Software Requirements Specification

for

CZ3003: Software Systems Analysis and Design

Version 1.0 approved

Prepared by Ultrareign Team

Nanyang Technological University

3 September 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Initial Requirement Specification	5/9/2022	-	v1.0.0

1. Introduction

1.1 Purpose

The purpose of this SRS document is to provide a detailed description of the requirements and use cases to build Project Bodyblast: Ultrareign, a user-interactive online educational game system. This document reports the requirements based on the official meetings of the members of Project Bodyblast: Ultrareign from CZ3003 Lab Group TR2. This document will cover the purpose for the development of the system. The system constraints, interface as well as system features will also be explained.

1.2 Document Conventions

All documentation are follows the IEEE SRS formatting requirements.

- Header: Times New Roman, Size 14
- Body: Times New Roman, Size 11
- Margin: Single spacing

1.3 Intended Audience and Reading Suggestions

The intended audiences for this document are developers from the Ultrareign team of NTU School of Computer Science and Engineering (SCSE), and NTU professors in charge of the module to understand the features of the system. This document will be reviewed frequently to check if the different phases of the project are being completed by meeting the given requirements.

The remainder of this document includes the system feature, overall description which are presented to provide functionalities and design of the game. Details of the functional and nonfunctional requirements and use case models will be elaborated in the specific requirements.

1.4 Product Scope

Ultrareign is an educational social ecosystem that gamifies and socializes the teaching and learning of the mathematics subject for audiences aged 6+. This application consists of the Social Education Game, Social Web Page, backend infrastructure and the persistence layer.

The Social Education Game Component is the Main Gaming Application where students are expected to train and learn the core aspect of the mathematics subject via activities designed. The Social Web Page Component is the Main Social User Interface where both the games design and analytics for the professor are displayed and managed. Students will be able to create customized levels as well as interact with other students online by sending challenge requests.

1.5 References

To help in our project development, the following documents are used as reference:

- RPG Maker MZ documentation
  https://developer.rpgmakerweb.com/rpg-maker-mz/
- Shneiderman's 8 Golden Rules
  https://webdesign.tutsplus.com/articles/8-golden-rules-for-better-interface-design--cms-30886

2. Overall Description

2.1 Product Perspective

The team was tasked to create an educational game ecosystem whereby professors would be able to create assignments via game world and students would be able to challenge each other with custom made game levels.

The purpose of the game is to make education fun and interesting for the students.

2.2 Product Functions

The game ecosystem will have the following key functions:

- User Authentication: Allows students and professors to create profiles within the ecosystem, and for login into the game client
- Database Facing Interactions: Allow students and professors to add, retrieve and delete game data from the database
- Game Data and Player Management: Provide students with analytics of their play history
- Gameplay: Allows students to select from challenged, customized or assigned worlds, sections and levels

2.3 User Classes and Characteristics

The product will have two main user classes:

- Professor - Professor will be able to create a new world and add new questions.
- Student - Student playing the game without admin privileges

2.4 Operating Environment

The minimum specifications needed to run the program include:

- Operating System: Windows 10 OS.
- Ram : 4 GB RAM.
- CPU : X64 architecture with SSE2 instruction set support

- GPU : DX10, DX11, and DX12-capable GPUs
- Internet connectivity is required

2.5 Design and Implementation Constraints

The project is designed with the following Design and Implementation Constraints:

Lack of Expertise:
- Developers are Junior Software Developers with Maximum Undergraduate Experiences
- Developers are not specialized in the gaming Development Sector

Lack of Funding:
- Team was required to achieve Project Requirements without Allocated Budget

Lack of Time:
- Team was expected to complete development in the time of 10 Weeks

Request for Documentation:
- Documentation of Software Requirement Specification
- Documentation of Use Case Diagram, Entity Relation Diagram, Dialog Map and other Analysis Models
- Documentation of Subsystem Interface Design and UML component Diagram and communication Diagrams

Request of use of external Interfaces:
- Use of external Social Media Authentication function
- Use of external Social Media Notification function

2.6 User Documentation

System Users Documentation will include the following:
- A demonstration video on core control flow of the application

Development/Testing Team Documentation will include the following:
- The API endpoint Docs The Software Requirement Specification generated via SwaggerUI
- The Use Case Diagram, Entity Relation Diagram, Dialog Map and other Analysis Models
- The Subsystem Interface Design and UML component Diagram and communication Diagrams

2.7 Assumptions and Dependencies

The following are the assumptions made during the development of Ultrareign:

Human Factors
- Users have a stable and fast connection to the internet

- Users have a relatively modern computer running a modern operating system i.e. Window 10 and above
- Users are able to install and run modern web browser i.e. Chrome, Edge, Safari

Software Factors

- RPGMaker's game engine is able to support the performance and communication requirements

The following are the dependencies of Ultrareign:

- Strapi Headless CMS - Used as the backend layer to provider RESTful API endpoints
- MYSQL Databases - Used as the Persistence Data Storage layer of our applications
- RPGMaker - Used as a game engine and framework for designing for the project's interactive games component
- Next.js Framework - Used for the Development of Frontend Web based user interface for world/section/level and user creation and management

3. External Interface Requirements

3.1 User Interfaces

The user interface would follow the following Shneiderman's 8 Golden Rules:

- **Strive for Consistency**
  - The main interface of the game provides a standardized visual layout and style throughout the screens
  - The game follows a consistent color scheme throughout the entire user interface
  - User interface for similar operations (i.e. student/ professor creating levels) will be similar

- **Enable Frequent Users to Use Shortcuts**
  - The system saves the user login information once they log into the system with the 'Remember me' button checked
  - The game would be able to support keyboard shortcuts

- **Offer Informative Feedback**
  - The user views the loading screen while the different pages are loading
  - The system would provide audio feedback in addition to visual feedback
  - The game would show which World, Section and Level the user is at every point in the game

- **Design Dialogs to Yield Closure**
  - The game would produce confirmation pages when the user confirms for an operation to be made

- **Offer Simple Error Handling**

- Users would be prompted to enter the correct information when they have inputted erroneous or no information
- The game would allow users to choose from a fixed list of people for the multiplayer option to reduce errors when searching by typing

- **Permit Easy Reversal of Actions**
  - The game would have "Back" and "Exit" buttons to enable user to reverse their action or exit the game easily

- **Support Internal Locus of Control**
  - There will be no spontaneous operations that occur naturally, and all operations are triggered by the user's actions
  - The system would be able to save user's preferences (e.g. character customization) even after they exit

- **Reduce Short-Term Memory Load**
  - Information are being passed to the next screen without having the user have to remember it
  - The design would be simple and minimalistic and within the recommended 7±2 chunks of information

3.2 Hardware Interfaces

The following hardware interfaces are recommended for use of our game ecosystem:

- Keyboard would be used for typing and gameplay
- Mouse would be used for navigation
- Device (i.e. computer) with Windows Operating System which will host the game
- Device (i.e. computer) supports cellular networks
- Audio for enhanced user experience

3.3 Software Interfaces

- RPG Maker (Ajax)
- RESTful API
- Windows Operating System
- MYSQL Database

3.4 Communications Interfaces

The game application will communicate with our database using a Hypertext Transfer Protocol (HTTP) call to the RESTful API endpoint. HTTP communications are also used when loading user's data including login details and social media sharing.

