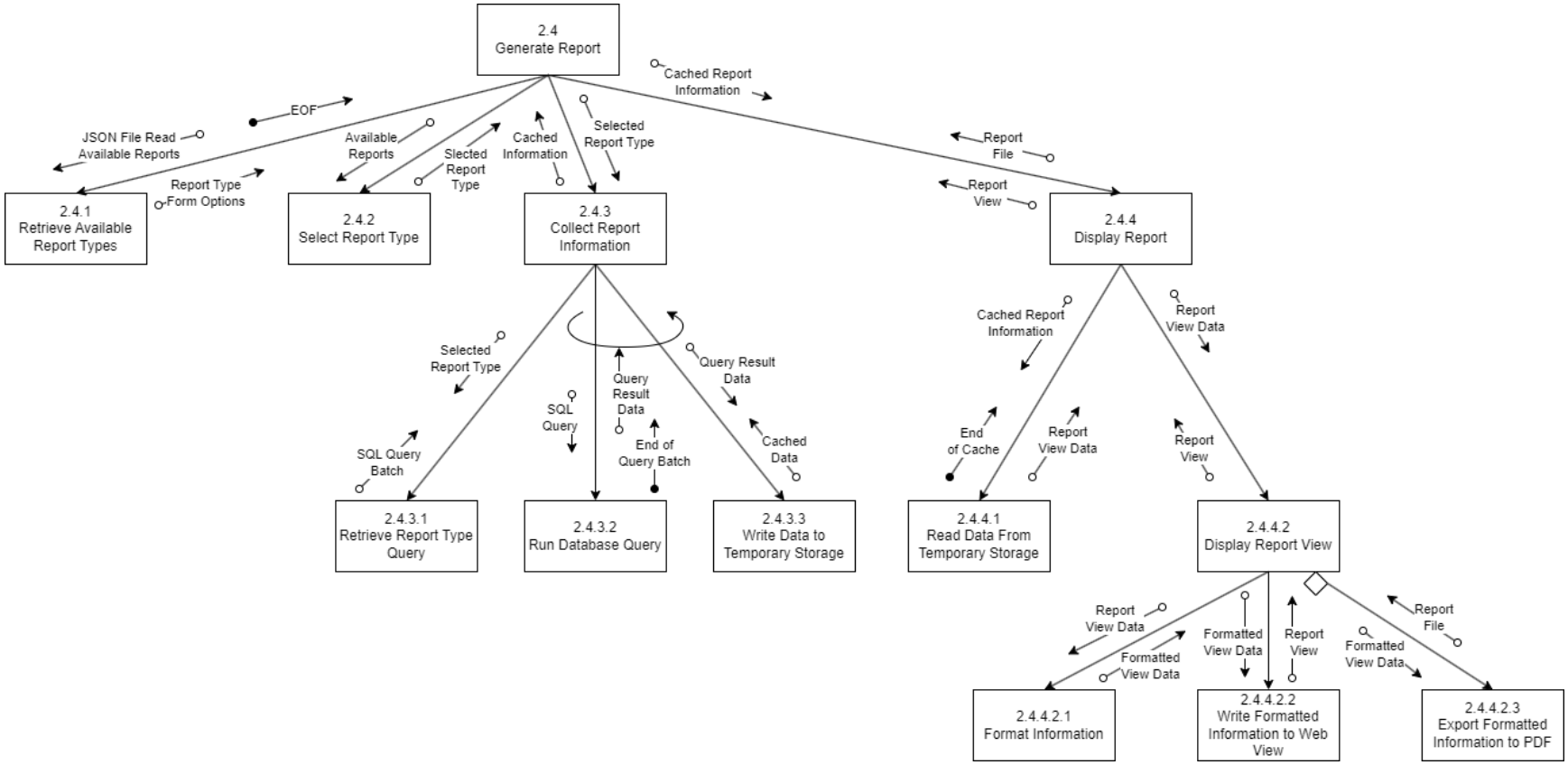



*Figure 7.2.2.2 - 48*

# Program Specifications for Process 2.4

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc:** | 2.4 Generate Report | | |
| **Purpose:** | Create the route that returns the view for generating reports | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| User navigates to report generator. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Web URL path | String | System navigation | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Web view | Blade template | 2.4.1, 2.4.2, and 2.4.4 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| //Add Route | | | |
| Route::get('/Reporting', function() | | | |
| return view('generateReportView') | | | |
| | | | |
| | | | |
| | | | |

*Figure 7.2.2.2 – 49*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc:** | 2.4.1 Retrieve Available Report Types | | |
| **Purpose:** | Read and display a report type form options. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Report web view renders. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| JSON File location | String | System | System translates request to file path |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Report type form options | String | 2.4.2 | |
| End of file control | System flag | 2.4 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Install prerequisites: use Storage | | | |
| $json = Storage::disk(local storage folder)->get(json file) | | | |
| $json = json_decode($json, true) | | | |
| //Populate form options | | | |
| <select name="available courses"> | | | |
| <option>Course Name variable</option> | | | |
| </select> | | | |

*Figure 7.2.2.2 – 50*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, et** | 2.4.2 Select Report Type | | |
| **Purpose:** | Accept user input for report to be generated. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, HTML | | |
| **Events:** | | | |
| User selects a form option. | | | |
| Submit form input button keystroke. | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Form options | String | 2.4.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Form option selected | String | 2.4.3.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| In CLI: | | | |
| php artisan make:controller ReportController | | | |
| In PHP inside the new controller file: | | | |
| $post = new Post; | | | |
| $post->*FieldName* = $request->*FieldName* //Repeat for each field | | | |

*Figure 7.2.2.2 – 51*

| Program Specifications for Project Open Road | | | |
|---|---|---|---|
| **Module Name, number, et** | 2.4.3 Collect Report Information | | |
| **Purpose:** | Recognize query completion and pass cache location | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| End of query batch flag received. | | | |
| Written cache received from subordinate. | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| End of query batch | Control flag | 2.4.3.2 | |
| Cached information | Query results | 2.4.3.3 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Cache location | Memory location | 2.4.4 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| if (end of batch) | | | |
| let last query happen | | | |
| $cachePath = location of cache | | | |

*Figure 7.2.2.2 - 52*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, et** | 2.4.3.1 Retrieve Report Type Query | | |
| **Purpose:** | Retrieves the appropriate batch of SQL queries for retrieving the report data. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, SQL | | |
| **Events:** | | | |
| Report type option selected. | | | |
| Form submit button pressed. | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Selected report type | String | 2.4.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| SQL Query batch | SQL | 2.4.3.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| In PHP inside the ReportController file: | | | |
| while (!end of batch) | | | |
| switch('reportType') | | | |
| case (financial): | | | |
| return(financial data query batch) | | | |
| case (student): | | | |
| return(student data query batch) //etc. | | | |

*Figure 7.2.2.2 – 53*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, et** | 2.4.3.2 Run Database Query | | |
| **Purpose:** | Run through individual queries provided in batch. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, SQL | | |
| **Events:** | | | |
| Query batch received from system. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| SQL Query | SQL | 2.4.3.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Query result data | String | 2.4.3.3 | |
| End of query batch flag | System flag | 2.4.4 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| In PHP inside the ReportController file: | | | |
| $detail = DB::select('select detail from table name'); | | | |

*Figure 7.2.2.2 – 54*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, et** | 2.4.3.3 Write Data to Temporary Storage | | |
| **Purpose:** | Stores query data in temporary storage. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| SQL Query result received from 2.4.3.2 | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Query result data | String | 2.4.3.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Cached data | Memory location | 2.4.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| //As query results are received from 2.4.3.2 | | | |
| memcached => [ key => value ] | | | |
| | | | |
| | | | |
| | | | |

*Figure 7.2.2.2 – 55*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, et** | 2.4.4 Display Report | | |
| **Purpose:** | Create route that enables the displaying of report vie | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Report data cache received from system. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Cached information | Memory location | 2.4.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| View route | Larvel route code | 2.4.4.2 | |
| Cache location | String | 2.4.4.1 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| //Create Route | | | |
| Route::get('/reportView', function() | | | |
| return view('formattedReport', ['$key' => 'value']) | | | |
| | | | |

*Figure 7.2.2.2 – 56*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, e** | 2.4.4.1 Read Data From Temporary Storage | | |
| **Purpose:** | Read from the cache until end of cache. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Cache location received from controlling module. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Cache location | String | 2.4.4 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Individual data variables | PHP variable | 2.4.4.2.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| While (!end of cache) | | | |
| Read value from cache | | | |
| Store value to local variable | | | |
| | | | |

*Figure 7.2.2.2 – 57*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, e** | 2.4.4.2 Display Report View | | |
| **Purpose:** | Call route to diplay formatted report view. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, HTML | | |
| **Events:** | | | |
| A user may choose to export view to PDF. | | | |
| Formatted information reeived from subordinate module. | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| View route | Larvel route code | 2.4.4 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Report view | Blade View | 2.4.4 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| //Get route details | | | |
| $request = Request::create(route info) | | | |
| //Handle request | | | |
| $response = app()->handle($request) | | | |
| $responseBody = $response->getContent(); | | | |

*Figure 7.2.2.2 – 58*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, e** | 2.4.4.2.1 Format Information | | |
| **Purpose:** | Validate and re-format values read from cache. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Report data variables initiated from controlling module. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Individual data variables | PHP variable | 2.4.4.1 | A wide variety of variable |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Correctly formatted data v | System variable | 2.4.4.2.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| for each (populated data variable) | | | |
| format as per report requirements | | | |
| restore data in variable | | | |
| send flag stating data is formatted | | | |

*Figure 7.2.2.2 – 59*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, e** | 2.4.4.2.2 Write Formatted Information to Web View | | |
| **Purpose:** | Populate view with formatted information. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Formatted information passed as variables from controlling module. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Formatted data | System variable | 2.4.4.2.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Route properties | System variable | 2.4.4.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| for each (formatted variable) | | | |
| route variable = formatted variable | | | |
| invoke appropriate route for selected report type | | | |

*Figure 7.2.2.2 – 60*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc:** | 2.4.4.2.3 Export Formatted Information to PDF | | |
| **Purpose:** | Optionally download a pdf containing formatted repo | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, HTML | | |
| **Events:** | | | |
| User selects optional "export to PDF button". | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Formatted view data | String | 2.4.4.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Report file | PDF | 2.4.4.2 | |
| | | | |
| | | | |
| | | | |
| | | | |

**Pseudocode**

```
//Install pre-requisites: composer require barryvdh/laravel-dompdf
//In app.php config file
providers' => [barryvdh\DomPDF\ServiceProvider::class
In CLI:
php artisan make:controller pdfReportController
In PHP inside the new controller file:
function downloadPDF(Request $request)
  if ($request->has('download')
    return $pdf->download($pdfname)
attach function to a download button in 2.4.4.2 view
```

*Figure 7.2.2.2 – 61*

## Process 2.5 Manage Class Resources

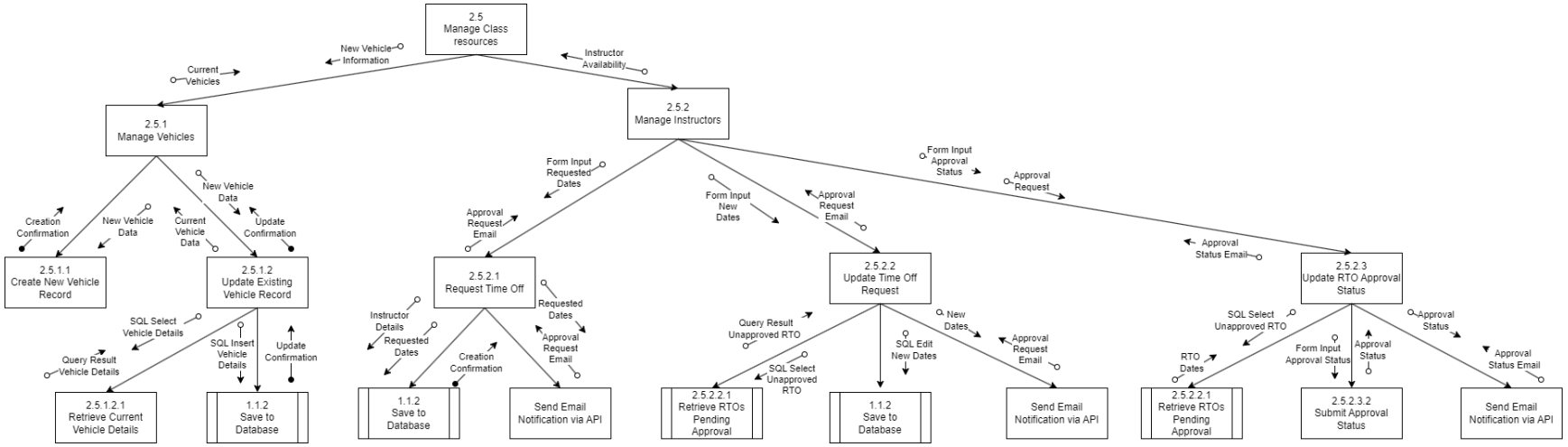



*Figure 7.2.2.2 - 62*

# Program Specifications for Process 2.5

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, e** | 2.5.1.2.1 Retrieve Current Vehicle Details | | |
| **Purpose:** | Retrieve existing vehicle records from vehicle table. | | |
| **Programmer:** | | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, SQL | | |
| **Events:** | | | |
| Vehicle selection received from controlling module. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Vehicle name | String | 2.5.1.2 | |
| SQL select vehicle details | SQL Query | 2.5.1.2 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Vehicle details | Query results | 2.5.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| Select * from vehicle table where vehicle_name == $vehicleName | | | |
| Store query in variable passed to route and displayed in view | | | |
| | | | |
| | | | |
| | | | |

*Figure 7.2.2.2 – 63*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, e** | 2.5.2 Manage Instructors | | |
| **Purpose:** | Create input form view for managing Instructor sche | | |
| **Programmer:** | | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, HTML | | |
| **Events:** | | | |
| Input button for instructor management route selected. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Form option selection | Navigation route | User | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Request time off view | Web view | 2.5.2.1 | |
| Update RTO view | Web view | 2.5.2.2 | |
| Update RTO approval | Web view | 2.5.2.3 | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| <form><br>  <input type="button" name="newRTO" onpress="new RTO route"><br>  <input type="button" name="editRTO" onpress="edit RTO route" ><br>  <input type="button" name="approveRTO" onpress="unapproved RTO route" ><br></form><br>//Create 3 relevant view routes<br>Route::get('/newRTO', function()<br>  return view(newRTO view)<br>Route::get('/editRTO', function()<br>  return view(editRTO view)<br>Route::get('/approveRTO', function()<br>  return view(approveRTO view) | | | |

*Figure 7.2.2.2 – 64*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, e** | 2.5.2.2 Update Time Off Request | | |
| **Purpose:** | Retrieve old details, replace with new user input. | | |
| **Programmer:** | | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, SQL, HTML | | |
| **Events:** | | | |
| Edit time off request view called by route. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Unapproved RTO Details | String | 2.5.2.2.1 | |
| Update RTO view | Web view | 2.5.2 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Selected RTO ID | String | 2.5.2.2.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| //Display details | | | |
| Select * from instructor availability table where approved == null && userID == sessionID | | | |
| <p>Current RTO details<p> | | | |
| //In the editRTO view created in 2.5.2 | | | |
| <form> | | | |
| <input type="text" name="newRTODetails" > | | | |
| <input type="button" name="submit" > | | | |
| <select> | | | |
| <option>RTO ID</option> | | | |
| </select> | | | |
| </form> | | | |
| **Other** | | | |
| RTO IDs generated in subordinate module. | | | |

*Figure 7.2.2.2 – 65*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, e** | 2.5.2.1 Request Time Off | | |
| **Purpose:** | Pull staff info from table and provide a form view for | | |
| **Programmer:** | | | |
| **Date due:** | | | |
| **Language/platform:** | HTML, SQL | | |
| **Events:** | | | |
| Request time off view called by route. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| View path | String | 2.5.2 | |
| RTO Date/time | Varchar | User input | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| RTO Date/time | Varchar | 1.1.2, 2.5.2.1.1 | |
| Instructor details | String | 1.1.2 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| //Pull staff details from user table using session ID | | | |
| $staffDetails = Select * from user table where userID == session userID | | | |
| //In the newRTO view created in 2.5.2 | | | |
| <form> | | | |
| <input type="text" name="newRTODetails" > | | | |
| <input type="button" name="submit" > | | | |
| </form> | | | |

*Figure 7.2.2.2 – 66*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, et** | 2.5.2.2.3 Update RTO Approval Status | | |
| **Purpose:** | Retrieve RTO details, replace with new user input. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP, SQL, HTML | | |
| **Events:** | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Unapproved RTO Details | String | 2.5.2.2.1 | |
| Update RTO view | Web view | 2.5.2 | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Approval status | String | 2.5.2.3.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
//Display details
Select * from instructor availability table where approved == null
<p>Current RTO details<p>
//In the editRTO view created in 2.5.2
<form>
  <input type="text" name="approvalStatus" >
  <input type="button" name="submit" >
  <select>
    <option>RTO ID</option>
  </select>
</form>
```

*Figure 7.2.2.2 – 67*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, et** | 2.5.2.2.1 Retrieve RTOs Pending Approval | | |
| **Purpose:** | Fetches all RTOs with null approval status. | | |
| **Programmer:** | | | |
| **Date due:** | | | |
| **Language/platform:** | | | |
| **Events:** | | | |
| Record selected and passed from controller. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Selected RTO Record ID | String | 2.5.2.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| RTO Details | String | 2.5.2.2 | |
| RTO IDs | Varchar | 2.5.2.2 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
$RTO id = Select * from instructor availability table where approved == null && userID == sessionID
```

