|  |
| --- |
|  |
| Behavior Rating Tool |
| Project Plan |
|  |
| **Amanda Isenor**  **Christopher MacDonald**  **Erik Moraru**  **Teng Liu**  **Zak McLure** |
| **24-Jan-21** |

|  |
| --- |
|  |

**Contents**

[Overview 2](#_heading=h.gjdgxs)

[Vision Statement 2](#_heading=h.1fob9te)

[Scope 2](#_heading=h.3znysh7)

[Project Community 3](#_heading=h.2et92p0)

[Key Requirements 3](#_heading=h.tyjcwt)

[Operational Requirements 3](#_heading=h.3dy6vkm)

[System Acceptance Tests 4](#_heading=h.1t3h5sf)

[Use Cases 5](#_heading=h.4d34og8)

[Use Case Diagram 5](#_heading=h.2s8eyo1)

[Use Case Sequence Diagrams 6](#_heading=h.17dp8vu)

[Use Cases Scenarios 12](#_heading=h.44sinio)

[Feature List 18](#_heading=h.3as4poj)

[User Management 18](#_heading=h.1pxezwc)

[General UI 18](#_heading=h.147n2zr)

[Data Display and Management 25](#_heading=h.2r0uhxc)

[Data Processing System 25](#_heading=h.kgcv8k)

[Miscellaneous 27](#_heading=h.1baon6m)

[Delivery 29](#_heading=h.1opuj5n)

[Dependency Chart 29](#_heading=h.48pi1tg)

[Delivery Plan 30](#_heading=h.2nusc19)

[Design 32](#_heading=h.1302m92)

[Use Case Activity Diagrams 32](#_heading=h.3mzq4wv)

[State Diagram 35](#_heading=h.2250f4o)

[Initial Class Definitions 36](#_heading=h.haapch)

[CRC Model 37](#_heading=h.1gf8i83)

# 

# 

# Overview

## Vision Statement

In partnership with the Atlantic Veterinary College, Dr. William Montelpare and his team presented us with the task of developing an educational video hosting platform that is both user-friendly and easy to navigate. The videos will contain footage of equines displaying a variety of behaviours that must be identified by the user in quiz format. The goal of the Equine Behavior Online Rating System is to improve equine welfare and enhance Veterinary Students’ knowledge of Equine Behavior. By working through the proposed self-directed learning module, students will improve their awareness and understanding of equine behaviors within various environments. The system will provide performance scores such as number of correct behaviors identified, number of trials and time taken on task.

## Scope

Equine behaviors rating system is used to enhance veterinary students’ awareness and understanding of equine behavior which translates to improved welfare for teaching horses and enhanced safety for students and horses. In the future, the system can be shared with other veterinary colleges, Animal Health Technology Programs, and other stakeholders for widespread improvement in understanding equine behavior. Not only for equine study, but also can be implemented for other animal studies. System will be presented to the students as a web-based application. After student login to the system, there are a suite of short segment videos randomly distributed to students. For each video student can identify what behaviors they observed, as well as interpretation of the behavior, what action does the behavior led to. Students can review their performance scores and are then presented the options to exit the system or return to the video page and review the video clips and re-submit a new observation sheet. The video management module can be used by the professor or author to upload new videos or edit corresponding descriptors. Students’ scores such as number of correct behaviors identified, number of attempts, and time spent on each task will be recorded and available to the author or admin to access. The scores are valuable in establishing the metrics associated with the implementation of the system as part of the standard curriculum for veterinary students. Student operating data will be collected and tracked such as scores, number of attempts and so on in multiple periods of time such as at the starting of the semester and again at the end of the semester. These data will be used to determine the changes for students in awareness and understanding of equine behaviors. To establish estimates of internal consistency for behavior ratings among equine experts, comparison of the frequency of correct identification of equine behaviors by the novices to the experts will help to determine if classes of errors exist.

## 

## Project Community

The project community consists of six students continuing to CS4820 as well as a professor and two clients and any personnel associated with the clients and this project. Their roles are outlined below:

* Dr. William Montelpare – Client
* Dr. Laurie McDuffee – Client
* Dr. David LeBlanc – Project Supervisor
* Amanda Isenor – Project Lead
* Christopher MacDonald – Technical Lead
* Erik Moraru – Developer
* Teng Liu – Developer
* Zak McLure – Developer
* ~~Zheyi Zheng~~ – **Transferred to Different Group**

## Key Requirements

The technical requirements for this project are as follows:

Language ‒ PHP

Framework ‒ Laravel

Database Management System ‒ MySQL

Web Server ‒ Apache

Web Framework ‒ React

Package Managers ‒ Composer & npm

Version Control ‒ GitHub

IDE ‒ PhpStorm

Text Editor ‒ Visual Studio Code

Environment ‒ Docker

## Operational Requirements

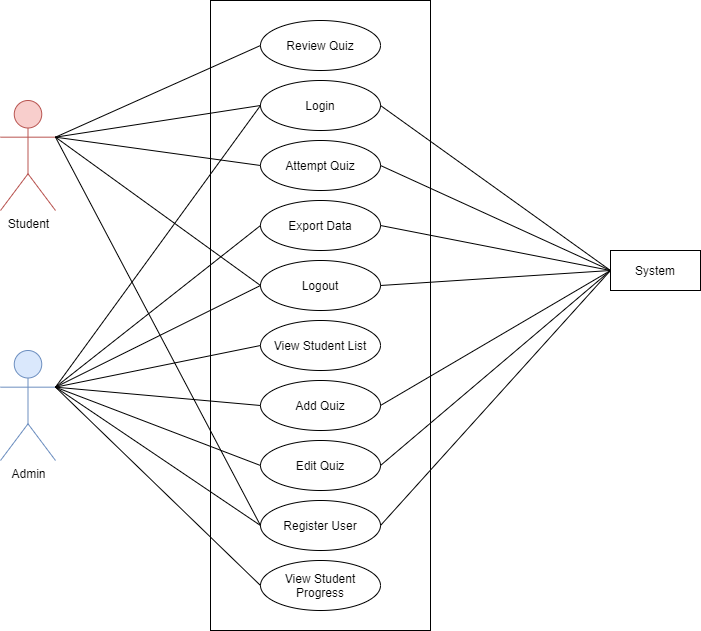
The key operational requirements for this project will include the best practices of documentation, security, ethics, and RESTful API. It will be important to create comprehensive and intuitive documentation at the beginning of the project and maintain that throughout the lifespan of the project to ensure the client can maintain the site on their own or with the help of others. All those who use the site will be assigned accounts and thus security is required, and the project will adhere to security standards and best practices. Given that personal data, such as name, email, and possibly student ID, is being collected, after five (5) years all records must be erased due to ethical implications. User’s scores will be maintained in relation to their graduating year however, their name and all personal information will be removed. Finally, a RESTful API will be implemented to provide better flexibility of data formats.

## System Acceptance Tests

The client will insist on seeing the following before approving the finished product:

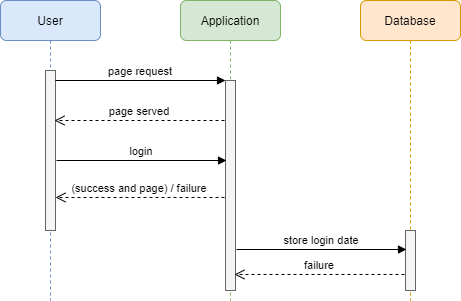
1. Appearance of the tool matches what is desired
2. Key functionality points are met
   1. Users can log in and out
   2. Admin can create accounts
   3. Users can take quizzes and receive feedback
   4. Users can review quizzes and scores
   5. Admin can review all user’s quiz attempts
   6. Admin can export data
3. Data is displayed in a meaningful, intuitive layout
4. The programming languages and techniques used are what was agreed upon
5. Users can only view their own scores unless they have admin credentials
6. All system components function together without conflict, this includes frontend, backend, and database access
7. The system passes all system tests created during development

# Use Cases

**Use Case Diagram**

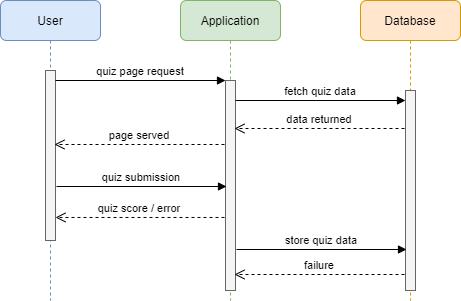
## Use Case Sequence Diagrams

* **Login (User & Admin)**



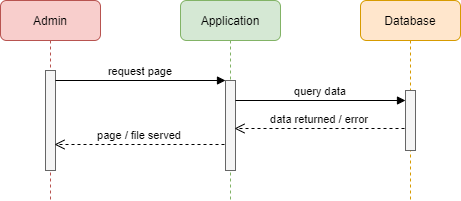
**Use Case Sequence Diagrams**

* **Attempt quiz**
* **Review quiz**



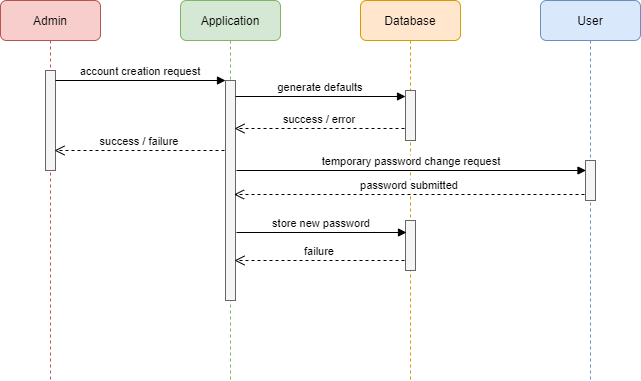
**Use Case Sequence Diagrams**

* **View student list**
* **Export data**
* **View student progress**



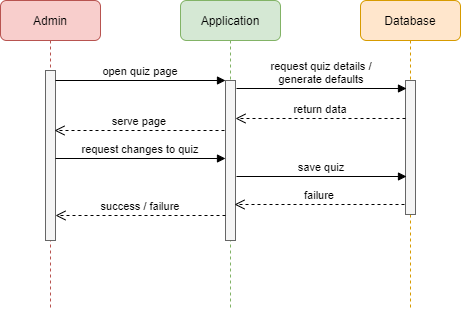
**Use Case Sequence Diagrams**

* **Register User**



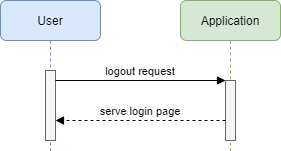
**Use Case Sequence Diagrams**

* **Create a quiz**
* **Edit a quiz**



**Use Case Sequence Diagrams**

* **Log Out (User & Admin)**



## Use Cases Scenarios

**Use Case - Student Registration**

**Primary Actors:** Unregistered student, administrator

**Secondary Actors:** Application

**Preconditions:**

• The unregistered student is in the veterinary program.