4. System Features

create use cases based on use case diagram and get functional requirements from use case diagram

User Authentication

4.1 User Login

ID and Name:	UC001 User Login		
Created by:	Hock Tuck	Date created:	03/09/2022
Primary actor:	Student/Professor	Secondary actors:	System
Description:	Student/Professor logs in to the game system.		
Trigger:	User starts the application		
Preconditions:	PRE-1: The user has successfully started the application		
Postconditions:	POST-1: User is logged in to the game system with the user credentials		
Normal flow:	NF-1: Login with user credentials 1. The user opens the application 2. The system displays the login page 3. The user enters his or her username and password 4. The user clicks the login button 5. The system verifies the input username and password 6. The user successfully logs in to the system		
Alternative flows:	AF-S6: If the authentication fails: 1. The system displays a message "Incorrect username or password" 2. The system returns to Step 2		
Exceptions:	EX1: If there is no access to the internet 1. The system displays a message "No internet access, please try again later" 2. The system returns to Step 2		

Functional Requirements:

1. The system shall be able to login to the game system.
2. The system shall be able to display an error message dialog box.

4.2 Login via Social Media

ID and Name:	UC002 User Facebook Login		
Created by:	Hock Tuck	Date created:	03/09/2022
Primary actor:	Student/Professor	Secondary actors:	System
Description:	Student/Professor logs in to the game system.		
Trigger:	User starts the application		
Preconditions:	PRE-1: The user has successfully started the application		
Postconditions:	POST-1: User is logged in to the game system with their Facebook account credentials		
Normal flow:	NF-1: Login with Facebook account 1. The user opens the application 2. The system displays the login page 3. The user clicks the “Login with Facebook” button 4. The system displays the Facebook login page 5. The user enters his or her username and password 6. The user clicks the login button 7. The Facebook authentication system verifies the input username and password 8. The user successfully logs in to the system		
Alternative flows:	AF-S7: If the authentication fails: 1. The system displays a message “Failed to login with Facebook ID” 2. The system returns to Step 2		
Exceptions:	EX1: If there is no access to the internet 1. The system displays a message “No internet access, please try again later” 2. The system returns to Step 2		

Functional Requirements:

1. The system shall be able to login to the game system with Facebook account credentials.
2. The system shall be able to display Facebook login page.
3. The system shall be able to display an error message dialog box.

ID and Name:	UC003 User Twitter Login		
Created by:	Hock Tuck	Date created:	03/09/2022
Primary actor:	Student/Professor	Secondary actors:	System
Description:	Student/Professor logs in to the game system.		
Trigger:	User starts the application		
Preconditions:	PRE-1: The user has successfully started the application		
Postconditions:	POST-1: User is logged in to the game system with their Twitter Account credentials		
Normal flow:	NF-1: Login with Twitter account 1. The user opens the application 2. The system displays the login page 3. The user clicks the “Login with Twitter” button 4. The system displays the Twitter login page 5. The user enters his or her username and password 6. The user clicks the login button 7. The Twitter authentication system verifies the input username and password 8. The user successfully logs in to the system		
Alternative flows:	AF-S7: If the authentication fails: 1. The system displays a message “Failed to login with Twitter ID” 2. The system returns to Step 2		
Exceptions:	EX1: If there is no access to the internet 1. The system displays a message “No internet access, please try again later” 2. The system returns to Step 2		

Functional Requirements:

1. The system shall be able to login to the game system with Twitter account credentials.
2. The system shall be able to display Twitter login page.

3. The system shall be able to display an error message dialog box.

4.3 Connect User Profile

ID and Name:	UC004 Connect User Profile to Facebook.		
Created by:	Hock Tuck	Date created:	03/09/2022
Primary actor:	Student	Secondary actors:	System
Description:	Student connects User Profile to their Facebook account.		
Trigger:	Student must be logged in		
Preconditions:	PRE-1: The user has successfully logged into the application as a student. PRE-1: The user has not connected their User Profile to Facebook.		
Postconditions:	POST-1: User is connected their user profile with their Facebook Account credentials		
Normal flow:	NF-1: Link User Profile to Facebook account 1. The user clicks on “Link account to Facebook” 2. The system displays the Facebook login page 3. The user enters his or her username and password 4. The user clicks the login button 5. The Facebook authentication system verifies the input username and password 6. The user successfully linked their User Profile to Facebook		
Alternative flows:	AF-S5: If the authentication fails: 1. The system displays a message “Failed to login with Facebook ID” 2. The system returns to Step 2		
Exceptions:	EX1: If there is no access to the internet 1. The system displays a message “No internet access, please try again later” 2. The system returns to Step 2		

Functional Requirements:

1. The system shall be able to link User Profile to Facebook account.
2. The system shall be able to display Facebook login page.
3. The system shall be able to display an error message dialog box.

ID and Name:	UC005 Connect User Profile to Twitter.

Created by:	Hock Tuck	Date created:	03/09/2022
Primary actor:	Student	Secondary actors:	System
Description:	Student connects User Profile to their Twitter account.		
Trigger:	Student must be logged in		
Preconditions:	PRE-1: The user has successfully logged into the application as a student. PRE-1: The user has not connected their User Profile to Twitter.		
Postconditions:	POST-1: User is connected their user profile with their Twitter Account credentials		
Normal flow:	NF-1: Link User Profile to Twitter account 1. The user clicks on "Link account to Twitter" 2. The system displays the Twitter login page 3. The user enters his or her username and password 4. The user clicks the login button 5. The Twitter authentication system verifies the input username and password 6. The user successfully linked their User Profile to Twitter		
Alternative flows:	AF-S5: If the authentication fails: 1. The system displays a message "Failed to login with Twitter ID" 2. The system returns to Step 2		
Exceptions:	EX1: If there is no access to the internet 1. The system displays a message "No internet access, please try again later" 2. The system returns to Step 2		

Functional Requirements:

1. The system shall be able to link User Profile to Twitter account.
2. The system shall be able to display Twitter login page.
3. The system shall be able to display an error message dialog box.

4.4 Create User Profile

ID and Name:	UC006 Create User Profile		
Created by:	Hock Tuck	Date created:	03/09/2022
Primary actor:	Professor	Secondary actors:	System
Description:	Professor Creates a User Profile.		

Trigger:	Professor must be logged in.
Preconditions:	PRE-1: The user has successfully logged into the application as a professor.
Postconditions:	POST-1: The user has created a user profile.
Normal flow:	NF-1: Create User Profile 1. The user clicks on “Create User profile”. 2. The user indicates the username, password, school and class. 3. If the user is a professor, he or she checks the “Is Staff” checkbox. 4. The user clicks on “Add User profile”. 5. The system will display a message “User profile successfully created” on the screen.
Alternative flows:	AF-S2: If the username is already taken 1. The system displays “Username is taken”. 2. The system returns to step 2.
Exceptions:	-

Functional Requirements:

1. The system shall be able to create a User Profile.
2. The system shall be able to display an error message dialog box.