*Figure 7.2.2.2 – 68*

| Program Specifications for Open Road | | | |
|---|---|---|---|
| **Module Name, number, etc** | 2.5.2.3.1 Submit Approval Status | | |
| **Purpose:** | Retrieve and store form input. | | |
| **Programmer:** | Josh | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | | | |
| Form submit button pressed. | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Approval status | String | 2.5.2.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Approval status flag | System control flag | 2.5.2.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| $status = Input::get('name'); | | | |
| if ($status == approved) | | | |
| $approved = true | | | |
| else $approved = false | | | |

*Figure 7.2.2.2 - 69*

### 7.2.2.3. P 3 – Web Management

Process 3.1.1. Update Ad Content

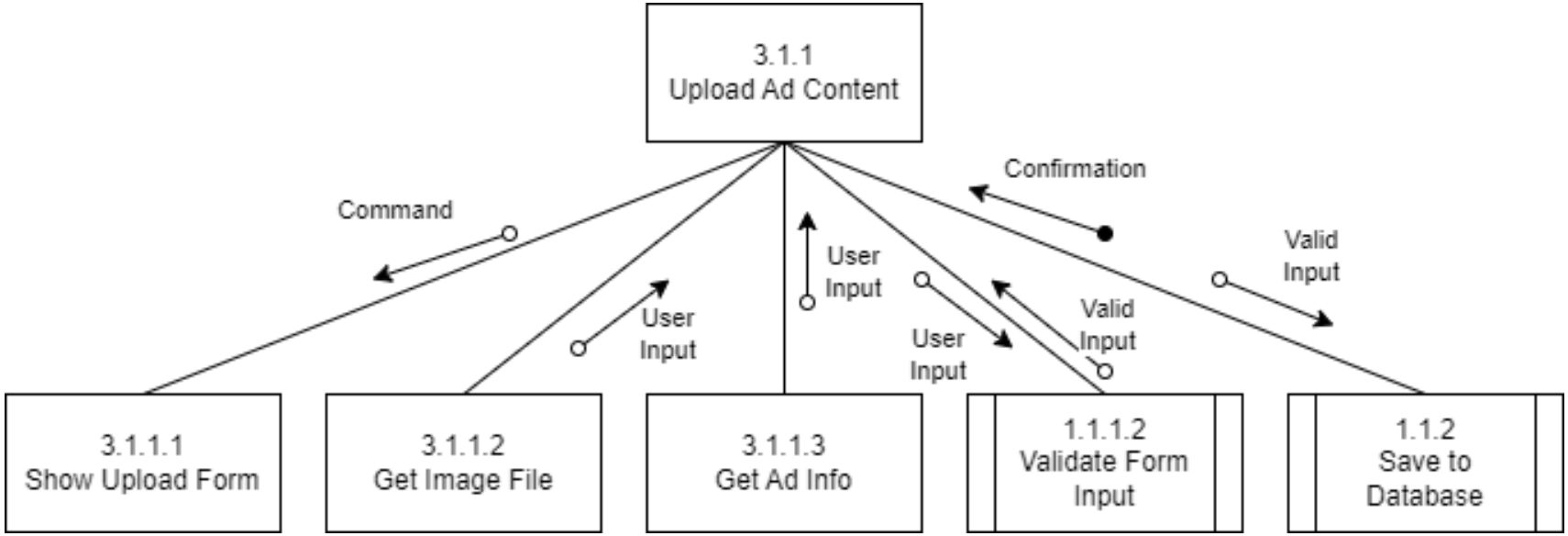



*Figure 3.2.3. - 1*

| | Program Specifications for | Open Road | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.1.1 Show Upload Form | | |
| **Purpose:** | Displays the form to upload Ad content | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP/HTML | | |
| **Events:** | User navigates to the desired form | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Upload Form | View | User | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| `adUploadForm.show()` | | | |

*Figure 7.2.2.3. – 2*

| | Program Specifications for | Open Road | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.1.2 Get Image File | | |
| **Purpose:** | Gets the Image File from the user | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | HTML | | |
| **Events:** | The user selects the image file to upload | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Image | Image File | User | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Image | Image File | 3.1.1.4 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| `<input type="file">` | | | |

*Figure 7.2.2.3. - 3*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.1.3 Get Ad Info | | |
| **Purpose:** | Gets info on the Ad from the user | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | HTML | | |
| **Events:** | The user inputs data in to a form | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Ad Title | String | User | |
| Ad Description | String | User | |
| Ad URL | URL | User | |
| Ad Location | Checkbox | User | |
| Start Date | Date | User | |
| End Date | Date | User | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| SQL Insert Query | SQL | 1.1.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
<input type="text" name="title" />
<input type="text" name="desc" />
<input type="text" name="url" />
<input type="checkbox" name="page" />
<input type="date" name="start" />
<input type="date" name="end" />
```

*Figure 3.2.3. – 4*

## Process 3.1.2 Update Static Content

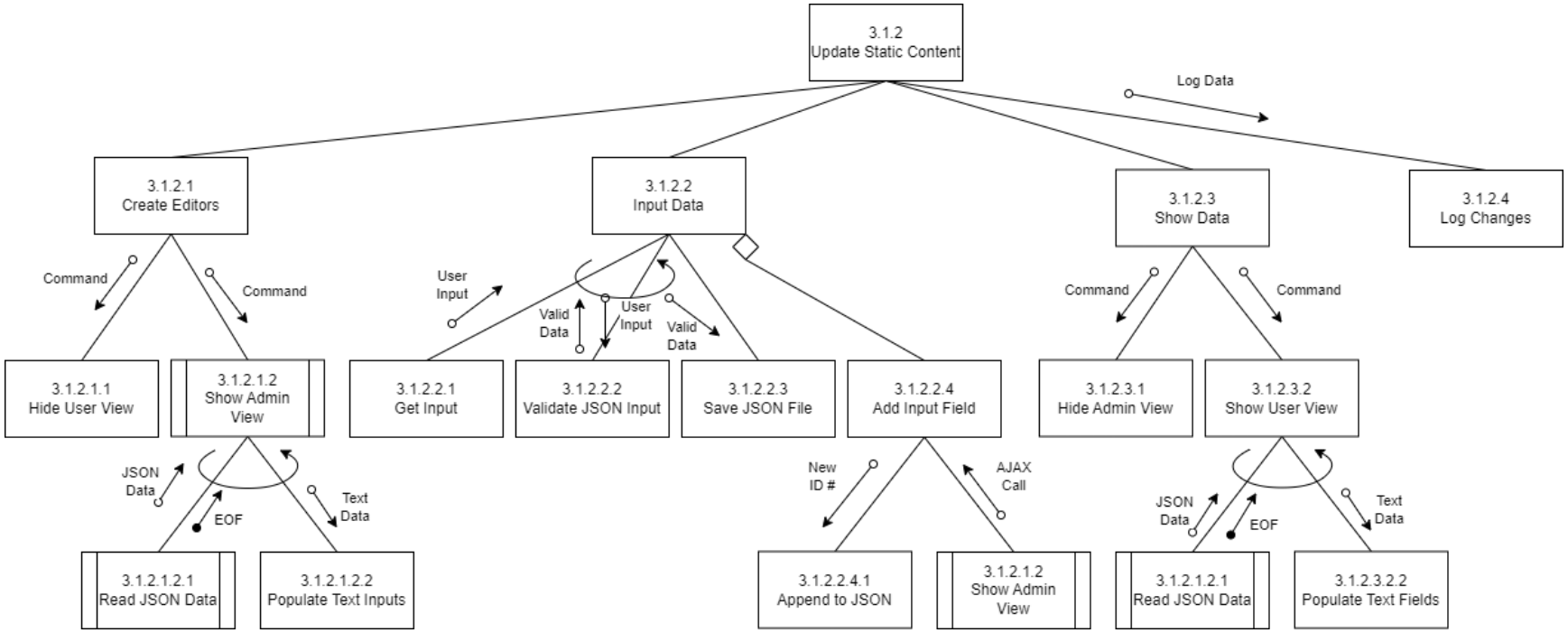



*Figure 7.2.2.3. – 5*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.1 Create Editors | | |
| **Purpose:** | Creates the text editors for editing static pages | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | Admin presses the button to edit static page content | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Button Press | Trigger | User | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Admin View | View | 3.1.2 | |
| **Pseudocode** | | | |

```
userView.hide()
adminView.show()
```

*Figure 7.2.2.3. – 6*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.1.1 Hide User View | | |
| **Purpose:** | Hide the text fields, so the text inputs may be visible | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | Command is sent from 3.1.2.1 | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Command | Trigger | 3.1.2.1 | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Current Page | String | 3.1.2.1.2 Show Admin View | |
| **Pseudocode** | | | |

```
change View to Admin View
```

*Figure 7.2.2.3. – 7*

| | | | |
|---|---|---|---|
| | **Program Specifications for** | Open Road | |
| **Module Name, number, etc:** | 3.1.2.1.2 Show Admin View | | |
| **Purpose:** | Show the text inputs and current values | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP/HTML | | |
| **Events:** | Command is sent from 3.1.2.1 | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Current Page | String | 3.1.2.1.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Current Page | String | 3.1.2.1.2.1 | |
| Admin View | View | 3.1.2 | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
adminView.show()
for each item in class
  readJSON(currentPage)
  fillInputs(textData)
```

*Figure 7.2.2.3. – 8*

| | | | |
|---|---|---|---|
| | **Program Specifications for** | Open Road | |
| **Module Name, number, etc:** | 3.1.2.1.2.1 Read JSON Data | | |
| **Purpose:** | Reads the data for the text inputs from the JSON file | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | JavaScript | | |
| **Events:** | Command sent from 3.1.2.1.2 | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Current Page | String | 3.1.2.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| JSON Data | String | 3.1.2.1.2.2 && 3.1.2.3.2.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
obj.faq[i].question
obj.faq[i].answer
```

*Figure 7.2.2.3. – 9*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.1.2.2 Populate Text Inputs | | |
| **Purpose:** | Fill the text inputs with the current values | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | JavaScript | | |
| **Events:** | Data sent from 3.1.2.1.2.1 | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Text Data | String | 3.1.2.1.2.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| `document.getElementById("question").value =` | | | |
| `obj.faq[i].question` | | | |
| `document.getElementById("answer").value =` | | | |
| `obj.faq[i].answer` | | | |
| | | | |
| | | | |

*Figure 7.2.2.3. - 10*

| | Program Specifications for | Open Road | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.2 Input Data | | |
| **Purpose:** | Add user input to the JSON files | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | JavaScript/AJAX | | |
| **Events:** | Editors have been shown, and user inputs data | | |

| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
|---|---|---|---|
| User Input | Text | User | |

| **Output Name:** | **Type:** | **Used by:** | **Notes** |
|---|---|---|---|

**Pseudocode**

```
if user adds field
 Append to JSON
 Show new field
getInput()
validate()
saveJSON()
```

*Figure 7.2.2.3 – 11*

| | Program Specifications for | Open Road | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.2.1 Get Input | | |
| **Purpose:** | Get the user input | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | HTML/JavaScript | | |
| **Events:** | The user inputs data in to the text inputs | | |

| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
|---|---|---|---|
| User Input | Text | User | |

| **Output Name:** | **Type:** | **Used by:** | **Notes** |
|---|---|---|---|
| User Input | Text | 3.1.2.2.2 | |

**Pseudocode**

```
<input type="text" />
```

*Figure 7.2.2.3. – 12*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.2.2 Validate JSON Input | | |
| **Purpose:** | Validates the input to match criteria | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP/JavaScript | | |
| **Events:** | The user inputs data | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| User Input | Text | User | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Validated Input | Text | 3.1.2.2.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
if input.length() > maxLength
 alert()
else
 return validText
```

Figure 7.2.2.3. – 13

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.2.3 Save JSON File | | |
| **Purpose:** | Saves the JSON file for later viewing | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | User presses the 'Save' button | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Text Inputs | Text | 3.1.2.2.2 | |
| Button Command | Button Press | User | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
for each (jsonarray as key => value)
 if (value['ID'] == i
  jsonarray[key][type] = data

file_put_contents('new.json', json_encode(jsonarray))
```

Figure 7.2.2.3. – 14

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.2.4 Add Input Field | | |
| **Purpose:** | Adds an Input Field for the user to add more content | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | User clicks to add another field. | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Command | Button Click | User | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Admin View | View | 3.1.2 | |

**Pseudocode**

```
jsonAppend()
AJAX{adminView.show()}
```

*Figure 7.2.2.3. – 15*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.2.4.1 Append to JSON | | |
| **Purpose:** | Appends the new field on to the JSON file | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | User clicks to add another field. | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Command | Button Click | User | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Array ID | Integer | 3.1.2.1.2 | |

**Pseudocode**

```
let i = jsonarray.size()
jsonarray[] = array('ID' => (i+1), data => '')
```

*Figure 7.2.2.3. – 16*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.3 Show Data | | |
| **Purpose:** | Show the updated Static Page | | |
| **Programmer:** | Claire | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | User clicks the Save Button | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Command | Button Click | User | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| `adminView.hide()` | | | |
| `userView.show()` | | | |

*Figure 7.2.2.3. – 17*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.3.1 Hide Admin View | | |
| **Purpose:** | Hide the admin view | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | User clicks the save button | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Command | Button Press | User | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Current Page | String | 3.1.2.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| `adminView.hide()` | | | |

*Figure 7.2.2.3. 18*

| | Program Specifications for | Open Road | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.3.2 Show User View | | |
| **Purpose:** | Shows the user view | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | User clicks the save button | | |

| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
|---|---|---|---|
| Command | Button Click | User | |
| Current Page | String | 3.1.2.3.1 | |

| **Output Name:** | **Type:** | **Used by:** | **Notes** |
|---|---|---|---|
| User View | View | 3.1.2 | |

**Pseudocode**

```
userView.show()
for each item in class
  readJSON(currentPage)
  fillFields(textData)
```

*Figure 7.2.2.3. – 19*

| | Program Specifications for | Open Road | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.3.2.2 Populate Text Fields | | |
| **Purpose:** | Fills the text fields for viewing | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | JavaScript | | |
| **Events:** | Data sent from 3.1.2.1.2.1 | | |

| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
|---|---|---|---|
| Text Data | String | 3.1.2.1.2.1 | |

| **Output Name:** | **Type:** | **Used by:** | **Notes** |
|---|---|---|---|

**Pseudocode**

```
document.getElementById("question").innerHTML =
obj.faq[i].question
document.getElementById("answer").innerHTML =
obj.faq[i].answer
```

*Figure 7.2.2.3. – 20*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.2.4 Log Changes | | |
| **Purpose:** | Logs changes to static pages | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | Static Page Content has been changed | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| User ID | Integer | System | |
| Page Edited | String | 3.1.2.1.2 | |
| Date | Date | System | |
| Time | Time | System | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Log Data | Text | Log File | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
logText = userID.toString + currentPage + date.toString + time.toString
logFile.append(logText)
```

*Figure 7.2.2.3. – 21*

Process 3.1.3. Remove Ad Content

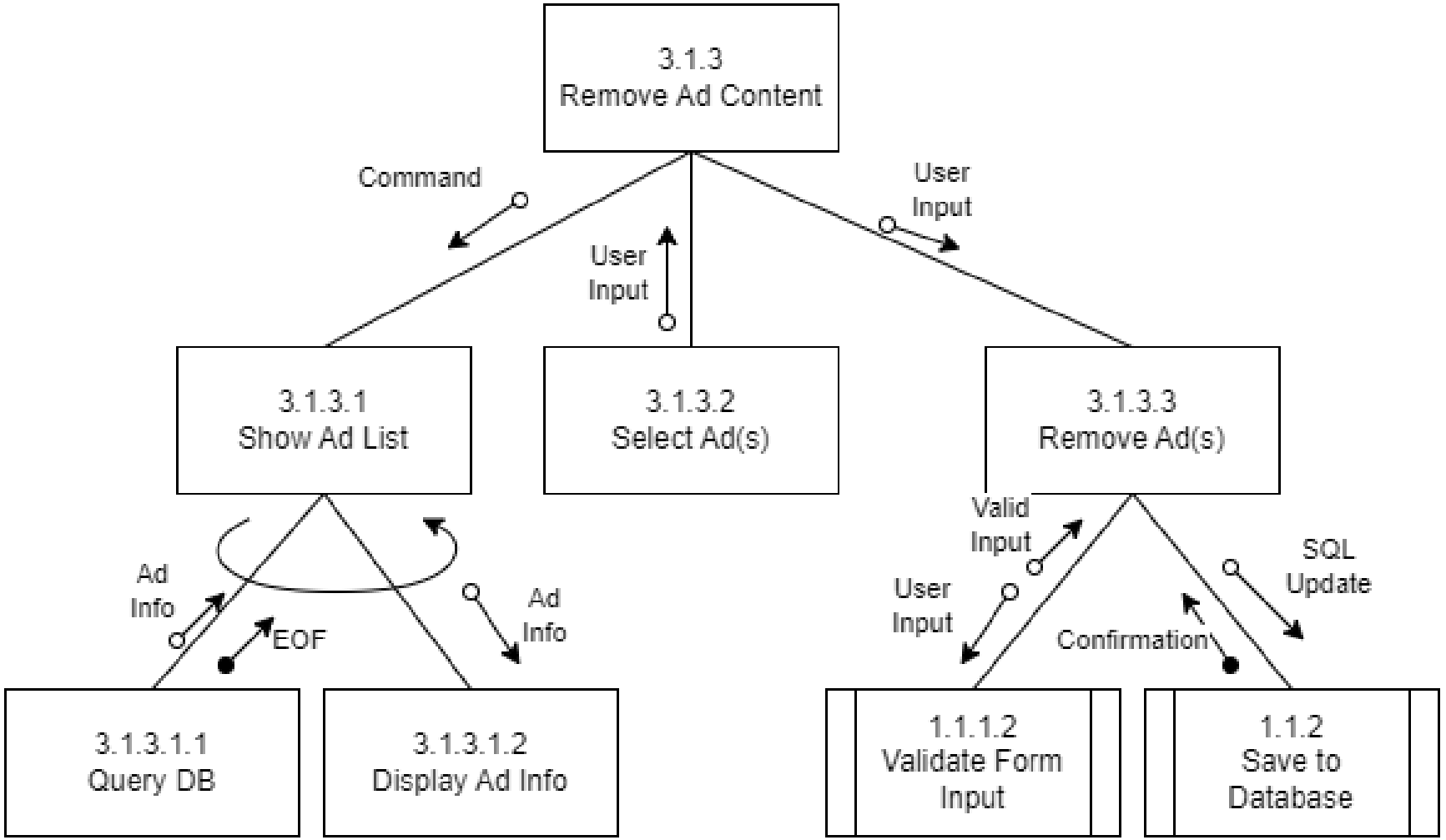



*Figure 7.2.2.3. – 22*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.3.1 Show Ad List | | |
| **Purpose:** | Shows the List of Current Advertisements | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | User presses button to remove ad content | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Button Press | Command | User | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Ad List | Text | 3.1.3 | |
| **Pseudocode** | | | |

```
queryDB()
dispAdInfo()
```

| **Other** |
|---|
| |

*Figure 7.2.2.3. – 23*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.3.1.1 Query DB | | |
| **Purpose:** | Pulls ad data from advertisements table | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | SQL | | |
| **Events:** | User views ad list | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Button Press | Command | User | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Ad Data | SQL | 3.1.3.1.2 | |
| **Pseudocode** | | | |

```
select title, desc, mime from advertisement where deleted is not null
```

| **Other** |
|---|
| |

*Figure 7.2.2.3. – 24*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.3.1.2 Display Ad Info | | |
| **Purpose:** | Displays the current list of advertisements | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | List of advertisements is requested | | |

| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
|---|---|---|---|
| Ad Info | SQL | 3.1.3.1.1 | |

| **Output Name:** | **Type:** | **Used by:** | **Notes** |
|---|---|---|---|
| Ad List | Text | 3.1.3.1 | |