**Postconditions:**

• The newly registered student has authenticated access to the application.

**Triggers**:

• The administrator moves to the “create a new account” page.

**Normal Flow:**

1. The administrator enters the unregistered student’s email into the account creation page in the administration panel.

2. The administrator submits the form.

3. The student receives an email confirmation with a temporary password.

4. The student confirms the creation of their account.

5. The student’s account information is stored in the database of the application.

6. The student is brought to a change password page.

7. The student submits a password to replace the generated version.

8. The hashed password is saved to the database of the application.

9. The student is directed to a one-time survey screen

**Alternative flow:**

1. a) The administrator wants to import a list of students all at once and clicks “Import Class List.”

2. a) The administrator enters an invalid email and is instructed to retry

7. a) The student enters a weak password and is required to retry

**Use Case - Login**

**Primary Actors:** User, Administrator

**Secondary Actors:** Application

**Preconditions:**

• The user is registered with the application.

**Postconditions:**

• The registered user has authenticated access to the application.

**Triggers:**

• The user opens the application unauthenticated.

**Normal Flow:**

1. The user attempts to open the application unauthenticated.

2. The user is automatically navigated to the login page.

3. The user enters and submits their information to the login form.

4. The user is authenticated and directed to the home page.

5. The user stores the student’s login date into its database

**Alternative flow:**

3. a)

     a. The user forgets their passwords and clicks the “forgot password” link.

     b. The user receives an email asking to reset their password.

     c. The user clicks the “reset” link and is navigated to a reset password page.

     d. The user enters a new password on the “reset” page and submits.

     e. The user logs in with their new information.

4. a) The user is an administrator and is brought to the admin panel.

**Use Case - Log Out**

**Primary Actors:** User, Administrator

**Secondary Actors:** Application

**Preconditions:**

• The user is in the application.

**Postconditions:**

• The user has logged out of the application.

**Triggers:**

• The user clicks the “logout” button available on any application page.

**Normal Flow:**

1. The user is logged out.

**Alternative flow:**

1. a) The user attempts to log out during a quiz.

     1. The user is prompted with a confirm or cancel dialog.

2. The user makes their selection and either logs out or continues to complete the quiz.

**Use Case - User Attempts Quiz**

**Primary Actors:** User (Primarily the student, but the administrator can access as well for demonstration purposes)

**Secondary Actors:** Application

**Preconditions:**

• The user is authorized to attempt the quiz.

**Postconditions:**

• The user finishes the quiz, adding to their data set.

**Triggers:**

• The user opens the quiz page within the application.

**Normal Flow:**

1. The user selects to start the quiz.

2. The application pulls a list of questions from the database.

3. The user watches the video provided by the application.

4. The user selects an answer from the list of possible solutions within a specific timeframe.

5. The user submits the answer.

6. The user is brought to the end of the quiz, where they are shown their score.

7. The user stores the test information into its database.

**Alternative flow:**

4. a)

     1. The user doesn’t select an answer.

     2. The user submits without an answer.

     3. The user is shown a validation error and is told to select an answer.

6. a) The user is brought to the next question of the quiz.

**Use Case - Administrator Exports Data**

**Primary Actors:** Administrator

**Secondary Actors:** Application

**Preconditions:**

• The administrator is in the admin panel.

**Postconditions:**

• The administrator receives exported data.

**Triggers:**

• The administrator clicks the “export data” button in the admin panel.

**Normal Flow:**

1. The administrator is prompted to select the students included in the data export.

2. The administrator selects the students they wish to be included in the export data.

3. The administrator selects the “submit” button.

4. The application generates a .dat file of all included students’ quiz data.

5. The application prompts the administrator to download the generated file.

6. The administrator downloads the file.

**Alternative flow:**

2. a) The administrator selects a checkbox that selects all students automatically.

**Use Case - Administrator Opens Student List**

**Primary Actors:** Administrator

**Secondary Actors:** Application

**Preconditions:**

• The administrator is in the application’s admin panel.

**Postconditions:**

• The administrator has access to a paginated view of all students.

**Triggers:**

• The user clicks the “students” button in the admin panel.

**Normal Flow:**

1. The application fetches a list of students from the database.

2. The application displays the list in paginated form to the administrator.

3. The administrator can view the paginated list of students.

**Alternative flow:**

3. a) The administrator can search the list of students.

b) The administrator can filter the list of students.

**Use Case - Administrator Adds Quiz**

**Primary Actors:** Administrator

**Secondary Actors:** Application

**Preconditions:**

• The administrator is in the admin panel.

**Postconditions:**

• The administrator added a new quiz to the application.

**Triggers:**

• The administrator clicks on the “new quiz” button in the admin panel.

**Normal Flow:**

1. The application prompts the administrator for quiz info, such as a title, video, answer, and possible answers.

2. The administrator submits the form.

3. The application stores it in the database.

4. The administrator is navigated to a clean “new quiz” page to add another without losing workflow.

**Alternative flow:**

3. a) The form could not be validated, and the administrator is directed to retry.