Database Facing Interaction

4.5 CRUD Question

ID and Name:	UC007 CRUD Questions		
Created by:	Ernest Tan Yan Heng	Date Created:	03/09/2022
Primary actor:	Professor	Secondary Actors:	System
Description:	User is able to Create, View, Update and Delete questions from the database		
Trigger:	Professor clicks on "Manage Questions" button		
Preconditions:	PRE-1: User is logged into their Professor Account		
Postconditions:	POST-1: User is logged into their Professor Account		
Normal flow:	NF-1: Create 1. System redirects user to the “Manage Questions” Page 2. System retrieves a list of existing questions within the database		

	3. System displays a listview of the existing questions 4. User clicks on "Add New Question" button 5. User is redirected to the Question Input Use Case, UC006 NF-2: Read 1. System redirects user to the “Manage Questions” Page 2. System retrieves a list of existing questions within the database 3. System displays a listview of the existing questions NF-3: Update 1. System redirects user to the “Manage Questions” Page 2. System retrieves a list of existing questions within the database 3. System displays a listview of the existing questions 4. User clicks on “Edit Question” button 5. User clicks on the question to be edited 6. User is redirected to the Question Input Use Case, UC006 NF-4: Delete 1. System redirects user to the “Manage Questions” Page 2. System retrieves a list of existing questions within the database 3. System displays a listview of the existing questions 4. User clicks on “Delete Question” button 5. User clicks on the question to be deleted 6. System displays a confirmation dialog box, with “Confirm” and “Cancel” options 7. User clicks on the “Confirm” button 8. System removes the question from the database 9. System returns user to the “Manage Questions” Page
Alternative flows:	AF-4-5: User cancels Deletion 1. User clicks on the “Cancel” button 2. System returns user to the “Manage Questions” Page AF-1-4 / AF-3-4 / AF-4-4: User returns to Main Page 1. User clicks on the “X” button at the top left corner of the “Manage Questions” page 2. User is redirected to the Main Menu page
Exceptions:	EX-1: No Existing Questions within database 1. System displays a message “There are currently no existing questions, please add one now!” 2. System automatically redirects user to Question Input Use Case, UC006

Functional Requirements:

1. The system shall be able to create new questions
2. The system shall be able to retrieve a list of questions from the database
3. The system shall be able to display a list of questions from the database
4. The system shall be able to update a question within the database

5. The system shall be able to delete a question from the database
6. The system shall be able to send User to the Question Management page
7. The system shall be able to send User to the Question Input page
8. The system shall be able to display an error message dialog box

4.6 Question Input

ID and Name:	UC008 Question Input		
Created by:	Ernest Tan Yan Heng	Date Created:	03/09/2022
Primary actor:	Professor	Secondary Actors	System
Description:	Users fill in an input form to add questions into the database		
Trigger:	User clicks on the “Add New Question” button		
Preconditions:	PRE-1: User is logged into Professor account		
Postconditions:	POST-1: User is logged into Professor account		
Normal flow:	NF-1: Create 1. System redirects user to the question input form 2. User enters the question 3. User enters the corresponding correct answer 4. User clicks on the "Create" button 5. System creates a new question within the database 6. System redirects User to the “Manage Questions” page NF-2: Update 1. System retrieves the question and answer data from the database 2. System redirects user to the question input form 3. User makes changes to the input within the input form 4. User clicks on the “Save” button 5. System saves the changes within the database 6. System redirects User to the “Manage Questions” page		
Alternative flows:	AF-1-4 / AF-2-4: User cancels the creation/update of question 1. User clicks on the “X” button 2. System redirects user to the “Manage Questions” page		
Exceptions:	EX-1: Required fields are empty 1. System displays an error dialog box with the message, “There are mandatory fields empty! Please fill them in before you save!” 2. User clicks on the “Ok” button 3. User is redirected to the question input form		

Functional Requirements:

1. The system shall be able to redirect Users to the Question Management page

4.7 CRUD Single Player Levels

ID and Name:	UC009, CRUD All Levels		
Created by:	Ernest Tan Yan Heng	Date Created:	03/09/2022
Primary actor:	Professor	Secondary Actors	System
Description:	User is able to Create, View, Edit or Delete all levels from the database		
Trigger:	User clicks on the “Manage Levels” button		
Preconditions:	PRE-1: User is logged into a Professor account		
Postconditions:	POST-1: User is logged into a Professor account		
Normal flow:	NF-1: Create 1. System redirects user to the “Manage Levels” Page 2. System retrieves a list of existing levels within the database 3. System displays a listview of the existing levels 4. User clicks on "Add New Level" button 5. User is redirected to the Level Input Use Case, UC009 NF-2: Read 1. System redirects user to the “Manage Levels” Page 2. System retrieves a list of existing levels within the database 3. System displays a listview of the existing levels NF-3: Update 1. System redirects user to the “Manage Levels” Page 2. System retrieves a list of existing levels within the database 3. System displays a listview of the existing levels 4. User clicks on “Edit Level” button 5. User clicks on the level to be edited 6. User is redirected to the Level Input Use Case, UC009 NF-4: Delete 1. System redirects user to the “Manage Levels” Page 2. System retrieves a list of existing levels within the database 3. System displays a listview of the existing levels 4. User clicks on “Delete Level” button 5. User clicks on the level to be deleted 6. System displays a confirmation dialog box, with “Confirm” and “Cancel” options 7. User clicks on the “Confirm” button 8. System removes the level from the database 9. System returns user to the “Manage Levels” Page		

Alternative flows:	AF-4-5: User cancels Deletion 1. User clicks on the “Cancel” button 2. System returns user to the “Manage Levels” Page AF-1-4 / AF-3-4 / AF-4-4: User returns to Main Page 1. User clicks on the “X” button at the top left corner of the “Manage Levels” page 2. User is redirected to the Main Menu page
Exceptions:	EX-1: No Existing Levels within database 1. System displays a message “There are currently no existing levels, please add one now!” 2. System automatically redirects user to Level Input Use Case, UC009

Functional Requirements:

1. The system shall be able to create new levels
2. The system shall be able to retrieve a list of levels from the database
3. The system shall be able to display a list of levels from the database
4. The system shall be able to update a level within the database
5. The system shall be able to delete a level from the database
6. The system shall be able to send User to the Level Management page
7. The system shall be able to send User to the Level Input page
8. The system shall be able to display an error message dialog box

4.8 CRUD Custom Levels

ID and Name:	UC010 CRUD Custom Levels		
Created by:	Ernest Tan Yan Heng	Date Created:	03/09/2022
Primary actor:	Student	Secondary Actors	
Description:	User is able to Create, View, Edit or Delete customized levels from the database		
Trigger:	User selects the “Manage Custom Levels” button		
Preconditions:	PRE-1: User is logged into a Student account		
Postconditions:	POST-1: User is logged into a Student account		
Normal flow:	NF-1: Create 1. System redirects user to the “Manage Levels” Page 2. System checks key ID value of user 3. System retrieves a list of existing levels tagged to the key ID value 4. System displays a listview of the existing custom levels 5. User clicks on "Add New Level" button		

	6. User is redirected to the Level Input Use Case, UC009 NF-2: Read 1. System redirects user to the "Manage Levels" Page 2. System checks key ID value of user 3. System retrieves a list of existing levels tagged to the key ID value 4. System displays a listview of the existing custom levels NF-3: Update 1. System redirects user to the "Manage Levels" Page 2. System checks key ID value of user 3. System retrieves a list of existing levels tagged to the key ID value 4. System displays a listview of the existing custom levels 5. User clicks on "Edit Level" button 6. User clicks on the level to be edited 7. User is redirected to the Level Input Use Case, UC009 NF-4: Delete 1. System redirects user to the "Manage Levels" Page 2. System checks key ID value of user 3. System retrieves a list of existing levels tagged to the key ID value 4. System displays a listview of the existing custom levels 5. User clicks on "Delete Level" button 6. User clicks on the level to be deleted 7. System displays a confirmation dialog box, with "Confirm" and "Cancel" options 8. User clicks on the "Confirm" button 9. System removes the level from the database 10. System returns user to the "Manage Levels" Page
Alternative flows:	AF-4-5: User cancels Deletion 1. User clicks on the "Cancel" button 2. System returns user to the "Manage Custom Levels" Page AF-1-4 / AF-3-4 / AF-4-4: User returns to Main Page 1. User clicks on the "X" button at the top left corner of the "Manage Custom Levels" page 2. User is redirected to the Main Menu page
Exceptions:	EX-1: No Existing Levels within database System displays a message "There are currently no existing levels, please add one now!" System automatically redirects user to Level Input Use Case, UC009