**Pseudocode**

```
while (!EOF)
  echo <checkbox> $title - $desc <img src="$mime"/>
```

**Other**

*Figure 7.2.2.3. – 25*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.3.2 Select Ad(s) | | |
| **Purpose:** | Gets user to select ads | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | HTML | | |
| **Events:** | A list of current ads has been generated | | |

| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
|---|---|---|---|

| **Output Name:** | **Type:** | **Used by:** | **Notes** |
|---|---|---|---|
| Selected Ads | Checkbox | 3.1.3.3 | |

**Pseudocode**

```
for each ad in list
  <input type="checkbox" name="ad" value=i /><label>Ad Info</label>
form submit button
```

**Other**

*Figure 7.2.2.3. – 26*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.1.3.3 Remove Ad(s) | | |
| **Purpose:** | Soft deletes the ads from the database | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | User Presses the Confirm Button | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Selected Ads | Checkbox | 3.1.3.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Confirmation | System Flag | 3.1.3 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |

```
for each checkedAd
  validateInput(update deleted in advertisement)
  saveToDB(update deleted in advertisement)
```

*Figure 7.2.2.3. - 27*

Process 3.2.1.2. Create Automatic Backup

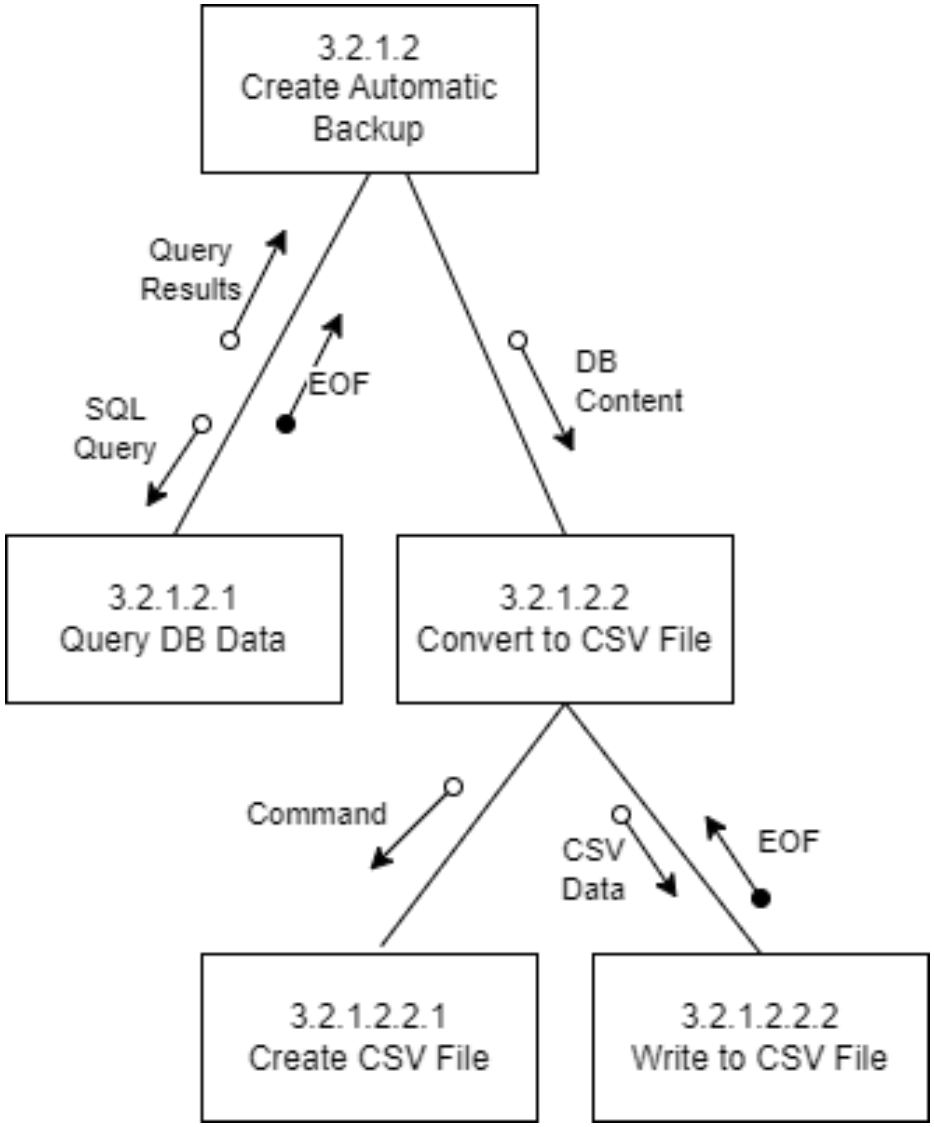



*Figure 7.2.2.3. – 28*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.2.1.2.1 Query Database Data | | |
| **Purpose:** | Pull data from the database for backup | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | SQL | | |
| **Events:** | Triggered by automatic cron job | | |

| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
|---|---|---|---|
| Database Query | SQL | CRON job | |

| **Output Name:** | **Type:** | **Used by:** | **Notes** |
|---|---|---|---|
| Database Data | SQL | 3.2.1.2 | |

**Pseudocode**

```
select * from *
```

**Other**

*Figure 7.2.2.3. – 29*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.2.1.2.2 Convert to CSV File | | |
| **Purpose:** | Converts DB data to CSV file | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | Triggered by automatic CRON job | | |

| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
|---|---|---|---|
| DB Content | SQL | 3.2.1.2.1 | |

| **Output Name:** | **Type:** | **Used by:** | **Notes** |
|---|---|---|---|
| Backup File | CSV File | 3.2.1.2 | |

**Pseudocode**

```
createCSV()
writeToFile()
```

**Other**

*Figure 7.2.2.3. 30*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.2.1.2.2.1 Create CSV File | | |
| **Purpose:** | Creates the CSV file for backup | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | Triggered by automatic CRON job | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Current Date | Date | system | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Backup File | CSV File | File System | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| `file.open(name=$date + ".csv")` | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Other** | | | |
| | | | |

*Figure 7.2.2.3. – 31*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.2.1.2.2.2 Write to CSV File | | |
| **Purpose:** | Writes database data to the CSV file | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | Triggered by automatic CRON job | | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
| Database data | SQL | 3.2.1.2.2.1 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Output Name:** | **Type:** | **Used by:** | **Notes** |
| Backup File | CSV File | 3.2.1.2 | |
| | | | |
| | | | |
| | | | |
| | | | |
| **Pseudocode** | | | |
| `for each table in database`<br>`  for each line in table`<br>`  append.csv(tableData)` | | | |
| | | | |
| | | | |
| **Other** | | | |
| The sql for this one will be a bit more exhaustive than the pseudocode suggests, but this should be clear enough to start with | | | |

*Figure 7.2.2.3. – 32*

Process 3.2.2. Restore Backup

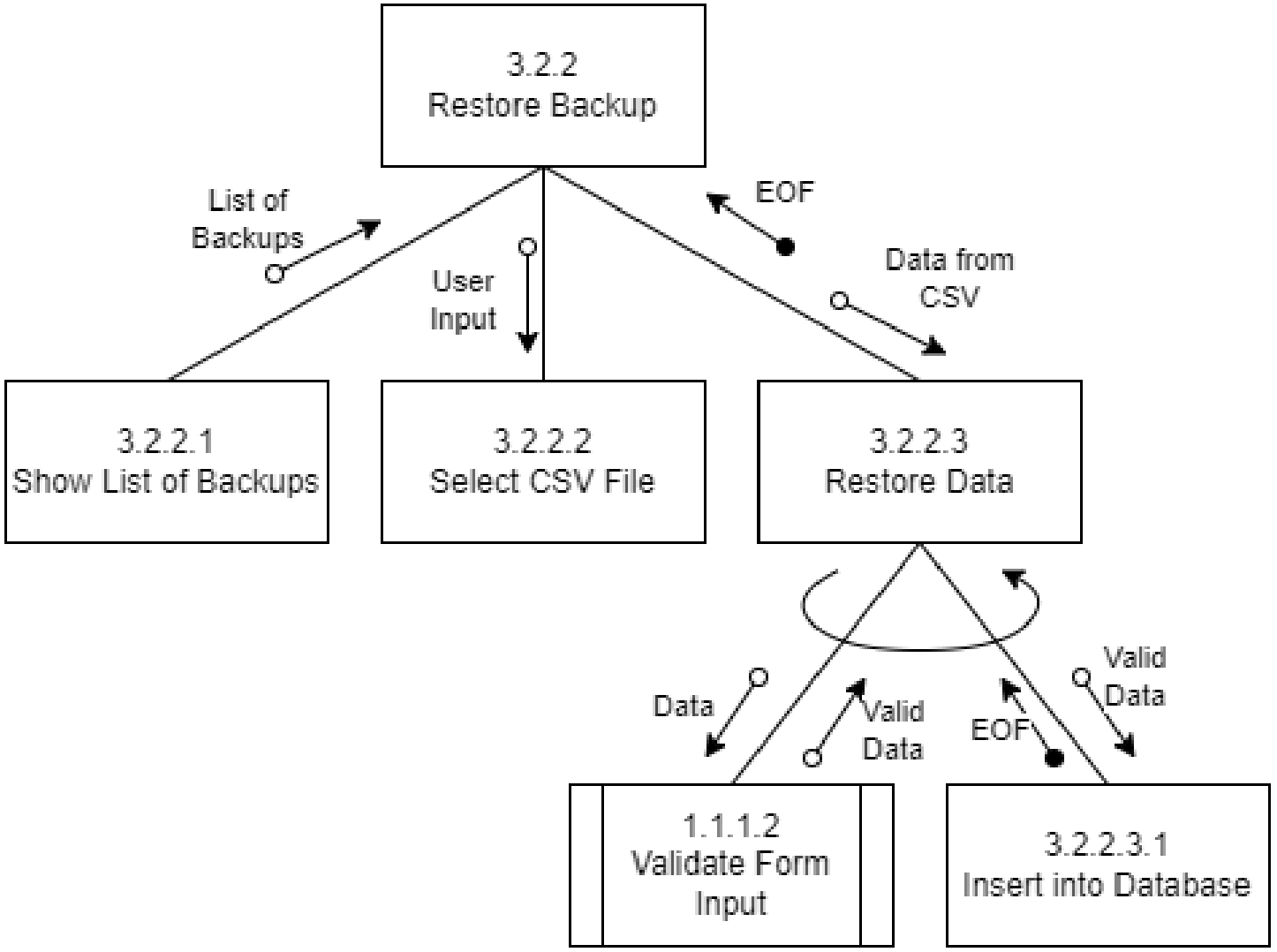



*Figure 7.2.2.3. - 33*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.2.2.1 Show List of Backups | | |
| **Purpose:** | Displays a list of available backups | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | PHP | | |
| **Events:** | Displayed when the user selects to restore a backup | | |

| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
|---|---|---|---|
| Command | Button Press | User | |

| **Output Name:** | **Type:** | **Used by:** | **Notes** |
|---|---|---|---|
| List of Backups | Text | 3.2.2.2 | |

**Pseudocode**

```
for each file in ~/backups/
  <input type="radial" name="backup" value=$fileName /> { $fileName }
```

**Other**

*Figure 7.2.2.3. – 34*

| Program Specifications for | Open Road | | |
|---|---|---|---|
| **Module Name, number, etc:** | 3.2.2.2 Select CSV File | | |
| **Purpose:** | Gets the user selection of CSV file | | |
| **Programmer:** | Claire Fleckney | | |
| **Date due:** | | | |
| **Language/platform:** | HTML | | |
| **Events:** | List of available backups has been displayed | | |

| **Input Name:** | **Type:** | **Provided by:** | **Notes** |
|---|---|---|---|
| User Input | Radial Button | User | |

| **Output Name:** | **Type:** | **Used by:** | **Notes** |
|---|---|---|---|
| File Name | Text | 3.2.2.3 | |

**Pseudocode**

```
<input type="radial" name="backup" value=$file />
form submit
```

**Other**

*Figure 7.2.2.3. – 35*

| | |
|---|---|
| **Program Specifications for** | Open Road |
| **Module Name, number, etc:** | 3.2.2.3 Restore Data |
| **Purpose:** | Restores data to the database |
| **Programmer:** | Claire Fleckney |
| **Date due:** | |
| **Language/platform:** | PHP |
| **Events:** | User has selected which CSV file to upload in to the database |

| Input Name: | Type: | Provided by: | Notes |
|---|---|---|---|
| Backup Data | CSV File | 3.2.2.2 | |

| Output Name: | Type: | Used by: | Notes |
|---|---|---|---|
| Restored Database | SQL | 3.2.2 | |

**Pseudocode**

```
validateData($csvFile)
insertData($csvFile)
```

**Other**

*Figure 7.2.2.3. – 36*

| | |
|---|---|
| **Program Specifications for** | Open Road |
| **Module Name, number, etc:** | 3.2.2.3.1 Insert into Database |
| **Purpose:** | Inserts the backup data into the database |
| **Programmer:** | Claire Fleckney |
| **Date due:** | |
| **Language/platform:** | SQL |
| **Events:** | Data has been validated |

| Input Name: | Type: | Provided by: | Notes |
|---|---|---|---|
| Validated Data | Text | 1.1.1.2 | |

| Output Name: | Type: | Used by: | Notes |
|---|---|---|---|
| Database Insertion | SQL | 3.2.2 | |

**Pseudocode**

```
for each row in CSV
  insert $data into $table
```

**Other**

*Figure 7.2.2.3. – 37*

## 7.3 Physical Architecture

The collection of hardware and software that together create the "Open Road" information system will be structured as presented in the following diagram (see figure 4.1. -1). Using a Client-Server base architecture, any user who wishes to connect to and use the application will only be required to have a modern web browser and an active network connection. The only other requirement outside the server requirements will be that the terminal on-site at the Academy will need adequate storage space to hold the on-stie backup file. All other requirements listed in the next section are specific to the hardware and software needed by the server for running and maintaining the web application.