**Use Case - Administrator Edits Quiz**

**Primary Actors:** Administrator

**Secondary Actors:** Application

**Preconditions:**

• The administrator is in the admin panel.

**Postconditions:**

• The administrator modified an already existing quiz within the application.

**Triggers:**

• The administrator clicks on the “Edit quiz” button in the admin panel.

**Normal Flow:**

1. The application prompts the administrator for quiz info, such as a title, video, answer, and possible answers.

2. The administrator submits the form.

3. The application stores it in the database.

4. The administrator is navigated back to the admin panel.

**Alternative flow:**

3. a) The form could not be validated, and the administrator is directed to retry.

**Use Case - User reviews quiz**

**Primary Actors:** User

**Secondary Actors:** Application

**Preconditions:**

• The user is in the paginated quiz view.

**Postconditions:**

• The user has successfully reviewed their work.

**Triggers:**

• The user clicks on the “Review Quiz” button on the quiz.

**Normal Flow:**

1. The application displays the user’s previous answers to the quiz.

**Alternative flow:**

1. a) There were no previous attempts on the quiz, so a message is displayed to the user by the application letting them know.

**Use Case - Administrator view student progress**

**Primary Actors:** Administrator

**Secondary Actors:** Application

**Preconditions:**

• The administrator is in the paginated student list view.

**Postconditions:**

• The administrator has access to the progress of a specific student.

**Triggers:**

• The administrator clicks on the “View” button on the student’s list item.

**Normal Flow:**

1. The application displays the user’s quiz history

**Alternative flow:**

1. a) There were no quiz attempts made by the student so nothing is shown.

# Feature List

## User Management

**Create user account model**

Description: Program the information held and behaviour of the user account

Priority: 1 (Very high)

Estimated effort required: 3

Estimated time required: 6 hrs

Acceptance Tests: Successful implementation of the features that require user account data.

**Create admin account model**

Description: Program the information and behaviour that will model an admin account

Priority: 1 (Very High)

Estimated effort required: 3

Estimated time required: 6 hrs

Acceptance Tests: Successful implementation of features that require admin account data.

Note: merged into same model, permission distinctions handled by dependency: Bouncer

## General UI

**Design user Landing Page**

Description:  Design how the landing page should look for both guests and Signed-In users

Priority: 1 (Very High)

Estimated effort required: 1 (Very Low)

Estimated time required: 4 hrs

Acceptance Tests: Landing page successfully displayed for guests and signed-in users.

**Implement visual design of user Landing Page**

Description: Program the interface to look as close as possible to the design

Priority: 1 (Very High)

Estimated effort required: 3

Estimated time required: 8 hrs

Acceptance Tests: Successful implementation of landing page visual design.

**Design login View for User**

Description: Design how the login view will look

Priority: 1 (Very High)

Estimated effort required: 2

Estimated time required: 2 hrs

Acceptance Tests: User is able to log in successfully from the login view.

**Implement visual design of login View for User**

Description: Program the interface to look according to design

Priority: 1 (Very High)

Estimated effort required: 2

Estimated time required: 6 hrs

Acceptance Tests: Successful implementation of user login view.

**Design user registration view for User**

Description: Design the “Create an Account” interface

Priority: 1 (Very High)

Estimated effort required: 2

Estimated time required: 4 hrs

Acceptance Tests: A test account is able to be registered.  
Note: reworked into a ‘confirmation’ view that the user is directed to from an email received due an admin creating the user account.

**Implement user registration view for User**

Description: Implement “Create an Account” interface

Priority: 1 (Very High)

Estimated effort required: 2

Estimated time required: 4 hrs

Acceptance Tests: Successful implementation of user registration view.

**Design of User credential recovery**

Description: Design the “Forgot my Password” view

Priority: 1 (Very High)

Estimated effort required: 2

Estimated time required: 1 hrs

Acceptance Tests: The password of a test account is able to be recovered/reset.

Note: using the default Laravel password recovery from the login page, or the admin can click a button to reset the user’s password

**Implement visual design of User credential recovery**

Description: Program the “Forgot my Password” interface

Priority: 1 (Very High)

Estimated effort required: 3

Estimated time required: 2 hrs

Acceptance Tests: Successful implementation of user credential recovery.

Note: using default Laravel ‘forgot password’ page

**Design of “My Account” view**

Description: Design how Signed-In Users “My Account” view will look

Priority:  (2) High

Estimated effort required: 2

Estimated time required: 4 hrs

Acceptance Tests: Test account is able to navigate to “My Account”.

**Implement visual design of “My Account” view**

Description: Program the interface of Signed-In Users “My  Account”

Priority: 2

Estimated effort required: 4

Estimated time required: 8 hrs

Acceptance Tests: Successful implementation of My Account view.

**Design of “Settings” view**

Description: Design the “Settings” view where users can change password, update personal information (name, email address, ~~student ID~~)

Priority: 2

Estimated effort required: 2

Estimated time required: 2 hrs

Acceptance Tests: Test account is able to navigate to “Settings”.

Note: functionality has been partially merged into the ‘My Account’ view/route.

Student ID no longer being used due to privacy concerns. Email is the unique identifier.

**Implement visual design of “Settings” view**

Description: Program the interface of Settings view, where users will be offered common functionality regarding Account Management.