Functional Requirements:

1. The system shall be able to check the UID of the student
2. The system shall be able to create new custom levels
3. The system shall be able to retrieve a list of custom levels from the database
4. The system shall be able to display a list of custom levels from the database

5. The system shall be able to update a custom level within the database
6. The system shall be able to delete a custom level from the database
7. The system shall be able to send User to the Custom Level Management page
8. The system shall be able to display an error message dialog box

4.9 Level Creation

ID and Name:	UC011 Level Input		
Created by:	Ernest Tan Yan Heng	Date Created:	03/09/2022
Primary actor:	Professor	Secondary Actors	System
Description:	Users fill in an input form to add levels into the database		
Trigger:	User clicks on the “Add New Level” or “Edit Level” button		
Preconditions:	PRE-1: User is logged into Professor account		
Postconditions:	POST-1: User is logged into Professor account		
Normal flow:	NF-1: Create 1. System redirects user to the level input form 2. User selects the questions for the specific level from the question bank 3. User clicks on the "Create" button 4. System creates a new level within the database 5. System redirects User to the “Manage Levels” page NF-2: Update 1. System retrieves the questions from the database 2. System redirects user to the level input form 3. User makes changes to the question selection within the level 4. User clicks on the “Save” button 5. System saves the changes within the database 6. System redirects User to the “Manage Levels” page		
Alternative flows:	AF-1-3 / AF-2-4: User cancels the creation/update of question 1. User clicks on the “X” button 2. System redirects user to the “Manage Levels” page		
Exceptions:	EX-1: Required fields are empty 1. System displays an error dialog box with the message, “There are mandatory fields empty! Please fill them in before you save!” 2. User clicks on the “Ok” button 3. User is redirected to the level input form		

Functional Requirements:

1. The system shall be able to redirect Users to the Level Management page

4.10 CRUD Worlds

ID and Name:	UC012 CRUD Worlds		
Created by:	Tan Yunliang	Date created:	03/09/2022
Primary actor:	Professor	Secondary actors:	System
Description:	Users need to be able to modify, create, delete and view Worlds.		
Trigger:	User enter the world listview page.		
Preconditions:	PRE-1: User is logged into their Professor account.		
Postconditions:	POST-1: User is logged into their Professor account.		
Normal flow:	NF-1: Create 1. User clicks on the Create World button. 2. User directed to World Creation Use Case UC011. NF-2: Read 1. User select world from listview to interact 2. The System pulls Data from the Backend with the key value of the world selected and greets User with a Pop Up World Description View Dialog. NF-3: Update 1. User select world from listview to interact 2. The System pulls Data from the Backend with the key value of the world selected and greets User with a Pop Up World Description View Dialog. 3. Pop Up World Description View Dialog will check for permission access to update selected world. A Update button will be shown if permission is granted. 4. User clicks on the Update button. 5. User directed to World Creation Use Case UC011 with the key value of the world selected. NF-4: Delete 1. User select world from listview to interact 2. The System pulls Data from the Backend with the key value of the world selected and greets User with a Pop Up World Description View Dialog. 3. Pop Up World Description View Dialog will check for permission access to update selected world. A delete button will be shown if		

	permission is granted. 4. User clicks on the Delete button. 5. User is shown a Confirmation Pop Up with Confirm and Cancel Button. 6. User clicks on the Confirm Button. 7. The System deletes the world from Database. 8. The System returns User to listview.
Alternative flows:	AF-NF-4-5 1. User clicks on the Cancel Button. 2. The System returns User to Pop Up World Description View Dialog. AF-NF-4-2 / AF-NF-3-3 / AF-NF-4-3 / AF-NF-4-5-2 1. User clicks on the "X" Button on the Dialog Box. 2. The System returns User to listview.
Exceptions:	EX1: Session Data is not found 1. User is returned to the login page Use Case UC001. EX2: No World in database 1. User is shown a friendly background image with text that displays "There is no world created yet. create one by pressing the create world button."

F**unction Requirement:**

1. The System shall be able to send List of World Data to the Web Frontend Interface.
2. The System shall be able to send User to the World Creation Page.
3. The System shall be able to send User to the World Creation Page with a World UID.
4. The System shall be able to display world listview (web).
5. The System shall be able to display Pop Up World Description View Dialog.
6. The System shall be able to display Pop Up World Description View Dialog with default data from the world Data.
7. The System shall be able to display background images and text behind the world listview (web).
8. The System shall be able to remove select Worlds from its database.
9. The System shall be able to update select Worlds from its database.
10. The System shall be able to create select Worlds from its database.

4.11 World Creation

ID and Name:	UC013 World Creation		
Created by:	Yunliang	Date created:	03/09/2022
Primary actor:	Professor	Secondary actors:	System
Description:	Users need a clean Form User Interface to Create or Modify World Data.		

Trigger:	User enters the World Creation Page.
Preconditions:	PRE-1: User is logged into their Professor account.
Postconditions:	POST-1: User is logged into their Professor account.
Normal flow:	NF-1 : Create 1. The System checks the Session Data for the Account ID. 2. The System checks the key value called with the World Creation Page. If the key value of the World is not present. The form User Interface is present with default values and the Create Button. 3. User is shown the required fields in the form via an asterisk next to the field Name. 4. User filled in all the required fields and optionally other fields that User would like to add. 5. User selects the Create Button. 6. The System checks Data that are filled in via all the fields and verifies input given. 7. The System creates a new world entry in the database. NF-2 : Update 1. The System checks the Session Data for the Account ID. 2. The System checks the key value called with the World Creation Page. If the key value of the World is present, the data of the keyed world is pulled from the Database. 3. The Session Data is then cross referenced with Database Data to check if the user is permissioned. If so, the form User Interface is presented with existing Database values and the Update Button. 4. User is shown the required fields in the form via an asterisk next to the field Name. 5. User makes the necessary modification to the Data field. 6. User selects the Update Button. 7. The System checks Data that are filled in via all the fields and verifies input given. 8. The System updates the world entry in the database.
Alternative flows:	AF-NF-1-4 / AF-NF-2-4 : 1. User Clicks the Cancel Button or the “x” Button. 2. User is shown a Confirmation Pop Up with text “are you sure, you would like to exit the page? all changes and modifications made will be lost.” with the Confirm and Cancel Button. 3. If the Confirm Button is clicked, the User is sent to the World listview. If the Cancel Button is clicked, the User is returned back to the form User Interface.

Exceptions:	EX1: Session Data is not found 1. User is returned to the login page Use Case UC001. EX2: Session Data is not permissioned 1. User Receives An Error Page notifying that his account is not permissioned.

Function Requirement:

1. The System shall be able to retrieve User Session UID from the Web Frontend Interface.
2. The System shall be able to retrieve UID of World via communication between Web Front Interface.
3. The System shall be able to send Specified World Data requested by its UID.
4. The System shall be able to send List of Assignment Data to Web Frontend Interface.
5. The System shall be able to send the user to the Error Page.
6. The System shall be able to display a Confirmation Pop Up.
7. The System shall be able to create World entry into the Database.