### 7.3.1. System Architecture

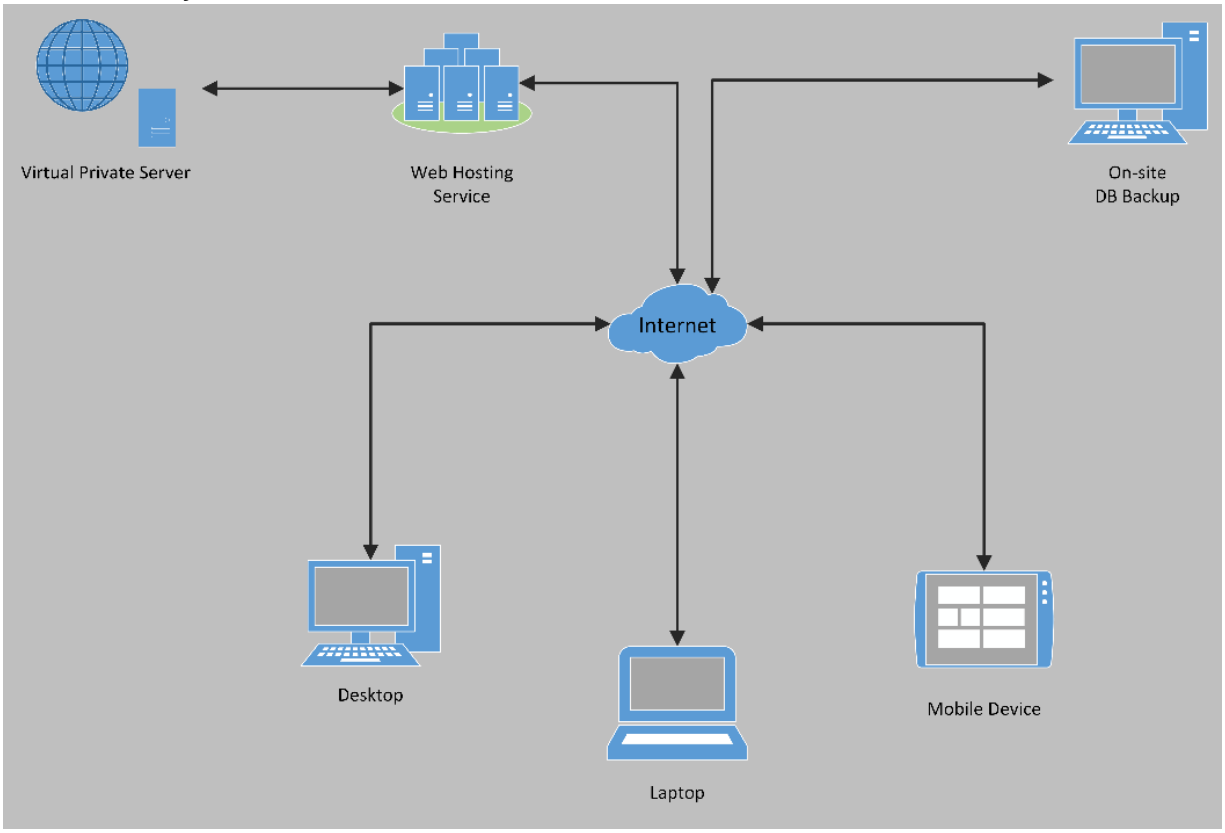



*Figure 7.3.1. - 1*

## 7.3.2. Hardware & Software Specifications

### 7.3.2.1. Software Specifications– Virtual Private Server

- Operating System: Linux based OS
    - Smaller memory and storage requirement than MSDOS (Windows). Being an open-source software has allowed Linux based operating systems to reach a higher level of reliability in terms of performance, customizability, and security.

- Web site/application Container Service: Docker
    - Docker will provide services that allow multiple web sites or applications to run within the same operating system without being able to affect each other in a negative fashion. It also provides a platform for running services that may be required by a particular site or application but don't necessarily need to be installed on the server as a whole. This saves system resources such as computing power, memory, and storage space, without sacrificing functionality.
    - Services to be run by Docker: Redis, Mailhog, others.

- Database Server: MariaDB Server
    - MariaDB server will provide the SQL database services required to operate the application. Using the logical and physical designs prepared in earlier milestones, an appropriate supporting database will be implemented within the MariaDB server instance to allow the "Open Road" web application to provide the services specified in the system requirements.
- Anti-Virus/Anti-Malware: TBD
    - While an appropriate anti-virus/anti-malware service is required to prevent, diagnose, and resolve potential threats and vulnerabilities that ay target the system or its resources, the specific service used will be determined by the web hosting service provider. Most providers will or can include anti-virus/anti-malware as part of the package purchase by the customer.

- Network Connection Security: SSL Site Certificate (REQUIRED)
    - Due to the sensitive nature of the data which will be collected, stored, and used by the information system, it is a requirement that the "Open Road" application operate under a Secure Socket Layer (SSL) certificate. These SLL certificates ensure that any network communication between the client devices and server are over secured and encrypted using HTTPS (Hyper Text Transport Protocol Secure) protocols. HTTPS reduces the likelihood that data will be intercepted during transmission between client and server, and should any data be intercepted the data encryption will reduce the risk of malicious individuals obtaining sensitive information.
- Primary Server Language: PHP >8.1
    - The web application framework that will be used to create the application, Laravel, is primarily written in the PHP server scripting language. To ensure proper operation, version 8.1 of PHP is required to be present and installed on the server.

#### 4.3.2.2 Hardware Specifications

Virtual Private Servers provide an excellent balance between the affordability of shared webhosting, and the reliable performance of a dedicated server. VP Servers share the same physical space as other VP Servers, but do not interact with each other, thereby increasing the reliability of your web services and ensuring availability of system resources. It is also much easier to increase the size of your current system resources should they prove to be inadequate in the future.

- Processing:
    - Minimum 4 virtual CPU cores.
- Main Memory:
    - Minimum 16GB RAM.
- Storage:
    - Minimum 200GB.
- Network Connection:
    - Min 1GBs, with redundant backups.

## 7.4. Conclusion

Throughout this milestone we worked on developing a site map and user interface mock-up which would facilitate implementation of system processes and views. Emphasis was placed on ensuring consistency is maintained between the existing Precision Power Sports site and the newly hosted Precision Riding Academy website, so that transitioning between the business pages remains seamless. Before creating our structure charts and program specification sheets we finalized our hardware and software specifications, to better understand how the processes being described will be implemented.

After finalizing our applications architecture, we transitioned our previously designed logical data flow diagrams into physical diagrams by adding implementation references and human-machine barriers to better describe the system as we now understand it. Once our DFDs were fully constructed, we used them to design our structure charts, from which we derived our program specification sheets. These sheets outline the steps we will take throughout Milestone 8 to develop and implement the system. Milestone 7 greatly helped to develop our understanding of the task at hand as well as helping us to realize mistakes that were made during the logical analysis and design phase.

# 8. Milestone Eight

Table of Contents

## 8.1. Coding Experience

### 8.1.1. Highlights

Due to the highly modular nature of the "Open Road" system, and the Laravel MVC framework, one of the biggest highlights of the group coding experience was the fact that each team member could almost exclusively focus on one section of the system at a time. Rather than having a group of people having to cooperate in making the front-end, back-end, and database work together, each team member was assigned a "section" of the site. With the design of the Laravel framework, writing all crud processes for one object in the system is much simpler than it would be with other tech stacks. Once the database and base layout were built, all team members were able to work mostly independently.

### 8.1.2. Challenges

Due to having to learn the unfamiliar Laravel framework, and initial difficulties in setting up a complicated development environment, our team started this milestone behind schedule. In addition, most of the work that we had accomplished in Milestone 7, especially the Structure Charts and Program Specifications weren't useful, as they were written without a full understanding of the MVC framework.

Another challenge that was faced by the team was a lack of communication. While the system is highly modular, there is still some overlap between the different areas. Without proper communication channels in place, there were some miscommunications and conflicts in the written code.

### 8.1.3. Code Review

After having our peers review our code, we received suggestions on what could be improved in the system. It was suggested that since our chosen framework interacts with a database so seamlessly, it would be easier to implement things such as the FAQ and Meet the Team page as database components with controllers, rather than having that information stored in a JSON file. They also suggested that added security around the system would be beneficial, so now many components in the system are protected by multiple layers of role and permission based security.

# 8.2. Updates to Documentation

## 8.2.1. Data Dictionary

Due to many changes in the way that our system interacted with the database that were discovered during the coding process, and a continued learning of the Laravel framework, many changes were made to our database. These changes can be reflected most accurately in our updated Data Dictionary. The following tables have been modified or added since our Data Dictionary was last submitted:

| Table Name | Attribute Name | Description | Data Type | Format/Mask | Required | PK or FK | FK Ref Table | Sample Data | |
|---|---|---|---|---|---|---|---|---|---|
| users | id | ID for the row | char(36) | | Y | | | | *table changed to reflect Laravel standards |
| | name | User Name | varchar(255) | | Y | | | | |
| | email | User email address | vharchar(255) | | Y | PK | | | |
| | email_verified_at | When the email was verified | Timestamp | | | | | | |
| | password | User Password | varchar(255) | | Y | | | | |
| | two_factor_secret | Two Factor Authentication | text | | | | | | |
| | two_factor_revocer_codes | Recovery codes | text | | | | | | |
| | two_factor_confirmed_at | Confirmation date of 2FA | Timestamp | | | | | | |
| | created_at | When the user was created | Timestamp | | | | | | |
| | updated_at | When the user was last updatesd | Timestamp | | | | | | |
| stuff | id | ID for the row | char(36) | | Y | | | | *table changed to reflect Laravel standards |
| | STUFF_ID | Unique Code to identify User | Char(5) | X9999 | Y | PK | | E4321 | |
| | STUFF_LEVEL | Level of site access for the User | Char(1) | X | Y | | | A | |
| | STUFF_PNAME | Preferred name of the User | Varchar(20) | | | | | Bobby | |
| | STUFF_FNAME | First name of the User | Varchar(20) | | Y | | | Bob | |
| | STUFF_LNAME | Last name of the User | Varchar(20) | | Y | | | Smith | |
| | STUFF_PHONE | Phone number of the User | Char(12) | 999-999-9999 | Y | | | 403-123-4567 | |
| | STUFF_EMAIL | Email address of the User | Varchar(40) | | Y | | | bobsmith@bob.com | |
| | STUFF_DOB | Users Date of Birth | Date | | | | | | |
| | STUFF_DLN | User DL# | char(10) | | | | | | |
| | STUFF_ADDR1 | Line one of the User's address | Varchar(30) | | Y | | | 123 Cherry Lane | |
| | STUFF_ADDR2 | Line two of the User's address | Varchar(30) | | | | | Appt 64 | |
| | STUFF_CITY | City the User lives in | Varchar(30) | | Y | | | Lethbridge | |
| | STUFF_PR_ST | Province or State the User member lives in | Char(2) | XX | Y | | | AB | |
| | STUFF_COUNTRY | Country the Staff member lives in | Varchar(30) | | Y | | | Canada | |
| | STUFF_POST_ZIP | Postal or Zip Code of the Staff member | Varchar(7) | | Y | | | A1B C2D | |
| | EMERGCONT_NAME | Name of the Staff members emergency contact | Varchar(40) | | | | | John Doe | |
| | EMERGCONT_PHONE | Phone number of the Staff members emergency contact | Char(12) | 999-999-9999 | | | | 403-987-6543 | |
| | user_id | User Account ID | char(36) | | Y | | | | |
| | created_at | When the account was created | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| | updated_at | When the account was last updated | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| | deleted_at | When the account was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| transaction_log | TRANSACTION_ID | Autogenerated number | UnsignedInt(5) | 99999 | Y | PK | | 54321 | *table changed to reflect Laravel standards |
| | id | ID for the row | char(36) | | Y | | | | |
| | TRANSACTION_DATE | Date the transaction occurred | Date | yyyy-mm-dd | Y | | | 1932-02-29 | |
| | TRANSACTION_STIME | Time the user logged in to their account | Time | 99:99:99 | Y | | | 15:26:50 | |
| | TRANSACTION_IP_ADDR | IP address of the user initiating transaction | Char(15) | 999.999.999.999 | Y | | | 172.022.002.029 | |

*Table 8.2.1-1 – Updated Data Dictionary Tables*

| Table Name | Attribute Name | Description | Data Type | Format/Mask | Required | PK or FK | FK Ref Table | Sample Data | |
|---|---|---|---|---|---|---|---|---|---|
| courses | id | ID for the row | Char(36) | | Y | PK | | | *table changed to reflect Laravel standards |
| | COURSE_ID | ID number for the course | Char(3) | 999 | Y | | | 101 | |
| | COURSE_NAME | Name associated with the course | Varchar(15) | | Y | | | Beginner | |
| | COURSE_DOCS | File path to docs related to the course | Varchar(256) | | | | | \Beginner\Docs\ | |
| | COURSE_MAX_SEATS | Maximum number of seats for the course | UnsignedInt(10) | 9 | Y | | | 5 | |
| | COURSE_FEE | Cost to register for the course | Double(5,2) | 999.99 | Y | | | 259.99 | |
| | created_at | When the account was created | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| | updated_at | When the account was last updated | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| | deleted_at | When the account was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| pra_classes | id | ID for the row | char(36) | | Y | | | | *table changed to reflect Laravel standards |
| | CLASS_ID | ID number for the class | Char(6) | yy-999 | Y | PK | | 23-101 | |
| | COURSE_ID | Course type of the class | Char(3) | 999 | Y | FK | course | 101 | |
| | PRIMARY_INST | Stuff ID of the main instructor | Char(5) | X9999 | Y | FK | stuff | E4321 | |
| | SECONDARY_INST | Stuff ID of the assistant instructror | Char(5) | X9999 | | | | E4321 | |
| | CLASS_START | Start date of the class | Date | yyyy-mm-dd | Y | | | 2023-05-20 | |
| | CLASS_END | End date of the class | Date | yyyy-mm-dd | Y | | | 2023-05-23 | |
| | created_at | When the account was created | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| | updated_at | When the account was last updated | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| | deleted_at | When the account was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| registration | REG_ID | Autogenerated number | Char(4) | 9999 | Y | PK | | 1917-12-04 | *table changed to reflect Laravel standards |
| | CLASS_ID | ID number for the class | Char(6) | yy-999 | Y | FK | class | 23-101 | |
| | INVOICE_ID | ID number of the invoice | UnsignedInt(4) | 9999 | Y | FK | invoice | 5678 | |
| | STUFF_ID | ID number for the student | Char(5) | X9999 | Y | FK | stuff | S1234 | |
| | VEHICLE_STOCK_NUM | Vehicle the student is initially assigned | Varchar(17) | XXX-999 | Y | FK | vehicle | YAM-123 | |
| | REGISTRATION_CANCELLED | Whether the registration has been cancelled | UnsignedInt(1) | 9 | Y | | | 0 | |
| | REGISTRATION_WAITLIST | Whether the registration has been waitlisted | UnsignedInt(1) | 9 | Y | | | 0 | |
| | created_at | When the account was created | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| | updated_at | When the account was last updated | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| | deleted_at | When the account was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| invoice | INVOICE_ID | ID number of the invoice | UnsignedInt(4) | 9999 | Y | PK | | 5678 | *table changed to reflect Laravel standards |
| | PAYMENT_SUBTOTAL | Subtotal before tax | Double(5,2) | 999.99 | Y | | | 259.99 | |
| | PAYMENT_TAX_AMT | Percentage of Sales Tax to be applied | Double(3,2) | 0.99 | Y | | | 0.05 | |
| | PAYMENT_TYPE | Method of payment | Varchar(15) | | Y | | | PayPal | |
| | created_at | When the invoice was created | Timestamp | | | | | | |
| | updated_at | When the invoice was last updated | Timestamp | | | | | | |
| | deleted_at | When the invoice was soft deleted | Timestamp | | | | | | |
| vehicle | id | ID for the row | char(36) | | Y | | | | *table changed to reflect Laravel standards |
| | VEHICLE_STOCK_NUM | Stock Number of the vehicle | Varchar(17) | XXX-999 | Y | PK | | YAM-123 | |
| | VEHICLE_VIN | VIN of the vehicle | Varchar(17) | 9XXXX99XXXX999999 | Y | | | 5YJSA1DG9DFP14705 | |
| | VEHICLE_YEAR | Year the vehicle was manufactured | Char(4) | 9999 | Y | | | 2002 | |
| | VEHICLE_MAKE | Make of the vehicle | Varchar(20) | | Y | | | Kawasaki | |
| | VEHICLE_MODEL | Model of the vehicle | Varchar(40) | | Y | | | Ninja | |
| | VEHICLE_ODO | Odemeter reading of the vehicle | Char(10) | 9999999999 | Y | | | 0002564821 | |
| | VEHICLE_TYPE | Type of vehicle | UnsignedInt(1) | 9 | Y | | | 3 | |
| | VEHICLE_COLOR | Color of the vehicle | Varchar(10) | | | | | Red | |
| | VEHICLE_SIZE | Engine displacement of the vehicle | Varchar(4) | 9999 | Y | | | 750 | |
| | AVAIL_STATUS | Availability status of the vehicle | UnsignedInt(1) | 9 | Y | | | 1 | |
| | NOTES | Notes on the vehicle | varchar(256) | | | | | | |
| | created_at | When the account was created | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| | updated_at | When the account was last updated | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| | deleted_at | When the account was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |

*Table 8.2.1-2 – Updated Data Dictionary Tables*

| Table Name | Attribute Name | Description | Data Type | Format/Mask | Required | PK or FK | FK Ref Table | Sample Data | |
|---|---|---|---|---|---|---|---|---|---|
| damage_report | INCIDENT_ID | ID# of the incident causing damage | UnsignedInt(4) | 9999 | Y | PK/FK | incident | 1346 | *table changed to reflect Laravel standards |
| | VEHICLE_STOCK_NUM | Stock Number of the vehicle | Varchar(17) | XXX-999 | Y | PK/FK | | YAM-123 | |
| | DAMAGE_DESCRIPTION | Description of the damage | mediumtext | | | | | | |
| advertisement | id | ID# of the advertisement | UnsignedInt(10) | | Y | PK | | | *table changed to reflect Laravel standards |
| | USER_ID | Unique Code to identify users | UnsignedInt(4) | 9999 | Y | FK | user_login | 1234 | |
| | AD_TITLE | Title of the advertisement | Varchar(20) | | Y | | | X-Mas Sale | |
| | AD_DESCRIPTION | Description of the advertisement | Varchar(256) | | Y | | | Sale for x-mas 2023 | |
| | AD_URL | URL that the advertisement would link to | Varchar(256) | | | | | https://precisionpowersportsltd.com/ | |
| | AD_PATH | Path that the advertisement image is linked from | Varchar(256) | | Y | | | Pictures/Ads/Xmas | |
| | AD_SPACE_TYPE | Type of ad (horizontal, vertical, or corner square) | UnsignedInt(1) | 9 | Y | | | 2 | |
| | START_DATE | Date the ad should start running (if left blank, default to currer | Date | yyyy-mm-dd | Y | | | 1932-02-29 | |
| | END_DATE | Date the ad should stop running | Date | yyyy-mm-dd | | | | 1932-02-29 | |
| | created_at | When the advertisement was created | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| | updated_at | When the advertisement was last updated | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| | deleted_at | When the advertisement was soft-deleted | Date | yyyy-mm-dd hh:mm:ss | | | | 1932-02-29 16:23:50 | |
| f_a_q_s | id | ID for the row | bigint(20) | | Y | PK | | | *table added to ease access to data |
| | QUESTION | The question | varchar(128) | | Y | | | | |
| | ANSWER | The answer | varchar(1024) | | Y | | | | |
| | created_at | When the FAQ was creaeted | Timestamp | yyyy-mm-dd hh:mm:ss | | | | | |
| | updated_at | When the FAQ was last updated | Timestamp | yyyy-mm-dd hh:mm:ss | | | | | |
| | deleted_at | When the FAQ was soft-deleted | Timestamp | yyyy-mm-dd hh:mm:ss | | | | | |
| meet_teams | id | ID for the row | bigint(20) | | Y | PK | | | *table added to ease access to data |
| | NAME | Name of the Staff member | varchar(50) | | Y | | | | |
| | ROLE | Role of the Staff member | varchar(128) | | Y | | | | |
| | BIO | Biography of the Staff member | varchar(2048) | | Y | | | | |
| | FILEPATH | File path for the Staff members picture | varchar(256) | | | | | | |
| | created_at | When the Staff member was created | Timestamp | | | | | | |
| | updated_at | When the Staff member was last updated | Timestamp | | | | | | |
| | deleted_at | When the Staff member was soft-deleted | Timestamp | | | | | | |

*Table 8.2.1-3 – Updated Data Dictionary Tables*

## 8.3. Test Plan

### 8.3.1. Plan

Since the "Open Road" system is very modular, each module can be tested almost independently from the others. Some portions of the system, such as the Database, User Interface, and Permission Management, are used throughout the system, however, and therefore need to be almost fully created and functional before the rest of the system is implemented and tested.