Priority: 3

Estimated effort required: 3

Estimated time required: 6 hrs

Acceptance Tests: Successful implementation of Settings view.

**Design of “Overview” view**

Description: Design the interface users where users are pointed to other parts of the app, such as Taking a Quiz, Reviewing past quizzes and see select relevant statistics of their account.

Priority: 1 (Very High)

Estimated effort required: 3

Estimated time required: 6 hrs

Acceptance Tests: Test account is able to navigate to “Overview” view.

Note: implemented navigation bar that lets users control where they want to go

**Implement visual design of “Overview” view**

Description: Program the interface where Signed-in users can access various parts of the app

Priority: 1 (Very High)

Estimated effort required: 3

Estimated time required: 12 hrs

Acceptance Tests: Successful implementation of “Overview” view.

**Design of “Take a Quiz” view**

Description: Design the user view when a user chooses to take a Quiz, including video player and answer options

Priority: 1 (Very High)

Estimated effort required: 4

Estimated time required: 6 hrs

Acceptance Tests: Test account is able to partake in a quiz.

**Implement visual design of “Take a Quiz” view**

Description: Program the interface for when a user starts taking a quiz, including the video player, answer choices, score, etc

Priority: 1 (Very High)

Estimated effort required: 4

Estimated time required: 16 hrs

Acceptance Tests: Successful implementation of “Take a Quiz” view.

**Design of “Review past Quizzes” view**

Description: Design the user view when they choose in the “Overview” to review past quizzes

Priority: 1 (Very High)

Estimated effort required: 2

Estimated time required: 6 hrs

Acceptance Tests: Test account is able to review the quizzes they have already completed.

**Implement “Review past Quizzes” view**

Description: Program the interface users will see when they choose to review past quizzes

Priority: 1 (Very High)

Estimated effort required: 2

Estimated time required: 16 hrs

Acceptance Tests: Successful implementation of “Review Past Quizzes” view.

**Design login view for Admin user**

Description: Design the sign-in portal for Admin users

Priority: 1 (Very High)

Estimated effort required: 1 (Very High)

Estimated time required: 4 hrs

Acceptance Tests: Test admin account is able to login.

**Implement admin login view**

Description: Program the interface that admins will use to login

Priority: 1 (Very high)

Estimated effort required: 3

Estimated time required: 4 hrs

Acceptance Tests: Successful implementation of admin login.

**Design admin user management view**

Description: Design the user account management view where they can moderate user accounts

Priority: 1 (Very High)

Estimated effort required: 3

Estimated time required: 12 hrs

Acceptance Tests: Test admin account can moderate user accounts.

**Implement visual design admin user management**

Description: Program the interface admins will use to moderate user accounts

Priority: 1 (Very High)

Estimated effort required: 2

Estimated time required: 8 hrs

Acceptance Tests: Successful implementation of admin user management features.

**Design admin user “View user” view**

Description: Design the interface for admins when they select “View user” for specific users

Priority: 1 (Very High)

Estimated effort required: 3

Estimated time required: 6 hrs

Acceptance Tests: Test admin account is able to see relevant information of user accounts.

Note: Missing ability for admins to view user’s quiz history

**Implement visual design for “View user”**

Description: Program the “View user” interface admins will use to inspect relevant information of user accounts (email, name, etc.)

Priority: 1 (Very High)

Estimated effort required: 2

Estimated time required: 6 hrs

Acceptance Tests: Successful implementation of admin “View User” feature.

**Design admin user “Create user account” view**

Description: Design the interface for admins to use when creating user accounts

Priority: 2 (High)

Estimated effort required: 3

Estimated time required: 8 hrs

Acceptance Tests: Test admin account is able to create a new user account.

**Implement visual design of admin user “Create user account”**

Description: Implement the interface admins will use to create user accounts

Priority: 2 (High)

Estimated effort required: 4

Estimated time required: 12 hrs

Acceptance Tests: Successful implementation of admin “Create User Account” feature.

**Design of admin content management overview**

Description: Design admin content management view where it will be used to create, view and edit quizzes on the database

Priority: 1 (Very High)

Estimated effort required: 4

Estimated time required: 6 hrs

Acceptance Tests: Test admin account is able to create and edit quizzes in the database.

**Implement visual design of admin content management index view**

Description: Program the interface where admins will see all quizzes/content and have options to create, view and edit

Priority: 1 (Very High)

Estimated effort required: 2

Estimated time required: 8 hrs

Acceptance Tests: Successful implementation of admin content management interface.

Note: Reworking admin views to a cleaner design, horizontal entries instead of boxes.

**Design admin create quiz view**

Description: Design admin create quiz form

Priority: 1 (Very High)

Estimated effort required: 3

Estimated time required: 4 hrs

Acceptance Tests: Test admin account is able to create a new quiz.

**Implement admin create quiz view**

Description: Program form used by the admin for creating a new quiz

Priority: 1 (Very High)

Estimated effort required: 4

Estimated time required: 8 hrs

Acceptance Tests: Successful implementation of admin create quiz feature.

**Design admin edit quiz view**

Description: Design admin edit quiz form

Priority: 2 (High)

Estimated effort required: 3

Estimated time required: 6 hrs

Acceptance Tests: Test admin account is able to edit existing quizzes.