Gamedata and player Management

4.12 Assign Worlds

ID and Name:	UC014 Assigned Worlds		
Created by:	Yunliang	Date created:	03/09/2022
Primary actor:	Professor	Secondary actors:	System
Description:	Users need to be able to assign worlds to students.		
Trigger:	User enters the Assignment Dialog Box / Widget.		
Preconditions:	PRE-1: User is logged into their Professor account. PRE-2: The System receives a key of the selected world.		
Postconditions:	POST-1: User is logged into their Professor account.		
Normal flow:	NF-1 Assignment Process : 1. The System pulls the data of the student and assignment from the database. A Listview of all students are shown along with a check mark indicating the particular student's assignment. A ticked checkmark, represented as assigned vice versa. 2. User clicks the checkmark mark to toggle the student assignment to the world. 3. User clicks the Submit Button. 4. The System updates the world entry in the database.		

	5. The System checks if Notification is Toggled. If so Message Assignment Notification via Use Case UC013 NF-2. NF-2 Setting Notification Message : 1. The System pulls the data of the student and assignment from the database. A Listview of all students are shown along with a check mark indicating the particular student's assignment. Below the Listview locates the Send Notification check mark. 2. User Toggles the Send Notification checkmark to "Ticked" for sending Notification of the change to the respective student Update Button press.
Alternative flows:	AF-NF-1-4 / AF-NF-2-4 1. User Clicks the "x" Button. 2. The User is returned back to the form User Interface by the System.
Exceptions:	EX1: Session Data is not found 1. User is returned to the login page Use Case UC001. EX2: No Student in database User is shown a friendly background image with text that displays "There is no world created yet. create one by pressing the create world button."

Function Requirement:

1. The System shall be able to send List of Students Data to Web Frontend Interface.
2. The System shall be able to send List of Assignment Data to Web Frontend Interface.
3. The System shall be able to send Custom Notification
4. The System shall be able to send Specified Assignment Data requested by its UID.
5. The System shall be able to call the internal function to send Notification with a List of Students UID.
6. The System shall be able to display the assignment listview (web).
7. The System shall be able to display background images and text behind the assignment listview (web).
8. The System shall be able to update the Assignment Database with A List of Student UID and their assignment status.

4.13 Message Assignment via Social Media

ID and Name:	UC0015 Message Assignment via Social Media		
Created by:	Yunliang	Date created:	03/09/2022
Primary actor:	Professor	Secondary actors:	System
Description:	Users need to be able to Message the Assignment of Worlds via Social Media to Students User.		
Trigger:	User Submit Assignment		

Preconditions:	PRE-1: The System receives a List of Students to notify. PRE-2: The System receives the Notification Message to send.
Postconditions:	POST-1: Success Message Return from Social Media Endpoint.
Normal flow:	NF-1 Sending Social Media Notification : 1. The System checks Connected Social Media Profile of the Student in the List. 2. The System invokes the internal function that iteratively selects other preset function calls to the student's connected social media platform. 3. For all Promises, The System Waits till all returns with a message. 4. All errors are consolidated into a log and returned to the user. 5. User receives a Success Message if there is no log produced else, receives an Error Message with log on display.
Alternative flows:	-
Exceptions:	-

Function Requirement:

1. The System shall be able to send List of Students Data to Web Frontend Interface.
2. The System shall be able to filter List of Students by their connected Social Media UID.
3. The System shall be able to call and use notification facebook API endpoint to send a notification Message.
4. The System shall be able to call and use Twitter facebook API endpoint to send a notification Message.
5. The System shall be able to email students a notification message.
6. The System shall be able to display a success Message box.
7. The System shall be able to display an error Message box.

4.14 See Personal Progress

ID and Name:	UC0016 See Personal Progress		
Created by:	Lydia	Date created:	04/09/2022
Primary actor:	Student	Secondary actors:	System
Description:	User views personal progress		
Trigger:	User selects on the personal progress button		
Preconditions:	PRE-1: User is logged into their student account		

Postconditions:	POST-1: User is logged into their student account
Normal flow:	NF-1: 1. User selects “See Personal Progress” button. 2. System retrieves and displays Individual Student Level Mastery. 3. User views Individual Student Level Mastery or selects “See Level Scoreboard”. 4. User views their personal progress. 5. System returns to respective page.
Alternative flows:	
Exceptions:	EX-1: No level available for display 1. User is shown a friendly background image with text that displays “No available data for display.”

Functional Requirements:

1. System should be able to retrieve scores from database.
2. System should be able to display the scores to the users.
3. System should be able to compute and filter scores of users.
4. System should be able to display computed and filtered scores in the level scoreboard.
5. System should be able to retrieve student’s level mastery.
6. System should be able to display student’s level mastery.

4.15 See Individual Student Mastery

ID and Name:	UC0017 See Individual Student Mastery		
Created by:	Lydia	Date created:	04/09/2022
Primary actor:	Student/Professor	Secondary actors:	System
Description:	User views individual student’s mastery		
Trigger:	User selects the “See Personal Progress” or “Get Summary Report”		
Preconditions:	PRE-1: User is logged into their student/professor account		
Postconditions:	POST-1: User is logged into their student/professor account		

Normal flow:	NF-1: 1. User selects the "See Personal Progress" or "Get Summary Report" button. 2. System pulls the levels data of students from the database. 3. System filtres data and select the most recent level of students. 4. System displays individual student level mastery (level). 5. User views displayed individual student level mastery.
Alternative flows:	
Exceptions:	EX-1: No level available for display 1. User is shown a friendly background image with text that displays "No available data for display."

Functional Requirements:

1. System should be able to retrieve scores from database.
2. System should be able to display the scores to the users.
3. System should be able to compute and filter scores of users.
4. System should be able to display computed and filtered scores in the level scoreboard.
5. System should be able to retrieve student's level mastery.
6. System should be able to display student's level mastery.

4.16 See Level Scoreboard

ID and Name:	UC0018 See Level Scoreboard		
Created by:	Lydia	Date created:	04/09/2022
Primary actor:	Student	Secondary actors:	System
Description:	User views level scoreboard		
Trigger:	User selects "View Level Scoreboard"		
Preconditions:	PRE-1: User is logged into their student account		
Postconditions:	POST-1: User is logged into their student account		
Normal flow:	NF-1: 1. User selects the "View Level Scoreboard" button. 2. System displays available levels. 3. User selects level. 4. System computes and filter scores of users in selected level from the database. 5. System pulls filtered data.		

	6. System displays user's name, score and level. 7. User views the scoreboard.
Alternative flows:	
Exceptions:	EX1: No data from level available for display 1. User is shown a friendly background image with text that displays "Try this level to get into the scoreboard!"

Functional Requirements:

1. System should be able to retrieve scores from database.
2. System should be able to display the scores to the users.
3. System should be able to compute and filter scores of users.
4. System should be able to display computed and filtered scores in the level scoreboard.
5. System should be able to retrieve student's level mastery.
6. System should be able to display student's level master

4.17 Get Summary Report

ID and Name:	UC0019 Get Summary Report		
Created by:	Lydia	Date created:	04/09/2022
Primary actor:	Professor	Secondary actors:	System
Description:	User gets student's summary report		
Trigger:	User selects "Get Summary Report"		
Preconditions:	PRE-1: User is logged into their professor account		
Postconditions:	POST-1: User is logged into their professor account		
Normal flow:	NF-1: 1. User selects the "Get Summary Report" button. 2. System retrieves and displays individual students mastery. 3. User views individual students mastery. 4. System returns to respective page.		
Alternative flows:			
Exceptions:	EX1: No data from level available for display 1. User is shown a friendly background image with text that displays "No data available for display."		

Functional Requirements:

1. System should be able to retrieve students mastery from database.
2. System should be able to display the students mastery scores to the users.
3. System should be able to return professor to respective calling page.