Aside from the mentioned portions of the system, the rest of the functionality is mostly focused around creating, reading, updating, and deleting objects and data (hereafter referred to as CRUD). Because most of the system is centered around CRUD functions, testing is fairly straightforward, and was mostly done, undocumented, while programming.

As the System is quite large, a full test of all system functions would likely need to be done in multiple sessions. After a full test has been completed, all deficiencies, bugs, and suggested usability improvements will be considered for implementation.

### 8.3.2. Test Forms

To aid in the testing process, a test form has been created that can be used for each of the modules that focus on CRUD. These forms track how fully implemented the CRUD functionality is for each module. Additionally, each form has a section for tracking how well implemented the User Interface is, and whether it meets design standards and fits within the rest of the system.

# SIT SOFTware Test Form

Module Being Tested: _____ Tester Name: _____

## Create Functionality

| | | |
|---|---|---|
| Fully Functional | ☐ | Additional Notes: |
| Partially Functional | ☐ | |
| Not Functional | ☐ | |
| Missing With Stub | ☐ | |
| Missing Without Stub | ☐ | |

## Read Functionality

| | | |
|---|---|---|
| Fully Functional | ☐ | Additional Notes: |
| Partially Functional | ☐ | |
| Not Functional | ☐ | |
| Missing With Stub | ☐ | |
| Missing Without Stub | ☐ | |



*Figure 8.3.2-1 – First Page of the Test Form*

## Update Functionality

| | | |
|---|---|---|
| Fully Functional | ☐ | Additional Notes: |
| Partially Functional | ☐ | |
| Not Functional | ☐ | |
| Missing With Stub | ☐ | |
| Missing Without Stub | ☐ | |

## Delete Functionality

| | | |
|---|---|---|
| Fully Functional | ☐ | Additional Notes: |
| Partially Functional | ☐ | |
| Not Functional | ☐ | |
| Missing With Stub | ☐ | |
| Missing Without Stub | ☐ | |



*Figure 8.3.2-2 – Second Page of the Test Form*

UI/UX

Meets Basic Design Standards ☐
Consistent Across Pages ☐
Easily Understandable ☐
Easily Learnable ☐
Additional Notes:

Any Additional Notes



*Figure 8.3.2-3 – Third Page of the Test Form*

## 8.4. Conclusion

During this milestone we completed the majority of the development of the system. Most of the major functions have been coded and tested. While some minor functionality and some security concerns remain, we are confident that we can complete all work in time to begin training the client on the new system.

# 9. Milestone Nine

Table of Contents

## 9.1. Training Plan

This document serves to describe the proposed plan for training new users of SIT SOFTware's Open Road web-application. It lays out in detail the subject of the initial trial training session as well as the proposed setting and timing for when this training session is to take place, followed by a proposal for an official training session. It also details the materials that will be required for the initial training instances as well as what materials will be required to be provided for the successful ongoing operation of the application before finally presenting the challenges we perceive will be faced as well as some proposed solutions.

### 9.1.1. Subject, Setting, & Timing

The trial training process is to be carried out with a single individual, Sanjeev Prabhakar. The subject is a young male IT student, with an above average insight into the design and implementation of custom business applications as a result of his education. His background as a young student also affords him an above average technical ability from the perspective of a user which may influence his expectations both in terms of user interface and overall system functionality.

The training is tentatively scheduled to take place on Tuesday, 2023-03-28 at 1:00pm. The proposed location is the Lethbridge College Instructional Building, either in room IB1133 or a breakout room, depending on the availability of the classroom. The location requirements are that there is adequate room for all members of both teams to witness or participate in the training, as well as adequate technical amenities to facilitate the training as described in the material requirement section of this document. The training of the user is expected to take no more than 45 minutes to complete.

The currently standing proposal for training the actual client will involve either a member of upper management, or a course instructor, depending on the availability of either. The method of delivery will remain the same as the trial session, following the same script and general guidelines to ensure efficacy and timeliness. Due to the small number of trainable staff, we feel a one-on-one approach will remain the most effective method of delivery, ensuring that there is a focal point within the business for knowledge regarding the system, from which further members of staff can be trained or advised as needed. Training is tentatively set to take place on April 14, after we have a fully implemented system on a test server.

### 9.1.2 Training Method & Required Materials

The selected method of training delivery is a single, one-on-one, in-person training session. We chose this method of training for several reasons, not the least of which is the fact that there are very few people to be trained which mitigates one of the pitfalls of 1-on-1 training, which is reach. The fact that there would be no cost of labor involved rendered the cost of delivery irrelevant, entirely skirting another common problem with this method of delivery. Another benefit of this delivery format is that the individual time spent pouring through aspects of the system will

prove more beneficial to developers as feedback on how to improve aspects of the system as well as a way of finding bugs and inconsistencies that have yet to be noticed.

The materials required to successfully complete the training session include:

- A laptop or desktop computer.
- A secondary monitor to duplicate the users display for developers to view and take notes.
- An up-to-date version of the Open Road application running in an environment populated with test data.
- A comprehensive user manual as written by the system developers.

To ensure the successful ongoing use of the newly implemented system a copy of the user manual will be provided to the actual client (Precision Riding Academy) as well as the teams general, shared email address as an initial point of contact, and the personal information of any consenting Open Road developers.

### 9.1.3. Perceived Challenges & Solutions

The development team anticipates several challenges which are likely to arise from delivering the training using this method at this point in the systems life cycle. The following possible problems have been identified and suggested solutions have been attached below:

1) a. Problem: Due to internal communication challenges accelerated by a short timeline for delivery, assuring the attendance and participation of all team members remains challenging.

b. Solution: An attempt was made to facilitate the attendance of the entire team by scheduling the training session during regular class times, at the same venue. This effort should be maintained throughout the genuine training effort to ensure participation.

2) a. Problem: Unfinished or buggy system components impeding the proper understanding of the system.

b. Solution: Debugging efforts can be increased prior to training, with currently unfixable aspects being stubbed off in the code and referenced briefly in the user documentation.

3) a. Problem: The test subject might feel overwhelmed by the large number of people scheduled to view them conducting their training.

b. Solution: Accommodate witnesses to the training by duplicating the subjects display so that viewers are spread, not watching directly over the shoulder of the subject.

### 9.1.4. Training Script

Introduction

Hello [trainee], my name is [trainer]. It's a pleasure to meet you, I hope you are looking forward to todays training session. I'll start off by providing you with some context about the application and how it aims to improve your daily workflow, before going into detail about specific aspects of the application that are relevant to you as an administrative user. If you have any questions throughout the training process, please let us know immediately.

The goal of today's training is to familiarize you with the Open Road web application to a point where you would feel comfortable helping others to navigate and make effective use of the system. The current version of the system is largely an administrative tool through which you can define courses, from which classes can be instantiated. The application also includes comprehensive user management and authorization systems that will effectively restrict and allow access to resources as needed. These tools, along with many others, are nested into our admin dashboard into which we will do a deep-dive shortly.

Landing Page Tour

We'll start as the average user would, at the sites home page. This landing point contains some standard components as well as some content that is unique to this page. The components that are standard across all pages include:

- The site header, with contact/location information as well as the sites logo, which links back to this homepage from anywhere.
- The navigation bar containing links and dropdowns to all areas of the site.
- The footer, which contains some more helpful links as well as pertinent information.

The advertisement carousel is selectively displayed on appropriate non-admin pages. Right now, no user is signed in and as such certain administrative resources are not being displayed on the navigation bar. Let's fix that by moving onto our first task!

Task 1: Sign In/Create Account

1) Click on the user account button in the navbar.
2) Select create account option to become familiar with this option.
3) Return to the sign in page and authenticate using the admin credentials: admin@example.com with the password: Admin1.

Task 2: Access Admin Dashboard

1) Now that you are signed in, we can see that the admin dashboard button is revealed, click on it.
2) You are now at the admin dashboard, take a moment to scroll through and read each of the 5 utility categories.
3) Be aware that you can collapse and expand these categories to better view the array of options at hand.

Task 3: Course Management

Create

1) Create a course by selecting the appropriate tile and filling out the necessary details.

Retrieve

2) Notice that on from this page there is a button linking directly to the course list view.
3) Use that button or return to the admin dashboard and select the appropriate tile to enter the course list view.

Update

4) You'll notice that from here we can select a course from the list to edit it, or we can use the convenience buttons below the heading to go to the add course or inactive course list views.
5) Select a course now to begin editing.
6) Edit a course to have a new price and save changes.
7) For quicker access to editing your array of courses select the "Edit Courses" tile, which presents you with all available listings, in an editable state.

Delete

8) Edit a course to be deleted before accessing the inactive course list view, try restoring a course type, taking note of the fact that you can permanently delete a course here as well.

Task 4: Class Management

Create

1) Now that you are familiar with how to create a course lets try making a class based on this course category by selecting the create class tile and entering relevant information.

Retrieve

2) Return to the admin dashboard and within the "Class Management" section select the "View Classes" tile to retrieve the class list view.

Update

3) Similarly to the course view, we can click on a record here to make an edit, or we can select from one of the two convenience buttons below the page heading to create a new record or view the inactive class list view.
4) Select a class to edit and make and save a small change to the record.
5) Much like the course management section, we can return to the dashboard and select the "Edit Classes" tile to immediately view all available listings, in an editable state.

Delete

6) Select a class to be edited once again, but this time delete the class.
7) In the class list view, select the inactive button to view the record you rendered inactive.

Task 5: Vehicle Management

Now that we have established and reiterated our resource management processes, you can apply it to other areas of the site, such as the vehicle and user management sections , which follow an almost identical pattern. Try creating, editing, viewing, and deleting some of these resources independently.

Task 6: Site Administration – Advertising

Retrieve

1) In the admin dashboard scroll to the Analytics and advertising section.
2) Select the "Manage Adverts" tile and familiarize yourself with the page layout.
3) There are three sections to this page:
    a. Image upload: Upload new assets to the site from which active adverts can be selected.
    b. Active ads: This section displays the ads currently being displayed on the carousel with a delete button to render them inactive.
    c. Inactive ads: This section displays ads that are stored on the server but not currently being shown on the carousel.

Create

4) Try uploading an image and observe the default active status as well as dynamic carousel display.

Update/Delete

5) Try deleting an image to remove it from the carousel.
6) Try restoring an image to return it to the carousel.

Task 7: Site Administration – FAQ/Team Page

1) In the navbar, hover over the "About" section and select FAQ.
2) From here you can add new questions or edit and delete old questions.
3) Try it now!
4) Once you've got the hang of this page hover over the "About" tab once more, this time clicking on the "Meet the Team" option.
5) Click the add team member button to familiarize yourself briefly with this section.

Task 8: User/Role Management

1) Hover over the "Manage" navbar item and select the "User Management" option to view a listing of site users with management options attached.
2) Try showing a user.
3) Try editing a user's details.
4) Try deleting a user.
5) Now select the "Role Management" option from the navbar and interact with these roles similarly.
6) Click on the "Create Role" button to familiarize yourself with this, should you need to adjust roles and permissions.

Task 9: Customer Perspective

1) For our final task I'd like you to become familiar with the customer side of the application.
2) From the home page, hover of the "Courses" navbar item and select the "Course Overview" option to see a list of currently running courses.
3) From here a customer can click on a course to begin the registration process, similar to what an administrator would do to edit the instance.
4) From the registration screen (which is alternatively reached through selecting the Register option from the courses dropdown), a user can view current registrations, or select a course type and view available classes of the type.
5) In the available classes, try registering for a class instance before returning to the registrations page to view the scheduled instance.

Conclusion / Q&A

To conclude the test please select the sign out button to ensure administrator access is protected from non-authorized users. Thanks very much for participating in today's training session. If you have any final questions, please let us know now.

## 9.2. Training Materials

### 9.2.1. Training Manual

Due to its large size, the Training Manual will be supplied as a separate document.

### 9.2.2. Feedback Form

Below is a blank feedback from to be provided to all trainees to be filled out after the training session. This will allow us to accurately gauge how well the training session went.

"Open Road" Training Feedback Form

_____ being trained by SIT SOFTware on the use of the "Open Road" Information System.

Date of Training: _____

**Instructions**: Please indicate your level of agreement with the statements listed below in #1-9

| | Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree |
|---|---|---|---|---|---|
| 1. The objectives of the training were clearly defined. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 2. Participation and interaction were encouraged. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 3. The topics covered were relevant to the use of the system. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 4. The content was organized and easy to follow. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 5. The materials provided were useful. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 6. The training experience will be useful towards the use of the system. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 7. The training team was knowledgeable about the system. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 8. The training team was well prepared. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 9. The time allotted for training was sufficient. | ☐ | ☐ | ☐ | ☐ | ☐ |

| | Yes | No | Maybe |
|---|---|---|---|
| 10. Would you feel fully prepared to use the system after this training session? | ☐ | ☐ | ☐ |

*Figure 9.2.2-1 – Page 1 of the Feedback Form*

11. What was the most useful aspect of this training?

12. What aspects of the training could be improved?

13. Please share any other comments or concerns you may have with this training session.

*Figure 9.2.2-2 – Page 2 of the Feedback Form*

# 9.3. Training Session Report

## 9.3.1. Filled Feedback Form

Below is the feedback forms that were provided to and filled out by Sanjeev Prabhakar.

### "Open Road" Training Feedback Form

Sanjeev Prabhakar being trained by SIT SOFTware on the use of the "Open Road" Information System.

Date of Training: 03/28/2023

**Instructions**: Please indicate your level of agreement with the statements listed below in #1-9

| | Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree |
|---|---|---|---|---|---|
| 1. The objectives of the training were clearly defined. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 2. Participation and interaction were encouraged. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 3. The topics covered were relevant to the use of the system. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 4. The content was organized and easy to follow. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 5. The materials provided were useful. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 6. The training experience will be useful towards the use of the system. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 7. The training team was knowledgeable about the system. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 8. The training team was well prepared. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 9. The time allotted for training was sufficient. | ☒ | ☐ | ☐ | ☐ | ☐ |

| | Yes | No | Maybe |
|---|---|---|---|
| 10. Would you feel fully prepared to use the system after this training session? | ☒ | ☐ | ☐ |

*Figure 9.3.1-1 – First Page of the Filled Feedback Form*

**11. What was the most useful aspect of this training?**

I really like the training.It was really a good experience every aspect of the training was better than the other but if I talk about how you breakdown every task and guide the subject to complete that was the best and that I think is the most important work to do in order to conduct a successful training session.It gives me an idea what the project is about and I am now confident enough to use your software again.

**12. What aspects of the training could be improved?**

I really don't find any area need to be improved whether it's content - it's relevant and up to date , delivery- the way you deliver the training was brilliant. You have done an excellent job.

**13. Please share any other comments or concerns you may have with this training session.**

*Figure 9.3.1-2 – Second Page of the Filled Feedback Form*

### 9.3.2. Training Session

For our team training session, we were paired up with Sanjeev Prabhakar, formerly a member of the Code Benders, to train him on the "Open Road" System. As he is a CIT student, he required no training on how to use the necessary hardware or supporting software, which ensured that the training session went smoothly.

One of our team members, Josh, had prepared a script and training plan before hand, which ensured that all areas of the system were thoroughly explained and demonstrated. While Josh walked Sanjeev through the system, Sanjeev was the only one interacting with it, with minimal "hand-holding" from Josh. This demonstrated to us that our User Interface was self-explanatory, and our training was comprehensive. This will be especially important once it comes time to train our client, as he is less "tech-savvy" than Sanjeev.