**Implement visual design of admin edit quiz view**

Description: Program the interface admins will use to edit quizzes

Priority: 2 (High)

Estimated effort required: 4

Estimated time required: 6 hrs

Acceptance Tests: Successful implementation of admin edit quiz feature.

## Data Display and Management

**Setup database**

Description: Setup database within the project

Priority: 1 (Very High)

Estimated effort required: 4

Estimated time required: 8 hrs

Acceptance Tests: Project features are able to successfully interact with the database.

**Create a privilege system for accounts**

Description: System that will enable admins to take actions related to moderating users and content.

Priority: 1 (Very High)

Estimated effort required: 4

Estimated time required: 12 hrs

Acceptance Tests: Successful implementation of privilege system, where admin accounts have access to all features, whereas user accounts have limited privileges.

Note: designed with Bouncer dependency in mind

**Create quiz model**

Description: Create model that stores relevant information and has appropriate behaviour and validations for a quiz

Priority: 1 (Very High)

Estimated effort required: 4

Estimated time required: 8 hrs

Acceptance Tests: Test quiz is able to be created and is fully functional.

## Data Processing System

**Create a system to track user quiz statistics**

Description: Implement a system that tracks quiz statistics, such as score (correct/wrong answers count, attempts)

Priority: 2 (High)

Estimated effort required: 4

Estimated time required: 6 hrs

Acceptance Tests: Test quizzes can be created and have their statistics tracked.

**Create a system to validate quiz creation**

Description: Implement a system for ensuring a quiz reaches a set of requirements, and validates its contents.

Priority: 2 (High)

Estimated effort required: 4

Estimated time required: 6 hrs

Acceptance Tests: Test quiz successfully validated and uploaded to quiz database.

**Create a system to enable video uploading**

Description: Implement system for uploading videos to the appropriate hosting service (e.g. Google Drive).

Priority: 1 (Very High)

Estimated effort required: 4

Estimated time required: 12 hrs

Acceptance Tests: Successful uploading of videos to hosting service.

**Research video players**

Description: Research on best approach how to (if needed) implement a video player.

Priority: 2 (High)

Estimated effort required: 4

Estimated time required: 8 hrs

Acceptance Tests: Present findings on how to implement a video player.

**Implement video player**

Description: Implement required dependencies and logic for a functional front-end video player

Priority: 1 (Very High)

Estimated effort required: 4

Estimated time required: 20 hrs

Acceptance Tests: Video player successfully implemented.

**Create API to handle video between system and storage**

Description: Implement system for upload and loading videos from storage hosting platform

Priority: 1 (Very High)

Estimated effort required: 4

Estimated time required: 8 hrs

Acceptance Tests: Successful implementation of all features involving the uploading and hosting of videos.

**Create API to handle quiz request**

Description: Implement system for when users will request a quiz, to have front-end request relevant information to display the content

Priority: 1 (Very High)

Estimated effort required: 4

Estimated time required: 8 hrs

Acceptance Tests: Test user account is successfully able to request, view, and interact with a quiz.

**Create system for exporting a user’s metrics to a human readable file**

Description: Implement system allowing admins to export the data associated with user accounts to a text file

Priority: 1 (Very High)

Estimated effort required: 4

Estimated time required: 8 hrs

Acceptance Tests: Test admin account is able to successfully export requested user data to a properly formatted file.

## Miscellaneous

**Research mailer services for our project**

Description: Explore available services, and figure out the optimal for our scope/usage

Priority: 2 (High)

Estimated effort required: 3

Estimated time required: 4 hrs

Acceptance Tests: Decide on how to approach mail services for the project.

**Setup chosen mailer service**

Description: Setup mailer service, which in turn enables the application to email the user regarding account administration (such as resetting passwords, account confirmation, account notifications)

Priority: 1 (High)

Estimated effort required: 3

Estimated time required: 8 hrs

Acceptance Tests: Mail services successfully implemented.

**Research Docker and familiarize with the tool**

Description: Based on the use cases, research optimal hosting service (eg. AWS)

Priority: 2\*

Estimated effort required: 3

Estimated time required: 4 hrs

Acceptance Tests: Group members successfully set up development environments on local machines.