Gameplay

4.18 See In Game Custom Level List

ID and Name:	UC020 See In Game Custom Level List		
Created by:	Yunliang	Date created:	03/09/2022
Primary actor:	Professor / Students	Secondary actors:	System
Description:	Users should be able to view and select from the Custom Level list to play in-game.		
Trigger:	Users enter the game Level List View.		
Preconditions:	PRE-1: User is in Game Session login into their Student Account.		
Postconditions:	POST-1: User is in Game Session login into their Student Account.		
Normal flow:	NF-1 : 1. The System pulls filtered Level data (Custom) from the database. 2. The System displays a Listview of Custom Level List. 3. The User selects the custom level. 4. The System pulls the select Level data from the database.		
Alternative flows:	AF-1 See Custom Level Filtered by Challenges : 1. The System pulls Challenge Data from the database. 2. The System pulls filter Level data (Challenges) Data from the database. 3. The System displays a Listview of Challenged Level List. 4. The User selects the challenged custom level. 5. The System pulls the select Level data from the database.		
Exceptions:	EX1: No Level in database: 1. User is shown a friendly background image with text that displays “There is no custom level created yet. create one @ { URL to platform’s student web portal}.” EX2: No Challenge in database: 1. User is shown a friendly background image with text that displays “No one has challenged you to an intellectual duel. Play a single player		

	world?”

Functional Requirement:

1. The System shall be able to send List of Level Data to its Game Frontend Interface.
2. The System shall be able to send a Specified Level Data filtered by Custom Level label to its Game Frontend Interface.
3. The System shall be able to send List of Challenges Data to its Game Frontend Interface.
4. The System shall be able to filter List of Level Data by Custom/Single Player Label.
5. The System shall be able to filter List of Level Data by List of Challenge (inner Join).
6. The System shall be able to filter List of Challenges Data by User Sessions’ UID.
7. The System shall be able to display a listview of Custom Level In Game.
8. The System shall be able to display the custom level listview (in game).
9. The System shall be able to display background images and text behind the custom level listview (in game).

4.19 See In Game Single Player Level List

ID and Name:	UC021 See Assigned World/Level filtered by Assignments		
Created by:	Yunliang	Date created:	03/09/2022
Primary actor:	Professor / Students	Secondary actors:	System
Description:	Users should be able to view and select from the Single Player Worlds list to play in-game.		
Trigger:	Users enter the game World List View.		
Preconditions:	PRE-1: User is in Game Session login into their Student Account.		
Postconditions:	POST-1: User is in Game Session login into their Student Account.		
Normal flow:	NF-1 : 1. The System pulls Assignment Data from the database. 2. The System pulls filtered World data (Assignment) from the database. 3. The System pulls filtered Level data (filtered World) from the database. 4. The System displays a Listview of Single Player World List and Level it is made of. 5. The User selects the Single Player World. 6. The System pulls the filtered Levels (selected World) data from the database.		

Alternative flows:	
Exceptions:	EX1: No World in database: 2. User is shown a friendly background image with text that displays "There is no custom level created yet. create one @ { URL to platform's student web portal}." EX2: World has no Levels: 1. The system removes the world from the listview.

Function Requirement:

1. The System shall be able to send List of Assignment Data to the Game Frontend Interface.
2. The System shall be able to send List of World Data to the Game Frontend Interface.
3. The System shall be able to send List of Level Data to the Game Frontend Interface.
4. The System shall be able to filter Level Data by World UID.
5. The System shall be able to filter World Data by List of Assignment (inner join).
6. The System shall be able to filter List of Assignment by User Sessions' UID.
7. The System shall be able to display a listview of Custom World and its Levels In Game.
8. The System shall be able to display the single player world listview (in game).
9. The System shall be able to display background images and text behind the single player world listview (in game).
10. The System shall be able to remove selected Worlds from its listview.

4.20 Play Education Game

ID and Name:	UC022 Plays Education Game		
Created by:	Sheryl Goh	Date Created:	03/09/2022
Primary actor:	Student	Secondary Actors	System
Description:	User plays the educational game		
Trigger:	User selects the "Play Game" button		
Preconditions:	PRE-1. User must be logged in		
Postconditions:	POST-1. User gets a score after every question POST-2. User's game data is saved into database		
Normal flow:	1. User selects "Play Game" button 2. User selects "Challenges", "Custom" or "SinglePlayer" button		

	3. User completes the specific levels 4. System would display user's score 5. System would save user's score details into the database 6. System returns to respective page
Alternative flows:	
Exceptions:	

Functional requirements:

1. System shall be able to retrieve questions
2. System shall be able to display questions to user
3. System shall be able to calculate scores the user obtained from the level
4. System shall be able to display the score the user has obtained
5. System shall be able to store the user's score details in the database

4.21 Play Challenge Level

ID and Name:	UC023 Plays Challenge Level		
Created by:	Sheryl Goh	Date Created:	03/09/2022
Primary actor:	Student	Secondary Actors	System
Description:	User plays the challenge level		
Trigger:	User selects "Challenges" button		
Preconditions:	PRE-1. User must be logged in PRE-2. The system has challenge levels in database		
Postconditions:	POST-1. User gets a score after completing the level POST-2. User's score details are saved into database		
Normal flow:	1. User selects "Challenges" button 2. System will display a list of challenge levels the user has received 3. User selects a level 4. User answers questions in the level 5. System would display the user's and their challenger's score 6. System would save user's score details in database 7. System returns to display other challenge levels the student has		
Alternative	AF5: Challenger has not completed the level		

flows:	1. System would display the user's score only AF7: User does not have anymore challenge levels 1. System display "No more challenges" message to user 2. User will acknowledge the message 3. System will bring user back to choose "Challenges", "Custom" or "SinglePlayer" button
Exceptions:	EX2: User does not have challenge levels 1. System display "No more challenge levels to complete" message to user 2. User will acknowledge the message 3. System will bring user back to choose "Challenges", "Custom" or "SinglePlayer" button

Functional requirements:

1. System shall be able to retrieve challenge levels assigned to user
2. System shall be able to display list of challenge levels assigned to user
3. System shall be able to retrieve questions from chosen challenge level
4. System shall be able to display challenge questions to user
5. System shall be able to calculate scores the user obtained from the level
6. System shall be able to display the score the user has obtained
7. System shall be able to retrieve challenger's score
8. System shall be able to store the user's score details in the database

4.22 Play Custom Level

ID and Name:	UC024 Plays Custom Level		
Created by:	Sheryl Goh	Date Created:	03/09/2022
Primary actor:	Student	Secondary Actors	System
Description:	User plays the custom level		
Trigger:	User selects "Custom" button		
Preconditions:	PRE-1. User must be logged in PRE-2. The system has custom levels in database		
Postconditions:	POST-1. User gets a score after completing the level POST-2. User's score details are saved into database		
Normal flow:	1. User selects "Custom" button 2. System will display a list of custom levels created by other students		

	3. User selects a level 4. User answers questions in the level 5. System would display the user's score 6. System would display the user's score compared to the rest of the students who have completed that level 7. System would save user's score details in database 8. System returns to display other custom levels
Alternative flows:	AF8: User does not have anymore custom levels to complete 1. System display "No more custom levels to complete" message to user 2. User will acknowledge the message 3. System will bring user back to choose "Challenges", "Custom" or "SinglePlayer" button
Exceptions:	EX2: User does not have custom levels 1. System display "No more custom levels to complete" message to user 2. User will acknowledge the message 3. System will bring user back to choose "Challenges", "Custom" or "SinglePlayer" button