### 9.3.3. Regarding the Feedback

Sanjeev provided very thorough feedback, assuring us that our training methods were detailed enough to train someone who would then be able to train other people on the use of our system. Sanjeev also pointed out that most of the processes performed through our system can be quite repetitive, creating a course is similar to creating a user, which is similar to creating a class, etc. This greatly helped in the training process, though, as it helped to cement the concepts. By the end of the training session, Sanjeev needed no direction from Josh, and was able to create, edit, and delete resources in a module of the system that he had not yet visited.

### 9.3.4. Changes to be Made

Based on our training session, and the feedback received from Sanjeev, we don't feel that there are many changes that must be made to our methods. The biggest improvement that could be added to our training would be the inclusion of a Training Manual, which was unfortunately not fully completed at the time of our training session with Sanjeev.

## 9.4. Conclusion

This milestone we took our completed system, and designed a training plan for it. We wrote a comprehensive training manual that can be used to train future users of the system and will also be provided to the client. Along with the training manual, we also devised a training script and feedback forms. The script will be used to train the client and other management members in the use of the system, and the feedback forms will provide our team with valuable information that we can then use to improve our training methods.

In order to be sure that everything that we had planned for training was effective, we trained another team on the use of our system, using the same materials and script that we have for the client. The "practice" training session went overwhelmingly well, and the feedback from the other team informed us that no major changes were needed to our training methods.

We have a tentative date scheduled to train our client, and possibly one other person in upper management. Using our newfound experience, and the valuable feedback from previous training, we have no doubt that we will be able to train one or two people within the Precision Riding Academy well enough that they will be able to fully train all other future users of the system.

# 10. Milestone Ten

Table of Contents

## 10.1. Implementation Plan

### 10.1.1. Data Entry

As per the project scope, the client will be responsible for inputting any company data associated with the new system. Such data includes:

- FAQ details (questions and answers)
- Meet the Team details
- Site Terms and Conditions
- Site Privacy Policy
- Advertisements
- Courses
- Scheduled Classes
- Instructor Accounts
- Etc.

SIT SOFTware has ensured that the processes associated with listed data is included in the accompanying training manual for the new system.

### 10.1.2. Implementation

Conversion Strategy

As the new system is web-based and does not require installation onto multiple platforms, it will be implemented as a whole-system, directly onto the VPS (virtual private server) provided by the web hosting service. SIT SOFTware will provide the URLs, data files, setup details, and installation tools (those not provided by the web hosting service). Installation of the system will include:

1. Securing of the VPS (usernames/passwords).
2. Installation of dependencies (system updates if required, Docker container service, and git).
3. Transfer and setup of the new system (cloning GitHub repository containing system source code and setup of production environment with required environment data not kept in source code for security concerns or platform uniqueness).
4. Initial testing of the system to ensure appropriate operational completeness.

Potential Issues

Any issues that may disrupt the system itself or the implementation thereof are as follows:

- Unforeseen conditions or misconfigurations in production platform during setup
- System bugs not encountered during coding and testing

Should any issues arise during implementation, SIT SOFTware will do their utmost to ensure they are resolved in a manner that impacts the new system as little as is possible before finalizing the implementation of the new system.

### 10.1.3. Schedule

The new system will be implemented according to the following schedule:

| Date | Time | Location |
|---|---|---|
| April 14th, 2023 | 12:00pm | Lethbridge College |

The client will be alerted upon completion of the implementation or receive regular updates should issues be encountered with a notice of completion once they are resolved and the system is operationally complete.

## 10.2. Implementation

### 10.2.1. Implementation Process

Configuration Details

| Platform/Software | Dependency | Notes |
|---|---|---|
| Platform Operating System | • Ubuntu Server (v22.04 or later)<br>• Docker Engine (v20.x or later)<br>• git (if not installed by default) | |
| Application | • Node.js (v18.x or later)<br>• NPM (v9.x or later)<br>• PHP (v8.1 or later)<br>• PHP Extensions<br>  ○ php-cli<br>  ○ php-ctype<br>  ○ php-curl<br>  ○ php-dom<br>  ○ php-fileinfo<br>  ○ php-filter<br>  ○ php-hash<br>  ○ php-mbstring<br>  ○ php-openssl<br>  ○ php-pcre<br>  ○ php-pdo<br>  ○ php-session<br>  ○ php-tokenizer<br>  ○ php-xml<br>• Composer (v2.5.x or later)<br>• Redis (latest)<br>• MariaDB (v10.5 or later)<br>• SMTP mail server<br>• PHPMyAdmin (v5.1 or later)<br>• Nginx HTTP Server (latest) | All application dependencies will be installed AFTER the Docker Container is up and running. |

Installation Instructions

1. Update OS platform and install system dependencies.

- `sudo apt update && sudo apt upgrade -y`
- `sudo apt-get install ca-certificates curl gnupg php-cli`
- `sudo apt install git`

2. Add Docker official apt repository and install Docker Engine and docker-compose.

- `sudo install -m 0755 -d /etc/apt/keyrings`
- `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg -dearmor -o /etc/apt/keyrings/docker.gpg`
- `sudo apt-get update && sudo apt-get upgrade -y`
- `sudo apt-get install docker-ce docker-ce-cli contanerd.io docker-compose-plugin docker-compose`

3. Clone git repository to transfer source files to local server platform.

- `git clone https://github.com/SIT-SOFTware/open-road.git /var/www/open-road`
- `sudo chown -R <username>:docker /var/www/openroad`

4. Upload required configuration files to project directory on server platform. (Provided by SIT SOFTware)
    a. Required config files and their final directory path:
        i. ./docker-compose.yaml
        ii. ./app.dockerfile
        iii. ./mariadb/my.cnf
        iv. ./redis/config/redis.conf
        v. ./redis/config/users.acl
5. Copy example.env file to make a .env

- `cp /var/www/openroad/.example/env /var/www/openroad.env`
- `cd /var/www/openroad`

6. Start docker container, install npm and composer dependencies and generate app key.

- `sudo docker exec -it app composer install`
- `sudo docker exec -it app npm install`
- `sudo docker exec -it app php artisan key:generate`
- `sudo docker exec -it app php artisan migrate`
- `sudo docker exec -it php artisan db:seed`

### 10.2.2. Installation/Configuration Instructions

## Project "Open Road": Dev Environment Setup Manual

*FRONT NOTE*

ALL dependencies should ALREADY be included in the composer.json currently within the project directory in the Open Road GitHub repository. DO NOT install any other dependencies or packages unless you have been directed to do so!

On a Fresh Ubuntu 22.04 Install (through WSL or a linux CLI), run the following commands:

- `sudo apt update && sudo apt upgrade -y`
    - This will install the latest versions of everything that is currently being used and make sure that anything that is installed in the future is updated to the latest version.
- `sudo apt-get install openssl php8.1-cli php8.1-common php8.1-curl php8.1-mbstring php8.1-mysql php8.1-xml php8.1-bcmath php8.1-zip unzip`
    - This will install PHP and the dependencies it needs, plus some extras

Installing Composer

Composer is a PHP extension manager that installs all dependencies needed for a project and can update them with a single command. The list of dependencies is contained in the composer.lock and composer.json files, which is a part of the "Open Road" source code.

- `php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"`
  - This command gets the latest version of the composer installer
- `php -r "if (hash_file('sha384', 'composer-setup.php') === '55ce33d7678c5a611085589f1f3ddf8b3c52d662cd01d4ba75c0ee0459970c2200a51f492d557530c71c15d8dba01eae') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"`
  - This command compares the official hash for composer with the one that is included in the installer to verify that it is legitimate. It should return "Installer Verified" if the hashes match.
- `php composer-setup.php`
  - This runs the composer setup file and installs composer
- `php -r "unlink('composer-setup.php');"`
  - This removes the composer setup file from the local filesystem
- `sudo mv composer.phar /usr/local/bin/composer`
  - This moves the composer.phar file so we can access composer commands and operations

Installing Docker

Docker is what the application actually runs on. All of our services (nginx webserver, php myAdmin management server, mailpit, redis, MariaDB, etc.) are installed via their own docker images and are located in a single container along with the application to allow the separation of these services and removes the need to individually install them on the server itself.

- `sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-compose`
  - This command installs docker as well as docker-compose along with some other dependencies that it requires

Cloning the Repository Onto the Server

Using Git, we can clone the repository that contains the source code for the "Open Road" application onto the filesystem so that it can be developed on.

- `Git clone https://github.com/[Your-UserName-Here]/[Repo-Name-Here].git <target-directory (optional)>`
    - This clones the repository into the folder you are currently in, or the specified target directory
    - You will also need to copy some extra files that are not in the repo, which will be provided along with instructions on how to copy them.

Installing NVM

NVM (short for Node Version Manager) is an all-in-one manager for both nodejs and npm (Node Package Manager). This will ensure that both node and npm are compatible and on the same version.

- `curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.2/install.sh | bash`
    - This clones the repository to your local filesystem
- `export NVM_DIR="$([ -z "${XDG_CONFIG_HOME-}" ] && printf %s "${HOME}/.nvm" || printf %s "${XDG_CONFIG_HOME}/nvm")" [ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"`
    - This loads NVM so it can be used
- `nvm install v18.14.1`
    - Installs nodejs v18.14.1 and its corresponding npm distro
    - If this command doesn't work, try `nvm install 18.14.1`

Project Directory Commands

Now that we have most of our applications installed globally, there are some things we need to do within the project directory we cloned previously to finalize our setup

- `cd <project-directory>`
- `composer install`
    - This will use the composer.json and composer.lock files mentioned earlier to install all dependencies for the project.
- `npm install`
    - This is like composer install but for nodejs. It will use similar package.json and package-lock.json files to install any needed node dependencies
- `php artisan key:generate`
    - This generates a unique application key and puts in into the .env file witch the application bases all of its encryption on

Configuring Git for Version Control

If you are developing this application, you will need to tell git who you are so it can push to whichever repo you are using.

- `git config --global user.name "GitHub-Username"`
- `git config --global user.email git-email@email.com`
    - These commands configure your username and email so you can push to a GitHub repository

### 10.2.3. Hosting Account Details

As the client has not yet chosen a suitable host for the new system, there are currently no Hosting Account Details to be provided.

## 10.3. Training

### 10.3.1. Training Report

Introduction

Milestone Ten of the Open Road Project involved primarily the implementation of the developed software and the training of the employees in the effective use of their new tool. This report lays out the profiles of the selected trainees, details the training process that was carried out and discusses its outcomes, bot in terms of successes and failures.

Trainee Profiles

As a result of the incomplete nature of the Open Road web application, the development team chose to train members of the general public, as opposed to Precision Riding Academy employees. We felt this would allow us to better gauge where to focus our efforts with the application, so that it might be possible to deliver a better end product, given further development time. Two trainees were selected for the training process: Brooke Hickman and Shylah Towpich. Both users possess a slightly above average understanding of computer technology and the web, are individuals who have basic experience using computers systems in a professional environment and are both individuals who prioritise ease of use when it comes to user experience.

Training Goals

The development team felt confident that the training plan used in the trial sessions carried out during the previous milestone was perfect for the execution of this round of training, and as such the same script was used. As with the trial training sessions, our primary goal was to successfully execute a one on one training sessions with a user that would take up to 45 minutes, maximizing the sessions potential influence by iterating the importance of the users role in the bigger picture of the tools implementation and ensuring that the trained users were comfortable navigating all aspects of the application.

Training Outcomes

The training proved successful in several key areas. Firstly, both users had a pleasant experience during the training and felt that the atmosphere was comfortable and conducive to learning, while still being relatively time efficient and informative. Secondly, the trainees felt mostly confident in their ability to use the system with one trainee agreeing that they would feel comfortable teaching another user to navigate the system. Lastly, the training plan proved once again to be a successful guideline for developing an understanding of the Open Road web application for users from a variety of technical backgrounds.

The training was, however, also unsuccessful in some respects. Multiple errors were encountered that acted as roadblocks, preventing the team from carrying out certain aspects of the training plan either entirely or until appropriate fixes were applied. Setup time was also too long compared to the amount of time required to deliver the actual training. Finally, the pace at which the training was delivered was too fast for users of a lower technical ability than our initial instance of training, leading to some details being left out or glossed over by the instructor of the session as well as the users not being able to fully internalize certain concepts. Both of these factors prohibited our ability to facilitate a better understanding of the system.

Benefits of the Training

This second round of training provided tangible value to the team as we move forward. It established the efficacy of many of our current backend processes, while highlighting the need for better error handling to stop small hiccups from causing large problems, primarily in terms of data and authentication/authorization, as well as emphasizing the need for better testing in certain areas of the code base. It also showed which aspects of the interface are working while pointing out areas for improvement in this regard. The training process also provided some insight into how to improve the approach to system training as the systems final implementation steps are carried out.

Conclusion

The implementation and training process of the Project Open Road project for Precision Riding Academy was successful in many areas but had some shortcomings in other regards. Both our successes and our shortcomings have proven valuably insightful. Certain areas of success and the necessary improvements that need to be made demonstrate the importance of conducting training exercises to ensure the efficacy of a system, both as it is being implemented and iterated upon as well as throughout development. The experiences gained throughout the training process will be beneficial throughout future projects in improving our approach to system design and ensuring that training initiatives are well-focused, time-efficient, informative, and impactful for the end user.

## 10.3.2. Training Feedback Forms

# "Open Road" Training Feedback Form

Brooke Hickman being trained by SIT SOFTware on the use of the "Open Road" Information System.

Date of Training: April 8 2023

**Instructions:** Please indicate your level of agreement with the statements listed below in #1-9

| | Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree |
|---|---|---|---|---|---|
| 1. The objectives of the training were clearly defined. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 2. Participation and interaction were encouraged. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 3. The topics covered were relevant to the use of the system. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 4. The content was organized and easy to follow. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 5. The materials provided were useful. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 6. The training experience will be useful towards the use of the system. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 7. The training team was knowledgeable about the system. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 8. The training team was well prepared. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 9. The time allotted for training was sufficient. | ☐ | ☒ | ☐ | ☐ | ☐ |

| | Yes | No | Maybe |
|---|---|---|---|
| 10. Would you feel fully prepared to use the system after this training session? | ☒ | ☐ | ☐ |

11. What was the most useful aspect of this training?

It was helpful to get the customer and staff perspective to understand everything better.

12. What aspects of the training could be improved?

Everything was organized and ready. The team knew their website very well. Maybe add more students and courses so that it feels more realistic as you're training.

13. Please share any other comments or concerns you may have with this training session.

I would have liked to have taken more time practicing in each section, instead of doing a little bit of each section.

# "Open Road" Training Feedback Form

Shylah Towpich being trained by SIT SOFTware on the use of the "Open Road" Information System.

Date of Training: April 7 2023

**Instructions:** Please indicate your level of agreement with the statements listed below in #1-9

| | Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree |
|---|---|---|---|---|---|
| 1. The objectives of the training were clearly defined. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 2. Participation and interaction were encouraged. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 3. The topics covered were relevant to the use of the system. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 4. The content was organized and easy to follow. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 5. The materials provided were useful. | ☐ | ☒ | ☐ | ☐ | ☐ |
| 6. The training experience will be useful towards the use of the system. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 7. The training team was knowledgeable about the system. | ☒ | ☐ | ☐ | ☐ | ☐ |
| 8. The training team was well prepared. | ☐ | ☒ | ☐ | ☐ | ☐ |
| 9. The time allotted for training was sufficient. | ☒ | ☐ | ☐ | ☐ | ☐ |

| | Yes | No | Maybe |
|---|---|---|---|
| 10. Would you feel fully prepared to use the system after this training session? | ☐ | ☐ | ☒ |

11. What was the most useful aspect of this training?

| |
|---|
| The software was user friendly and easy to navigate. It was helpful learning where to go on the website for certain tasks. |

12. What aspects of the training could be improved?

| |
|---|
| Software was broken in some parts of the website so some of the links weren't available to be trained on. |

13. Please share any other comments or concerns you may have with this training session.

| |
|---|
| The training team was knowledgeable with the program as well as friendly and patient. They were also apologetic for the aspects of the program that didn't work. It would have been more beneficial to have the whole system working, before learning it to have the full experience. |

## 10.4. Conclusion

This milestone we took our completed system and designed the implementation plan for it. We completely uninstalled our entire development environment, and went through it step by step to ensure the accuracy of our implementation plan. This process enabled us to find and fix some additional problems that will no longer be an issue.

We then took our implemented system and trained some more individuals on our system, using our previously submitted Training Plans. This helped us to improve our Training Plans, and provided some additional testing and debugging opportunities for our implemented system.

While the system has not been implemented for the client on their chosen host, we are confident that with our thorough planning, proper implementation for the Precision Riding Academy will be easy and seamless.

# 11. Final Conclusion

Throughout the course of this program, our group has fully committed to the Systems Development Life Cycle. We were thorough in all of our planning and documentation, fully ensuring the success of the project.

We initially started with forming our team. We found our six members, decided on a team name and branding. We allocated roles, understanding that as the project progressed, our roles within the group would likely change and evolve. We then chose our client, the Precision Riding Academy. The champion for the project is Bruce Streibel, the department head of the Academy.

We then performed a feasibility analysis, looking at the Technical, Economic, and Organizational feasibilities of the project. We concluded that the project was fully feasible.

In Milestone Two, we looked into the project requirements and scope. We met with our client, and made a list of all of the features that he would want or like to have in the system. We then broke that list down and separated our the list into three different scope lists: Required, Wish List, and Out of Scope.