**Prepare deployment to production hosting environment for production**

Description: Setup an public environment where production build will be hosted

Priority: 2\*

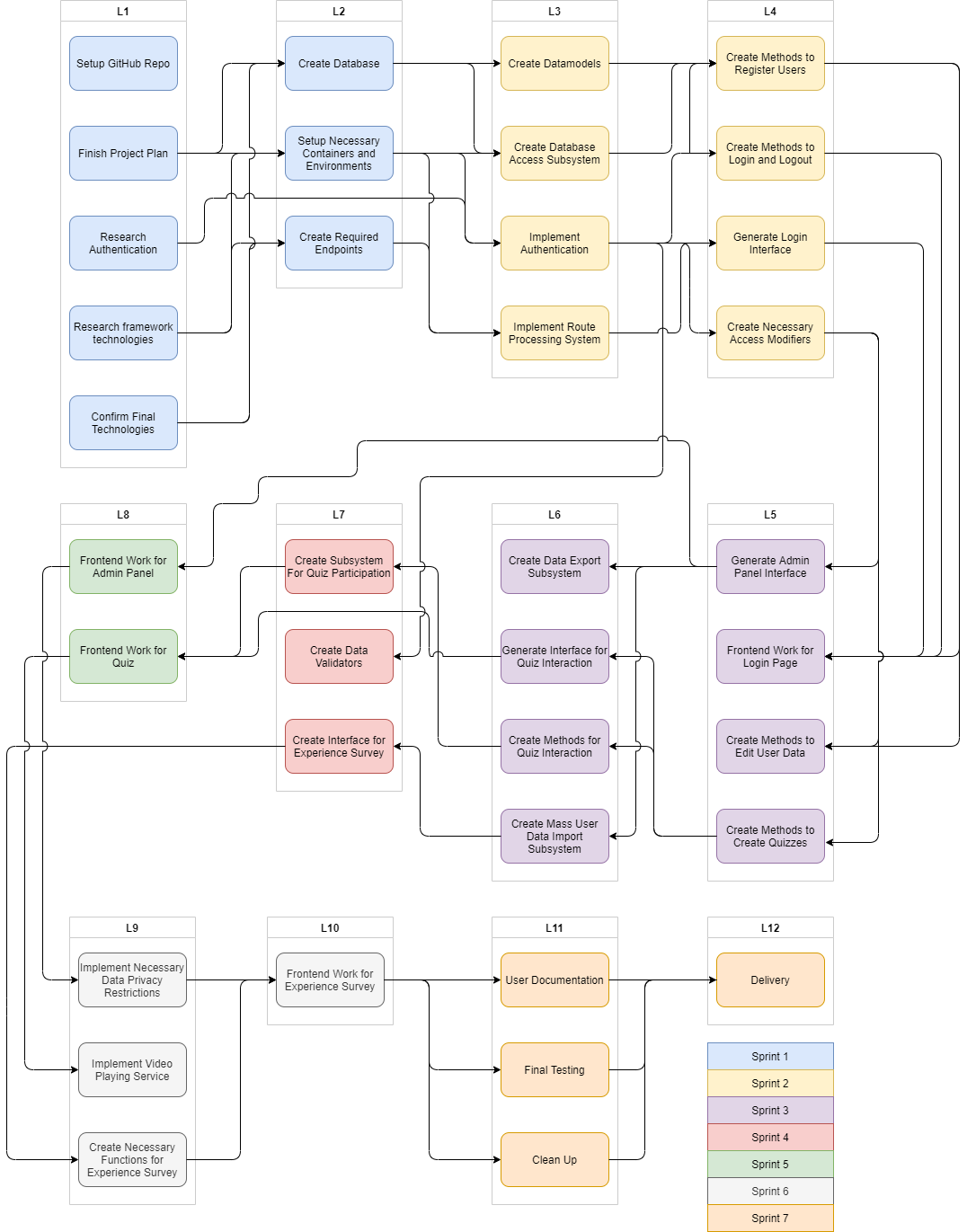
Estimated effort required: 3

Estimated time required: 4 hrs

Acceptance Tests: Successful hosting of the project.