Functional requirements:

1. System shall be able to retrieve custom levels in the database the user has yet to complete
2. System shall be able to display list of custom levels to user
3. System shall be able to retrieve questions from chosen custom level
4. System shall be able to display questions to user
5. System shall be able to calculate scores the user obtained from the level
6. System shall be able to display the score the user has obtained
7. System shall be able to retrieve other student's score from the level
8. System shall be able to display the scores of other students
9. System shall be able to store the user's score details in the database

4.23 Play Single Player World

ID and Name:	UC025 Plays Single Player Level		
Created by:	Sheryl Goh	Date Created:	03/09/2022
Primary actor:	Student/ Professor	Secondary Actors	System
Description:	User plays the single player level		
Trigger:	User selects "SinglePlayer" button		

Preconditions:	PRE-1. User must be logged in PRE-2. The system has single player levels in database
Postconditions:	POST-1. User gets a score after completing the level POST-2. User's score details are saved into database
Normal flow:	1. User selects "SinglePlayer" button 2. System will display a list of assigned worlds by their professor 3. User selects a world 4. User selects a Section and Level from the world 5. User answers questions in the level 6. System would display the user's score 7. System would save user's score details in database 8. System returns to display other worlds assigned
Alternative flows:	AF8: User does not have anymore assigned worlds to complete 4. System display "No more assigned worlds to complete" message to user 5. User will acknowledge the message 6. System will bring user back to choose "Challenges", "Custom" or "SinglePlayer" button
Exceptions:	EX2: User does not have assigned worlds 4. System display "No more assigned worlds to complete" message to user 5. User will acknowledge the message 6. System will bring user back to choose "Challenges", "Custom" or "SinglePlayer" button

Functional requirements:

1. System shall be able to retrieve worlds assigned to user by their professor
2. System shall be able to display list of assigned worlds to user
3. System shall be able to retrieve questions from chosen world, section and level
4. System shall be able to display questions to user
5. System shall be able to calculate scores the user obtained from the level
6. System shall be able to display the score the user has obtained
7. System shall be able to store the user's score details in the database

1. Other Nonfunctional Requirements

1.1 Performance Requirements

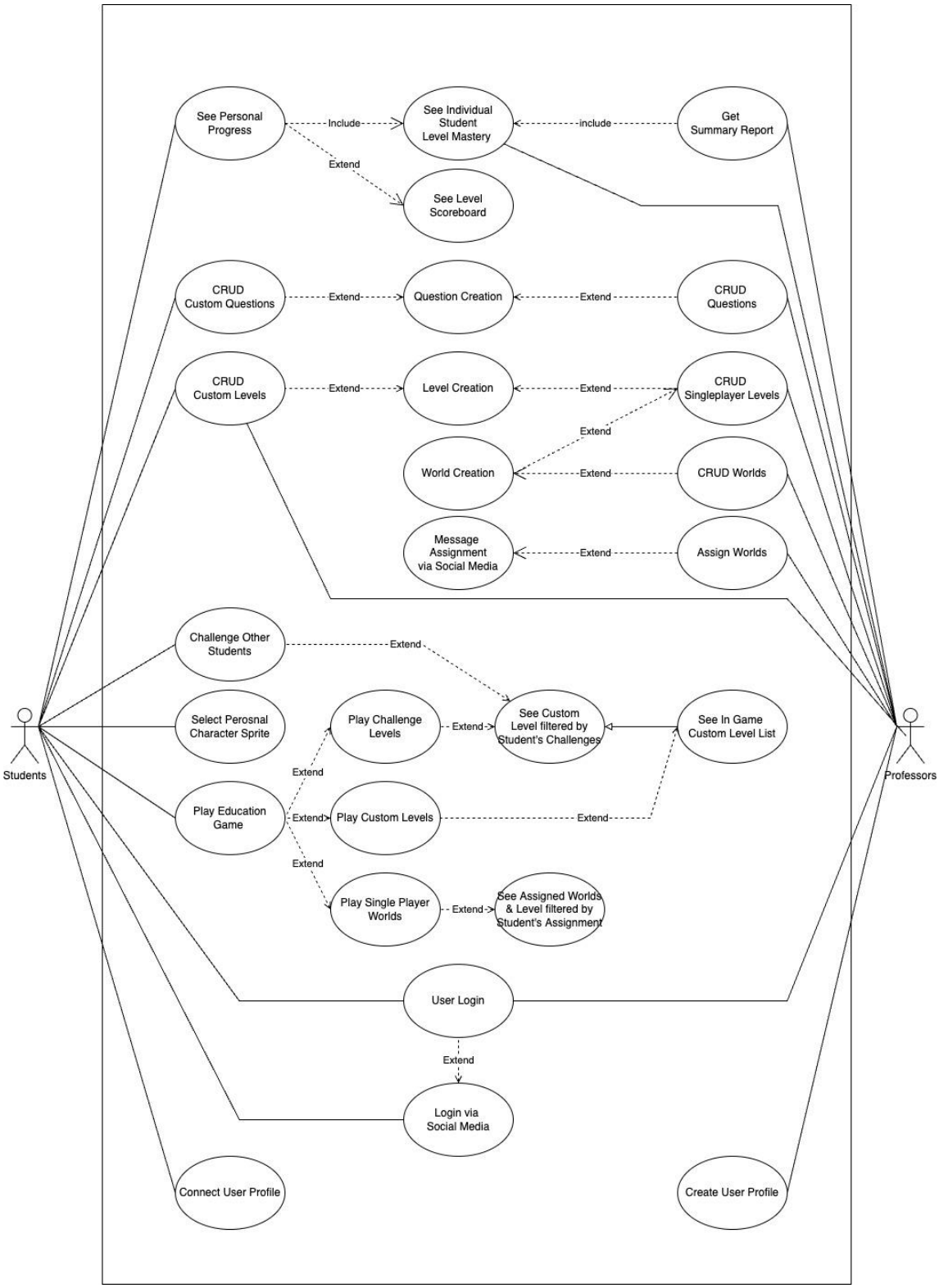
- The system shall have a response time of less than 10 seconds after every user action
- The system must offer informative feedback
- The system must display error messages that allows the user to know what went wrong when an error occurs