We looked at the System Requirements, the Community of System Users, and the Business and System Policies that would be in place. We investigated the physical site to be sure that all requirements of the scope could be met.

We then planned out our project timeline, generating a rough overview of how long each phase of the Development Life Cycle should take us, and setting dates for our Milestones.

In Milestone Three we began generating Use Cases for the basic functions of both the currently implemented system and the proposed system. Along with these use cases, we generated Data Flow Diagrams. Both the Use Cases and DFDs helped us plan how the system would be used, and how data would flow amongst the system.

This became useful in Milestone Four, when we began to plan our database. We started by analysing the different Business Components, and their roles and purposes. We then generated a list of business rules. Using these, we developed our Entity Relationship Diagrams and initial Data Dictionary, both of which were used to design the base of our database.

In Milestone Five we performed another Feasibility Analysis, using our newly generated knowledge of the proposed system. We generated an Alternative Matrix, so that the client could be made fully aware of other possible solutions to their business problems. We made a System Diagram for our proposed system, along with a Budget and a more in-depth timeline. We then presented all of this information to the client, along with the recommendation that they use our proposed system, which they decided to do.

In Milestone Six we moved on to the physical data design, creating Physical Entity Relationship Diagrams, and a more detailed Data Dictionary. We also developed our Data Backup and Data Archiving plans, to ensure that the client's data never gets lost or corrupted.

In Milestone Seven we began the Process Design for the system, starting with physical Data Flow Diagrams. We then started to design the Program Structure and Specifications, using Structure Charts and Specification Sheets, which detailed all processes in the system, and how they would be executed. We also looked at the physical architecture of the system, planning what hardware and software would be needed for the final version of the system.

We also designed the User Interface for the system, using our process and good UI design standards to design the layout and look of the finished system. We designed all inputs and outputs, carefully planning the User Experience.

In Milestone Eight we took all of the planning and documentation that we had accumulated, and began programming the system. While we were programming, we also designed our Testing Plans, and tested each module of the system as it was coded. Once we had enough of a system to use, we had another team review our code. This helped us to ensure that we were following proper code standards in regards to documentation and function modularity.

In Milestone Nine we developed our Training Plan. We looked at the subject of the training, and decided on the most appropriate setting and time for training. We also developed a training script to ensure that all training sessions followed the same pattern and that no section of the system was missed. We also developed a Training Manual to accompany both the system and the training sessions, along with Feedback Forms for trainees to fill out. These forms were used to improve both the system and our training methods.

We then had the opportunity to train another team on the use of our system, and they provided us with invaluable feedback to improve ourselves and the system.

In Milestone Ten we developed our implementation plan. We decided upon the environment that the system would need to be deployed in, and wrote detailed instructions for setting up that environment. We then wrote a detailed document describing the proper implementation of the system in the newly created environment. We then tested our implemented system by training more people on it. This not only helped us ensure that the system was functional, but also helped us perfect our training methods.

While the system is not yet implemented on the clients host server, we are fully confident that with all of the planning and documentation that has been made available to them, it will be a seamless process.

While there may have been some struggles along the way, we feel that this project was a valuable learning experience for all team members, and one that we will use to continue to grow and better ourselves as people and developers.

# 12. Cumulative Lessons Learned

## 12.1. Milestone Ten

| Lessons Learned Document | | | | | |
|---|---|---|---|---|---|
| Milestone #10 | | | | | |
| **Project Name:** | Open Road | | **Project Manager:** | Nathaniel Meyer | |
| **Date Identified:** | **Entered By:** | **+/-** | **Lesson** | **Type of Lesson** | **Impact** |
| 2023-04-14 | Josh | + | Spaghetti is netiher spag nor is it hetti. It actually gets this name from its shape. | Professional | I'll be more aware of the fact that spaghetti is the plural form of the Italian word 'spaghetto', which comes from the word 'spago', meaning cord, string or twine. Cool! |
| 2023-04-14 | Nathaniel | - | Don't get sick on the last week of class | Professional | As finals approach in the future, I will refrain from leaving the house or making contact with another human being to avoid sickness |
| 2023-04-14 | Tim | + | Resolving problems that arise during implementation takes time too. | Professional | Ensure that during you planning, you remember to account for time that may be spent resolving any issues that arise during implementation. |
| 2023-04-13 | Shealyn | + | Getting fustrated is o.k. it just means that you are learning and resreaching new topices and working though the thought process on how it works. | Personal | I have struggled a lot thought these coruse but just cause I stuggled doesn't mean i learnd nothing |
| 2023-04-12 | Bryan | + | College was mostly fun | Personal | I enjoyed my time at school for the most part and am grateful for the wonderful Teachers |
| 2023-04-05 | Claire | + | A deficiency in one area of skills does not mean that one cannot meaningfully contribute to the success of the project. | Professional | I have learned that I have skills in the areas of documentation and record keeping. I will carry these skills forward with me. |

## 12.2. Milestone Nine

| Lessons Learned Document | | | | | |
|---|---|---|---|---|---|
| Milestone #9 | | | | | |
| **Project Name:** | Open Road | | **Project Manager:** | Nathaniel Meyer | |
| **Date Identified:** | **Entered By:** | **+/-** | **Lesson** | **Type of Lesson** | **Impact** |
| 2023-03-30 | Bryan | + | Training someone who already knows the system isn't a learning experience | Project | When training people look for someone who isnt at all familiar with the system. You wont get good data otherwise |
| 2023-03-30 | Josh | + | Asking for advice is a sign of security and strength rather than weakness. | Personal | In the future I'll try and better understand when to put aside my pride and ask my peers and mentors for advice when I'm at a loss for my next move. |
| 2023-03-30 | Nathaniel | - | Communication is key | Project | A lot has been going on that could have been prevented If we communicated more. In future applications, I will ensure proper communication |
| 2023-03-30 | Shealyn | + | some times it's o.k. to study at the college instead of at home and if you are going to study at home make sure the distractions are kept to a minimum | Personal | I have been staying at home more then I would like this week cause of sickness and even thought I can remote in and learn on the side a messy house and loud naugbors make it hard to focus so. For next time if I have not time I will study at the college commons |
| 2023-03-29 | Tim | + | Time spent learning should be taken into account when determining project goals and tasks. | Professional | Expecting any required learning to be done in an individuals personal time can quickly lead to people being overworked and the project becoming roadblocked. Time spent learning new tools and processes should be included when determining how long a project may take. |
| 2023-03-28 | Claire | - | Personal and Professional disputes will cause issues that must be overcome to complete a project. | Personal | Leaving or silencing official channels of communication is not the proper way to deal with professional or personal disputes with team members. I cannot allow such disputes to impact my performance within the group. |

## 12.3. Milestone Eight

| Lessons Learned Document | | | | | |
|---|---|---|---|---|---|
| Milestone #8 | | | | | |
| **Project Name:** | Open Road | | **Project Manager:** | Nathaniel Meyer | |
| **Date Identified:** | **Entered By:** | **+/-** | **Lesson** | **Type of Lesson** | **Impact** |
| 2023-03-20 | Nathaniel Meyer | - | Being too lenient of a leader can cause confusion and impede work flow | Professional | During the beginning of this milestone, I was too lenient when it came to leading the team. It caused confusion among the group and negatively impacted our workflow which caused us to fall behind |
| 2023-03-20 | Shealyn Cossette | - | When nothing is assigned that doesn't mean you can be complacent. If you are given nothing take something. | professional | There was a lack of structure when it came to coding everyone knew what needed to be done but dividing up of the tasks was handled poorly. Because of this I was tasked with nothing, I wasn't put anywher, I wasn't told to focus on anything leaving me complacent. Now I know when this happens to take what I want to do and continue with it. |
| 2023-03-20 | Josh Hickman | - | It is easy for a group to spend too much time talking, planning and thinking about a problem without generating anything substantial or making progress. | professional | In the future I will prioritize speaking my mind and ensuring the success of the groups primary objective over my desire to maintain a static free social climate. |
| 2023-03-20 | Tim Streibel | - | New tools are cool, but not always the best choice. | Professional | It can be fun to discover some new piece of tech that makes everything seem easy at first, but once you start to dig into you being to realize just how big the learning curve is. In the future I will do better at assessing what my current skills are and finding tools that align with those skills. |
| 2023-03-18 | Claire Fleckney | - | I will end up working with new and unfamiliar tech stacks. | Professional | I'm going to have to get used to doing my own research and teaching myself how to use new and unfamiliar frameworks. |
| 2023-03-11 | Bryan Towpich | - | It is wise to listen to my Gut. It tends to be right | Personal | There have been multiple times during this project where I should of listened to my Gut. I didn't and now I'm stuck in a situation I'd really rather not be. I wont be making this mistake again. |

## 12.4. Milestone Seven

| Lessons Learned Document | | | | | |
|---|---|---|---|---|---|
| Milestone #7 | | | | | |
| **Project Name:** | Open Road | | **Project Manager:** | Nathaniel Meyer | |
| **Date Identified:** | **Entered By:** | **+/-** | **Lesson** | **Type of Lesson** | **Impact** |
| 2023-02-15 | Bryan Towpich | - | Time management skills are an important part of any project. Lacking them puts undue stress on the entire team | Project | Having a calender with all tasks helps ensure things are going great and will help ensure that tasks are completed on time |
| 2023-02-15 | Claire Fleckney | - | An large task doesn't always have to be overwhelming to the point of burnout | Project | Breaking a large task down in to smaller ones makes projects much more manageable |
| 2023-02-15 | Josh Hickman | + | While extensive planning can feel tedious it has definite benefits | Design Planning | I'm more willing to do extensive preparation work after seeing the benefits. |
| 2023-02-15 | Nathaniel Meyer | + | Leading a team takes a lot more effort than I thought | Project | This leading experience will prepare me for future milestones and my career |
| 2023-02-10 | Shealyn Cossette | + | Feeling Paralized in one's own work means that you want it to be better than what you think you can provide, the only thing you can do is to take it and do it! Don't obsess over the details and the small matters. | Field Work | People are willing to help you, you just need to ask. Stop worrying whether they think less of you get the project done before its to late. |
| 2023-02-02 | Tim Streibel | + | Personal stakes in a project will always affect how you lead | Project | It doesn't matter who you are, it is unlikely that you can appropriately separate personal stakes and professional responsibilities. Letting someone else lead is often the better choice. |

## 12.5. Milestone Six

| Lessons Learned Document | | | | | |
|---|---|---|---|---|---|
| Milestone #6 | | | | | |
| **Project Name:** | Open Road | | **Project Manager:** | Tim Streibel | |
| **Date Identified:** | **Entered By:** | **+/-** | **Lesson** | **Type of Lesson** | **Impact** |
| 2023-01-28 | Nathaniel | + | Arriving to class on time may prove useful for the future | Academic | I now know that COBOL is the best programming language |
| 2023-01-28 | Bryan | + | We have a very capable group of students. despite illness we can still do everything | Academic | I am way less stressed out about what happens if illness cripples several members |
| 2023-01-28 | Josh | - | I should better prioritize tasks so that I'm not rushing to do certain tasks while less important work is completed | Workload | I will adopt a broader perspective when planning my time spent working. |
| 2023-01-23 | Shealyn | - | Double check your work and read the instuctions. Making stupid misktakes cost you little at a time. | Academic | Quiz one of linux! |
| 2023-01-20 | Claire | + | Attend class to obtain additional information that is important for good grades. | Academic | I will remember everything that I know about "computers and stuff" |

## 12.6. Milestone Five

| Lessons Learned Document | | | | | |
|---|---|---|---|---|---|
| Milestone #5 | | | | | |
| **Project Name:** | Open Road | | **Project Manager:** | Tim Streibel | |
| **Date Identified:** | **Entered By:** | **+/-** | **Lesson** | **Type of Lesson** | **Impact** |
| 2022-12-08 | Bryan | - | Cancer Sucks | Personal | I will learn to rely on my teammates to support me in times of personal troubles. |
| 2022-12-07 | Josh | - | Disregarding personal health in favour of productivity has diminishing returns. | Personal | I will place more personal significance on my physical health and mental wellbeing |
| 2022-12-07 | Shealyn | + | leaving just a little bit early isn't a bad thing | Personal | I will admit being on time is nice leaving a little bit of breathing room for mistakes to happen is useful to make any day a less stressful day. |
| 2022-12-07 | Tim | - | It's easy to come up with excuses. | Personal | I will try harder to follow up on my commitments as a team member when it comes to working on an assigned task with another team member. |
| 2022-12-05 | Nathaniel | - | Be on time more often | Project | While poor time management is a common side effect of ADHD, I need to work harder to stop myself from holding everyone up when it comes to meetings and work periods |
| 2022-12-02 | Claire | + | Branching out from my comfort zone can be beneficial both to myself and the group | Project | I will make more of an effort to learn new and unfamiliar programs and processes. |

## 12.7. Milestone Four

| Lessons Learned Document | | | | | |
|---|---|---|---|---|---|
| Milestone #4 | | | | | |
| **Project Name:** | Open Road | | **Project Manager:** | Tim Streibel | |
| **Date Identified:** | **Entered By:** | **+/-** | **Lesson** | **Type of Lesson** | **Impact** |
| 2022-11-20 | Josh | - | I should take more notes during meetings and brainstorming sessions | Personal | Discussion often contain too much for one person to remember accurately, and details get repeated or lost without descriptive notes. |
| 2022-11-20 | Nathaniel | + | Asking for a second opinion is always a good idea | Personal/ Project | Inquiring with others about a topic I may be stuck on can provide new insight and will likely end up improving results overall |
| 2022-11-18 | Bryan | - | Ensure slide show graphics are appropriate when using a projector before presentation. | Project | I will not use little red lines on black backgrounds anymore. It be better to consistently use white backgrounds for large images. |
| 2022-11-18 | Shealyn | + | Focuse on only one coding Languge at a time even if the other languages is being learned on your own time. It is to easy to confuse information with other languages you are learning. | Coding lesson | Putting my studies over my intrests are important, so making a wish list of things I want to learn for later will help me grow my intrests and not interfere with my studies. |
| 2022-11-14 | Tim | + | When team members are discussing suggested design, ensure that all parties completely understand the perspectives presented before comparing them. | Project | It is hard to have a constructive discussion when those involved don't understand what the other party is trying to say. |
| 2022-11-09 | Claire | - | Don't let side projects distract from current SAaD due dates | Project | I will try and set other, less urgent, projects on the back-burner, so that my assigned work for Open Road isn't negatively impacted |

## 12.8. Milestone Three

| Lessons Learned Document | | | | | |
|---|---|---|---|---|---|
| Milestone #3 | | | | | |
| **Project Name:** | Open Road | | **Project Manager:** | Tim Streibel | |
| **Date Identified:** | **Entered By:** | **+/-** | **Lesson** | **Type of Lesson** | **Impact** |
| 2022-10-30 | Shealyn | - | Managing a better sleep schdule! Even though I work till 11pm that doesn't mean I can't get a good 6 hours in. | Personal | When coming to class at 8am I will work out before hand to be ready to learn and listen. But I will also take the days where I don't get up eary to sleep. |
| 2022-10-29 | Bryan | + | I am able to rely on my team for help when I'm in no condition to work. | Personal | I will make sure to give everyone else the same courtesy they gave me this past week in the future |
| 2022-10-29 | Josh | - | Checking emails and messages at scheduled intervals reduces stress & maintains proffessionalism | Personal | I will schedule a regular time each work day to check and reply to direct/group messages and emails. |
| 2022-10-28 | Nathaniel | - | Don't snooze my alarm on presentation day | Personal | I had significantly less time to prepare for the presentation and if it continues, my performance could suffer as a result |
| 2022-10-27 | Claire | - | Team members (including myself) are capable of working independently on parts to create a finished whole. | Project | I will work on being better at working independently, without needing input from the team on how some things should be done. |
| 2022-10-27 | Tim | - | Focusing on semantics can hinder understanding and reduce productivity | Personal | I will try to not to get so stuck on the way something is worded, and focus more on the process behind it. |

## 12.9. Milestone Two

| Lessons Learned Document | | | | | |
|---|---|---|---|---|---|
| Milestone #2 | | | | | |
| **Project Name:** | Open Road | | **Project Manager:** | Tim Streibel | |
| **Date Identified:** | **Entered By:** | **+/-** | **Lesson** | **Type of Lesson** | **Impact** |
| 2022-10-17 | Josh Hickman | + | Important ideas and concepts are often shared through casual conversations. | Project | I will strive to more actively participate in social activities involving members of my team. |
| 2022-10-17 | Shealyn | - | Managing Stress Provides better learning abiltys and better Product | Personal | I will manage my stress level's first before dealing with anything else. This way my work will be better and my end Product isn't rushed. |
| 2022-10-14 | Bryan Towpich | + | I need to be confident in my own abilities. | Personal | I need to be confident in my abilities. Being Anxious only adds stress to myself and work to others |
| 2022-10-07 | Tim Streibel | + | Learning and understanding will be done while completing tasks | Project | I need to remember that we are ALL learning as we participate in this project. Time spent learning should be considered in the Porject Plan. |
| 2022-10-03 | Claire Fleckney | + | Faking confidence can be just as effective as having actual confidence. | Personal | I will do my best to not let public speaking cause me so much stress in the future. |
| 2022-10-03 | Nathaniel Meyer | - | Plan farther ahead for presentations | Personal | I will make sure to create notes and refine them a few days before the presentation |

# Appendix A – Team Charter

# SIT SOFTware Team Charter

## 1. SIT SOFTware Team Objectives

### 1.1. Purpose

The purpose of the SIT SOFTware team is to provide a completed and functional information system as part of the Lethbridge College CIT Systems Analysis and Design capstone project. SIT SOFTware will carry out the following specific activities:

- Create an Information System solution through implementing the SDLC processes
- Applying technical knowledge and professionalism in a simulated IT environment
- Following the Lessons Learned procedures
- Presenting project progress and results
- Submitting required course work as specified by course syllabus

### 1.2. SIT SOFTware: Scope of Relevance

The SIT SOFTware development team is responsible for:

- Selecting a client with an appropriate business need from the list provided by college instructor.
- Apply Business Process Management to identify opportunities for automating, improving, or redesigning business processes.
- Designing and Building an Information System solution.
- Implementing the Information System solution including support documentation and user training.
- Recognizing and reporting learning opportunities, both team related and individual.
- Fostering a constructive team environment to encourage learning, growth, and success.
- Communicating with chosen client through progress reports, meetings, and other channels as necessary.