# Delivery

## Dependency Chart

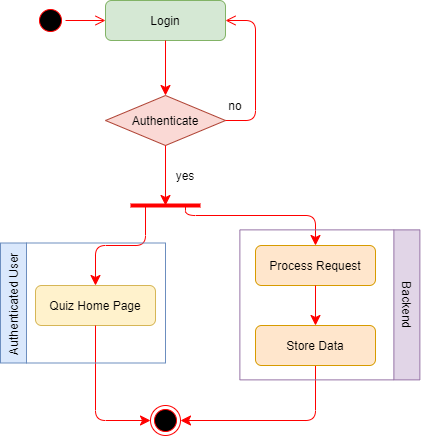


# Design

## Use Case Activity Diagrams

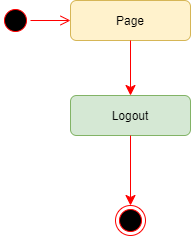
Use Case Activity Diagrams

• Log In



Use Case Activity Diagrams

• Log out



Use Case Activity Diagrams

• Create a quiz

• Edit a quiz

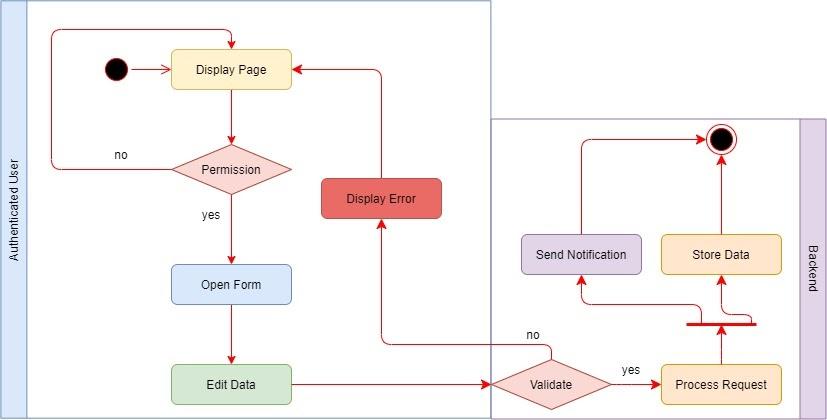
• Attempt quiz

• Review Quiz

• View student list

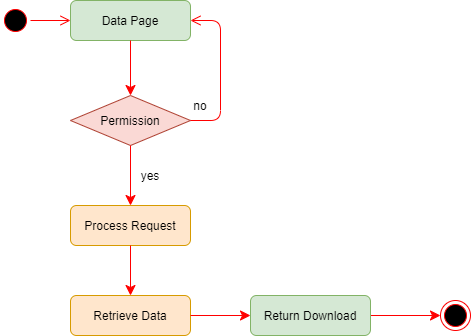
• View student progress

• Register User

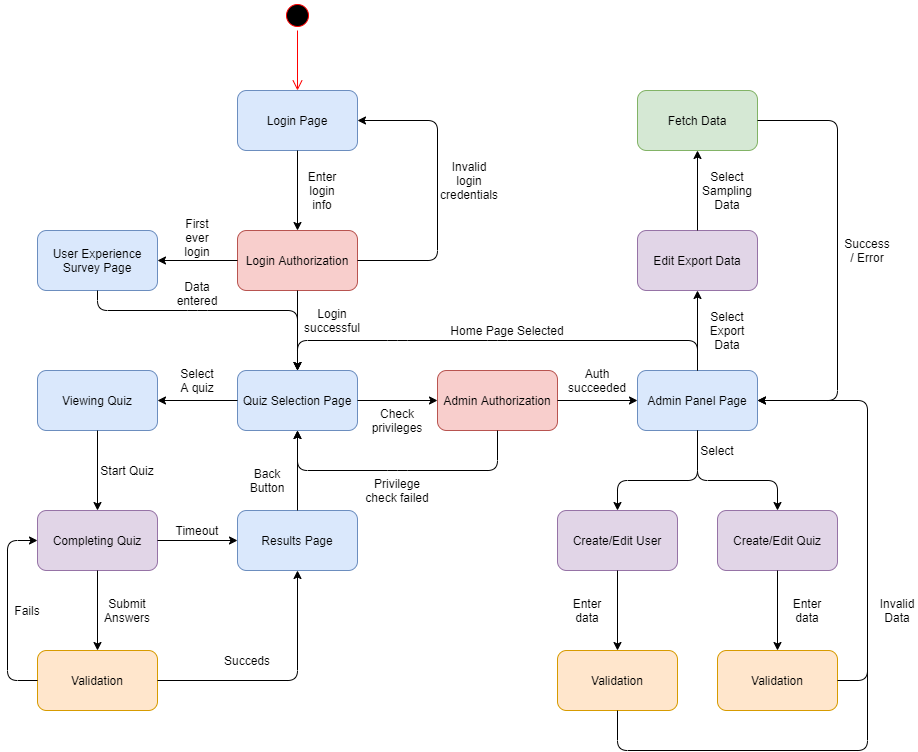


Use Case Activity Diagrams

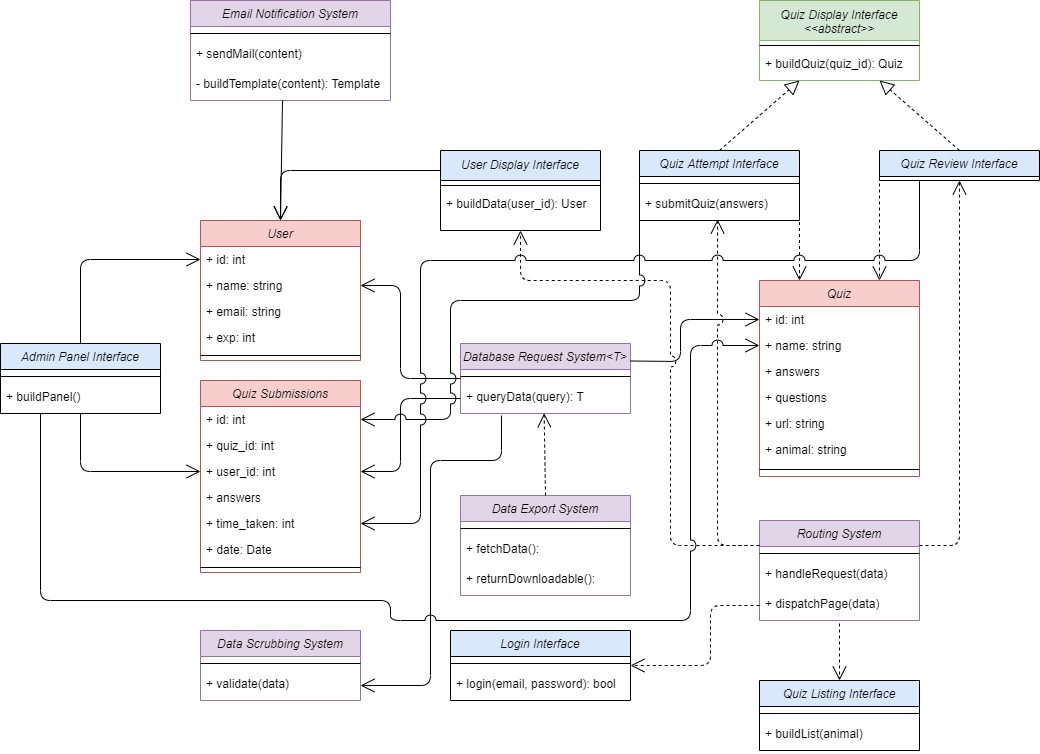
• Export Data



## State Diagram



## Initial Class Definitions

****

## CRC Model