- The system must allow easy reversal of the user's actions

Appendix A: Data Dictionary

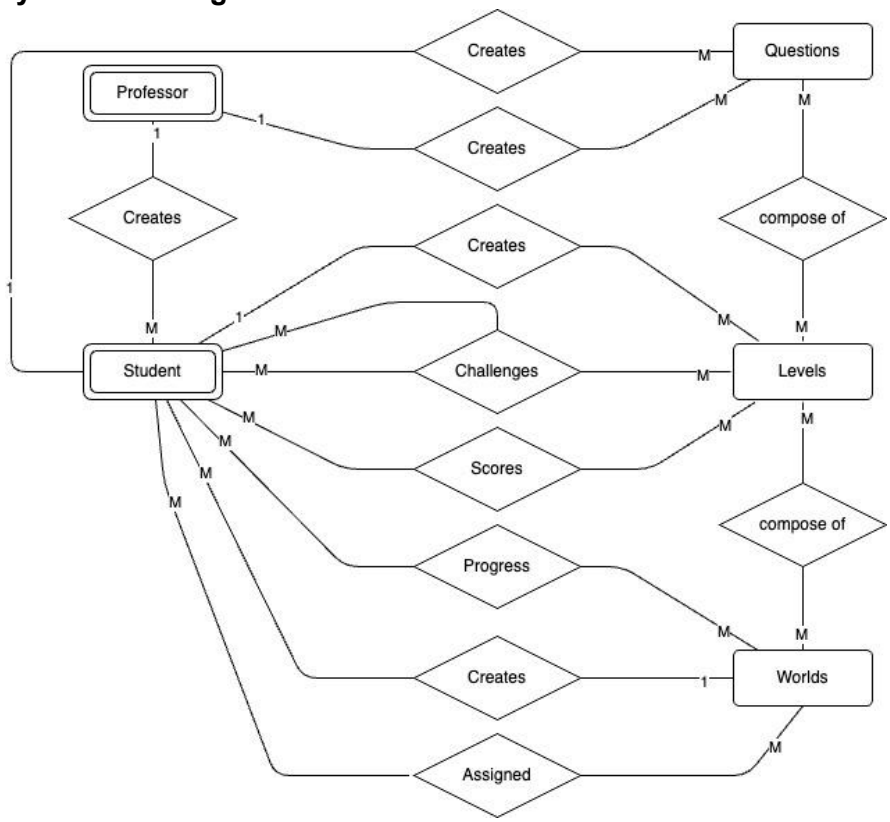
Name	Description
HTTP	Refers to Hypertext Transfer Protocol which is used to sends requests from our game client
JSON	Refers to the standard file format used to store and transfer data
CRUD	Refers to the 4 basic operations of persistent storage: Create, Read, Update, Delete
World	Refers to a specific course or topic
Section	Refers to a specific chapter within the specified course
Level	Refers to a collection of questions of varying difficulty within a specified chapter
User	Refers to either a student or professor
RPG Maker	Refers to the game client engine used to develop the game
Next.js	Refers to a frontend JSON framework
Strapi	Refers to a backend headless content management system

Appendix B: Use Case Model

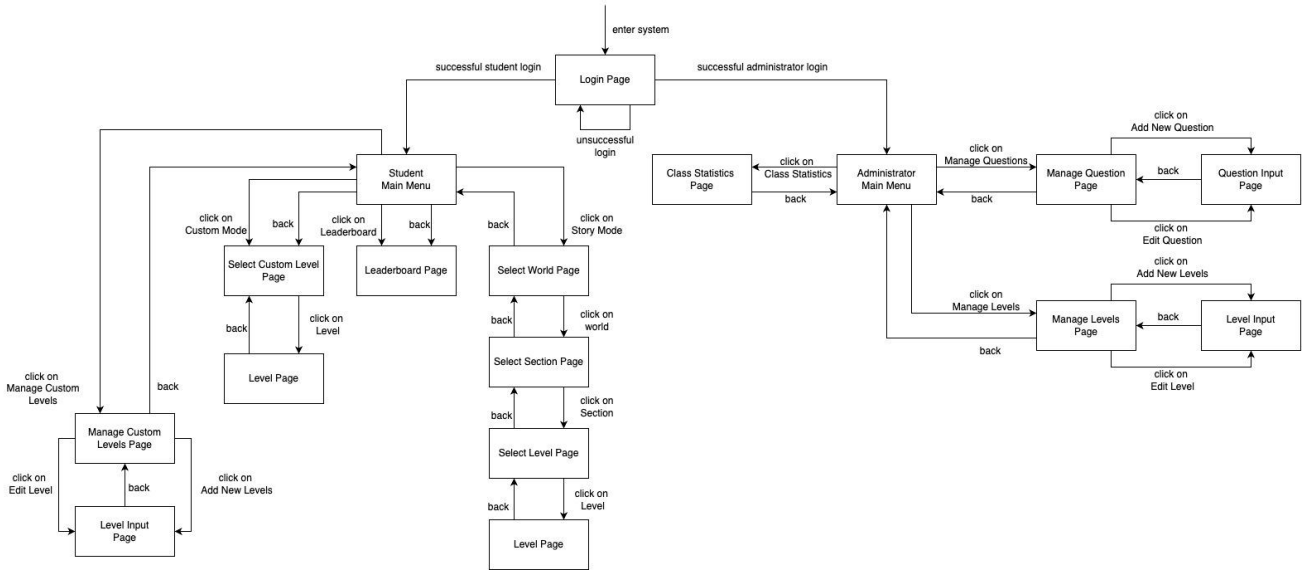


Appendix C: Analysis Models

Simplified Entity Relation Diagram:



Dialog Map :



Appendix D: Risk Reward Analysis

Use Case	*Benefit*	*Penalty*	*Total Value*	*Value %*	*Rel. Cost*	*Cost %*	*Rel. Risk*	*Risk %*	*Priority*
User Login	*8*	*7*	*15*	*4.37%*	*7*	*3.80%*	*3*	*4.76%*	0.51
Login Via Social Media	*4*	*7*	*11*	*3.21%*	*7*	*3.80%*	*4*	*6.35%*	0.32
Connect User Profile	*4*	*7*	*11*	*3.21%*	*7*	*3.80%*	*4*	*6.35%*	0.32
Create User Profile	*6*	*8*	*14*	*4.08%*	*8*	*4.35%*	*4*	*6.35%*	0.38
CRUD Question	*8*	*10*	*18*	*5.25%*	*10*	*5.43%*	*2*	*3.17%*	0.61
Question Input	*8*	*7*	*15*	*4.37%*	*7*	*3.80%*	*2*	*3.17%*	0.63
CRUD All Levels	*8*	*10*	*18*	*5.25%*	*10*	*5.43%*	*2*	*3.17%*	0.61
CRUD Custom Levels	*7*	*8*	*15*	*4.37%*	*8*	*4.35%*	*2*	*3.17%*	0.58
Level Input	*8*	*6*	*14*	*4.08%*	*6*	*3.26%*	*2*	*3.17%*	0.63
CRUD Worlds	*8*	*10*	*18*	*5.25%*	*10*	*5.43%*	*2*	*3.17%*	0.61
World Creation	*8*	*10*	*18*	*5.25%*	*10*	*5.43%*	*2*	*3.17%*	0.61

Assign Worlds	*8*	*8*	*16*	*4.66%*	*8*	*4.35%*	*2*	*3.17%*	0.62
Message Assignment Via Social Media	*5*	*7*	*12*	*3.50%*	*7*	*3.80%*	*4*	*6.35%*	0.34
See Individual Student Mastery	*7*	*10*	*17*	*4.96%*	*10*	*5.43%*	*3*	*4.76%*	0.49
See Level Scoreboard	*7*	*10*	*17*	*4.96%*	*10*	*5.43%*	*3*	*4.76%*	0.49
Get Summary Report	*7*	*9*	*16*	*4.66%*	*9*	*4.89%*	*3*	*4.76%*	0.48
See All Custom Levels	*8*	*7*	*15*	*4.37%*	*7*	*3.80%*	*3*	*4.76%*	0.51
See All Singleplayer Worlds	*8*	*7*	*15*	*4.37%*	*7*	*3.80%*	*4*	*6.35%*	0.43
Play Education Game	*8*	*9*	*17*	*4.96%*	*9*	*4.89%*	*3*	*4.76%*	0.51
Play Challenge Level	*8*	*9*	*17*	*4.96%*	*9*	*4.89%*	*2*	*3.17%*	0.61
Play Custom Level	*8*	*9*	*17*	*4.96%*	*9*	*4.89%*	*4*	*6.35%*	0.44
Play Singleplayer World	*8*	*9*	*17*	*4.96%*	*9*	*4.89%*	*3*	*4.76%*	0.51
Totals	*159*	*184*	*343*	*100.00 %*	*184*	*100.00 %*	*63*	*100.00 %*	