# 2. Team Membership and Roles

Below is a list of the team members that make up the SIT SOFTware student development team. Each team members role and contact information is also included in this list.

## 2.1. SIT SOFTware Members

| Team Member Name | Roles/Responsibilities | Contact Information |
|---|---|---|
| Timothy Streibel | Project Manager | timothy.streibel@lethbridgecollege.ca<br>cell (403) 394-5787 |
| Nathaniel Meyer | System Analyst | nathaniel.meyer@lethbridgecollege.ca<br>cell (403) 894-8386 |
| Bryan Towpich | System Analyst | bryan.towpich@lethbridgecollege.ca<br>cell (403) 360-7344 |
| Claire Fleckney | Business Analyst | claire.e.poulsen@gmail.com<br>cell (403) 929-1237 |
| Josh Hickman | Infrastructure Analyst | josh.hickman@lethbridgecollege.ca<br>cell (403) 894-6955 |
| Shealyn Cossette | Change Management Analyst | shealyn.cossette@lethbridgecollege.ca<br>cell (403) 635-9978 |

## 2.2. Additional Members

Unless directed by the S.A.D. instructor and unanimously agreed upon by the team, no additional members will be added to the group during its operation time.

## 2.3. Additional Attendees

Additional attendees may include:

- Timothy Frantz (Lethbridge College Instructor)

# 3. Team Ground Rules

## 3.1. Meetings

- The team shall meet at least twice per week on Tuesdays and Thursdays. Additional meetings may be scheduled as circumstances dictate. Meetings may also be rescheduled as needed provided the team unanimously agrees.
- All team meetings taking place on Tuesdays will be general progress/update meetings.
- Team meetings taking place on Thursdays can be designated as either general progress/update meetings or "specialized" meetings where major decisions about project work must be discussed/decided upon.
- Each regularly scheduled meeting will be accompanied by a Team Meeting Agenda, which will be circulated 24 hours prior to a general meeting, or 48 hours prior to a specialized meeting. The agenda will state the order of business for each meeting.

## 3.2. Decision Making

- All major project decisions will decide by majority vote, with the appropriate section lead holding the tie breaking vote.
- In the case of both System Analysts being in opposition, the Project Manager holds the tie breaking vote.
- All pertinent information regarding a major decision **must** be shared to **all** team members **before** a vote can take place.

## 3.3. Communications

- Discord
  - All project specific communication should take place ONLY in the specified SIT SOFTware chat/voice channels.
  - Messages posted in the SIT SOFTware channel should receive a response from the person(s) associated with the message within 4 hours of the time the message is posted during regular hours (specified as 8:00am to 9:00pm).
  - Messages posted outside of normal hours should be responded to on the next regular business day.
  - If a posted message cannot be replied to with the requested information, the recipient(s) should at the very least acknowledge the message to ensure the sender knows it has been received.

- Email
  - Team members are required to check their email AT LEAST twice per regular business day.
  - Any email received should be responded to within 4 hours of the time the email is received during regular business hours (specified as 8:00am to 9:00pm).
  - Emails sent outside of regular business hours should be responded to on the next regular business day.
  - If an email cannot be replied to with the requested information, the recipient(s) should at the very least acknowledge the email to ensure the sender knows it has been received.
- Weekends
  - Team members are not required to respond to messages on the weekends unless other arrangements have been agreed upon beforehand.

## 3.4. Team & Personal Boundaries

Properly established boundaries encourage a safe and healthy workplace. Whether they be physical, emotional, or mental boundaries, setting goals and limits can help raise productivity, lower work-related stress, maintain reasonable workloads, and reduce the risk of burnout. As such, the following section is dedicated to boundaries that have been set for the team as well as boundaries that have been set by individuals which each team member will respect. They are not to be regarded as rules to take away, but guidelines that help ensure we have the space we need to do whatever it is we need to do.

- Team Boundaries
  - Non-School related activities – No S.A.D. talk during activities that are not related to S.A.D. It is important to have times where we can just have fun together without focusing on S.A.D.
  - Conversation Topics – Conversation topics relating to politics, religion, and/or personal beliefs should not be held during official SIT SOFTware work/meeting times.
  - Family related business and/or emergencies takes priority over ALL S.A.D. responsibilities. The team members collectively agree to provide support for the affected individual(s) regarding S.A.D. matters in as much as they are able.

- Individual Boundaries
    - Claire Fleckney
        - While my Lethbridge College email address is acceptable for use within the team for software tool integration purposes, I would like the client to only have access to my claire.e.poulsen@gmail.com email address.
    - Timothy Streibel
        - As I have a personal connection with the client and am involved with Precision Riding Academy as an assistant instructor, I will not talk about topics related to project progress with the client outside of official meetings or unless requested by the team.

# 4. Leadership

## 4.1. Project Manager

As voted upon by unanimous agreement, Timothy Streibel is designated as the project manager for the SIT SOFTware development team.

## 4.2. Roles and Responsibilities

The project manager will have the following responsibilities:

- Project management and timelines
- Conflict resolution and disciplinary steps are enforce as required
- Course related team documentation finalization and submission
- Management of online document repository

# 5. Conflict Resolution Rules

## 5.1. Project Manager Responsibilities

This section dictates the specific responsibilities the project manager has regarding team conflict resolution and are detailed as follows:

- The project manager will work to uncover the issues and motivations that are driving the conflict.
- The project manager will communicate with each member individually to ensure that they understand the individual motivations behind the conflict.
- Once the project manager has spoken with each team member, they will schedule a meeting to discuss the impact the conflict could have on team objectives (see step 2 below).
- The project manager agrees to observe the privacy of all members involved in a conflict.

## 5.2. Team Rules of Conflict Resolution

The team members recognize that conflict may arise during the time that the team is operating. If a conflict within the team arises, any team member is encouraged to report the conflict using the report template available through the GitHub repository, to the project manager. After the form has been filled with the appropriate information it should be EMAILED DIRECTLY to the project manager. All submitted reports will be kept confidential.

Once a report has been received by the project manager, the following steps will be performed until the conflict has been resolved:

1. The project manager will consult the offending individual(s) who were reported as the source(s) of the conflict to obtain all appropriate information regarding the conflict. The project manager will then move on to step two. If the offending individual(s) refuse to cooperate, the project manager may move on to step three after consulting with the team members not involved in the conflict if any exist.
2. Once the project manager has obtained and reviewed the necessary information, a conflict resolution meeting will be held with the offending individual(s), with the intent to resolve the conflict. The project manager will work with the offending individual(s) to create a solution. The affected individual(s) will also be required to attend the meeting. Anonymity practices will be upheld should the person who submitted the conflict resolution report not be a part of the conflict (i.e., is neither the offending nor affected individual(s)). If a resolution to the conflict cannot be achieved, the project manager may move on to step three after consulting with the team members not involved in the conflict if any exist.
3. Should conflict resolution not be achieved in the previous steps, the project manager will then submit a report containing all pertinent information regarding the conflict to the Systems Analysis & Design instructor, Timothy Frantz, along with a request for counsel. If a resolution to the conflict cannot be achieved, the project manager may move on to step four

after consulting with the team members not involved in the conflict if any exist and the instructor.

4. Should conflict resolution still not be achieved at this point, the project manager may enact appropriate disciplinary measures, up to and including termination, as unanimously agreed upon by the team members and defined by the team charter.

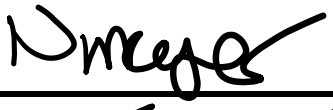
### 5.3. Team Rules of Disciplinary Action

At any time, if it is deemed appropriate, the following actions may be applied according to the severity of the issue at hand.

- Coordinating with the Instructor and by unanimous agreement of those not involved, adjustments for graded assignments related to the individual(s).
- Coordinating with the Instructor and by unanimous agreement of those not involved, termination of team member status of the related individual(s)

### 5.4. Team Member Agreement

By signing below, the following team members have defined and unanimously agreed upon the above rules for resolving conflict within the team and disciplinary action if it is required.

| Team Member Name | Signature |
|---|---|
| Shealyn Cossette |  |
| Claire Fleckney |  |
| Josh Hickman |  |
| Nathaniel Meyer |  |
| Timothy Streibel |  |
| Bryan Towpich |  |

## 6. Team Charter Acknowledgement

By signing below, the following team members unanimously agree upon the above rules, assignment of roles, and guidelines of the SIT SOFTware development team.

Name: Shealyn Cosette Signature:  Date: September 29, 2022

Name: Claire Fleckney Signature:  Date: September 29, 2022

Name: Josh Hickman Signature:  Date: September 29, 2022

Name: Nathaniel Meyer Signature:  Date: September 29, 2022

Name: Tim Streibel Signature:  Date: September 29, 2022

Name: Bryan Towpich Signature:  Date: September 29, 2022

## Appendix B – Client Letter of Commitment

# Client Letter of Commitment

### Client Contact Information

Company Name: Precision Riding Academy

Address: 1505 2nd Avenue South, Lethbridge AB T1J 0E8

Client Contact Name: Bruce Streibel

Email: ridingacademy@ppslethbridge.com

Telephone: (403) 394-6228

As part of Milestone One of our capstone project, we are required to write a client letter of commitment. This letter will summarize your individual business problem, what my team will do to understand your business process, how we will propose and develop a solution, and how we plan to implement the solution and what support we will provide through training and documentation.

### Team Introduction

**Team Name:** SIT SOFTware

**Team Slogan:** Sit Back, Relax.

**Team Logo:**


**Project Name:** Project "Open Road"

Team Members:

| Name | Phone # | Email |
|---|---|---|
| Shealyn Cossette | (403)635-9778 | shealyn.cossette@lethbridgecollege.ca |
| Claire Fleckney | (403) 929-1237 | claire.e.poulsen@gmail.com |
| Josh Hickman | (403) 894-6955 | josh.hickman@lethbridgecollege.ca |
| Nathaniel Meyer | (403) 8948-386 | nathaniel.meyer@lethbridgecollege.ca |
| Timothy Streibel | (403) 394-5787 | timothy.streibel@lethbridgecollege.ca |
| Bryan Towpich | (403) 360-7344 | bryan.towpich@lethbridgecollege.ca |

Team Roles:

| Team Member | Role |
|---|---|
| Timothy Streibel | Project Manager |
| Shealyn Cossette | Change Management Analyst |
| Claire Fleckney | Business Analyst |
| Josh Hickman | Infrastructure Analyst |
| Nathaniel Meyer | System Analyst |
| Bryan Towpich | System Analyst |

## Project Purpose:

The purpose of this project is to create an information system that supports that supports Precision Riding Academy in its mission to offer accessible motorcycle rider safety training to all those that seek to learn or improve the skills required to operate a motorcycle safely.

## Methodology:

To propose and develop a solution my team will be following the systems development lifecycle (SDLC) which breaks the solutions development into four individual parts planning, analysis, design, and implementation. Each part of the SDLC is based on producing specific documents and files that explain various elements of the system. As part of this specific course these deliverables have been broken apart into ten individual milestones which will be completed throughout the following seven months. Upon the completion of each milestone, you will be emailed a copy to review after its submission approval.

During the actual implementation of the system, we will be providing training and support to enable you to make as smooth a transition as possible to the new system. This letter is a part of Milestone One, and upon reading it we will have already completed our first step. The date guidelines for completing the rest of the milestones are as follows:

Milestone 2 Approximate Due Date: Oct 18, 2023

Milestone 3 Approximate Due Date: Oct 31, 2023

Milestone 4 Approximate Due Date: Nov 21, 2023

Milestone 5 Approximate Due Date: Dec 9, 2023

Milestone 6 Approximate Due Date: Winter 2023

Milestone 7 Approximate Due Date: Winter 2023

Milestone 8 Approximate Due Date: Winter 2023

Milestone 9 Approximate Due Date: Spring 2023

Milestone 10 Approximate Due Date: Spring 2023

Throughout the development process my team will be primarily contacting you via email to provide progress updates and requests for information and materials. It is important that we discuss and set specific meeting times and dates as your schedule permits in order to discuss important design decisions to make sure the completed system meets the College's requirements.

Our first step in the first phase of development is Planning. To begin this, we need to meet with you to learn about your business and your needs. We can meet on-site or online as soon as it is convenient for you. As you know, timely communication is crucial to the success of any project, so we are committed to that, as we anticipate you to be, too. Please suggest a couple of possible meeting days morning, or afternoon, and we will arrange to be there, based on our availability within your suggestions.

I look forward to hearing back from you soon!

Regards,

Shealyn Cossette

Shealyn Cossette, Change Management Analyst

# Appendix C – Client Correspondence

**From:** Shealyn Cossette <shealyn.cossette@lethbridgecollege.ca>
**Sent:** Friday, September 23, 2022 8:44 AM
**To:** Precision Riding Academy <ridingacademy@ppslethbridge.com>
**Cc:** Timothy Streibel <timothy.streibel@lethbridgecollege.ca>
**Subject:**

Hello,

My Name is Shealyn, I'm the Change Management Analyst for team SIT SOFTware and I will be your main point of contact for these next upcoming months. I would like to also give you a brief introduction to the other members of this team:

- Timothy Streibel our Project Manager
- Nathaniel Meyer our System Analyst 1
- Bryan Towpich our System Analyst 2
- Claire Fleckney our Business Analyst
- Josh Hickman our infrastructure Analyst

I was hoping to establish a starting point from which we can schedule our initial meeting, if you could tell me a few different times you are available I will try to set up a meeting as best as possible. There will also be a few attachments with these emails, please fill out the forms to the best of your availability. If you have any questions, we will be able to answer them at our next meeting if that works for you.

We can't wait to get started as we are looking forward to meeting with you! Thank you for the opportunity and please, sit back and relax.

Hi Shealyn,

Here's the form you requested I fill out. Let me know when you need more info.

Cheers!

Bruce Streibel

PRA Administrator

**From:** Shealyn Cossette <shealyn.cossette@lethbridgecollege.ca>
**Sent:** Monday, September 26, 2022 6:51:15 PM
**To:** Precision Riding Academy <ridingacademy@ppslethbridge.com>
**Subject:** RE: Meeting Time

Good Evening Bruce,

Thank you for the timely response, I will forward all documents to our project manager for review and cataloging.

The SIT SOFTware team has decided that Friday, Sep 26 at 9am would be the optimal time for our first meeting. We would like to come to Precision Powersports and discuss the project further with you, would this be acceptable?

**From:** Precision Riding Academy
**Sent:** September 27, 2022 9:27 AM
**To:** Shealyn Cossette
**Subject:** Re: Meeting Time

Hi Shealyn,

That time will work fine for me. I look forward to meeting you all.

Cheers!

Bruce

**From:** Shealyn Cossette <shealyn.cossette@lethbridgecollege.ca>
**Sent:** Tuesday, October 11, 2022 2:17:58 PM
**To:** Precision Riding Academy <ridingacademy@ppslethbridge.com>
**Subject:** Mile stone 2

Hey Bruce,

Hope everything has been well, we are looking forward to seeing you and presenting our Progress.

The following Insert is our agenda and mile stone check list feel free to look it over.

Please take note our meeting will be in the Lethbridge College Buchanan Library, in the Breakout Room CE1331 at 5pm. I can't wait and we will see you there!

Hi Shealyn,

Sounds good. Can you suggest a good place to park for that location?

Cheers!

Bruce

**From:** Shealyn Cossette
**Sent:** October 11, 2022 7:22 PM
**To:** Precision Riding Academy

No Problem, Bruce I would suggest Parking lot V it is for guest parking and is close to the library as well. You can enter thought the sports side or walk to the front and the parking is a better price when it comes to time spent.

**From:** Shealyn Cossette <shealyn.cossette@lethbridgecollege.ca>
**Sent:** Sunday, October 23, 2022 3:39:06 PM
**To:** Precision Riding Academy <ridingacademy@ppslethbridge.com>
**Subject:** Milestone 2

Good evening, Bruce, I hope everything is going well and I would like to have you look at the milestone 2 and sign for its completion. Other than that, a reminder as well that on the 27th Thursday we have our client meeting at 5pm, but we have booked a room at the Kingsman!

Shealyn. C.

**From:** Precision Riding Academy
**Sent:** October 25, 2022 11:17 AM
**To:** Shealyn Cossette
**Subject:** Re: Milestone 2

Hi Shealyn,

Sorry for the delay. Here is a signed copy of the Milestone document. Looking forward to Thursday.

Cheers!

Bruce

**From:** Shealyn Cossette <shealyn.cossette@lethbridgecollege.ca>
**Sent:** Wednesday, November 2, 2022 3:06:59 PM
**To:** Precision Riding Academy <ridingacademy@ppslethbridge.com>
**Subject:** RE: Milestone

Hey Bruce, we have the milestone 3 done and ready for you to sign.

**From:** Precision Riding Academy
**Sent:** October 25, 2022 11:17 AM
**To:** Shealyn Cossette
**Subject:** Re: Milestone 2

Very impressive!