
Software Requirements Specification

for

HUF

Version 1.4

Prepared by Team HUF

Nanyang Technological University

13th Nov 2021

1. Table of Content

1. Table of Content.....	1
2. Team Members.....	2
3. Revision History.....	2
4. Introduction	3
4.1. Purpose	3
4.2. Document Conventions	3
4.3. Intended Audience & Reading Suggestions	3
4.4. Product Scope	4
4.5. Quality Attributes	4
4.6. References	5
5. Overall Description	5
5.1. Product Perspective	5
5.2. Product Functions	6
5.3. User Classes and Characteristics	6
5.4. Operating Environment	7
5.5. Design and Implementation Constraints	7
5.6. Assumptions and Dependencies	8
6. External Interface Requirements	8
6.1. User Interfaces.....	8
6.2. Hardware Interface	20
6.3. Software Interface	20
6.4. Communications Interface.....	20
7. Functional Requirements.....	20
7.1. System Functionality to be Performed.....	20
7.2. Use Case Diagram	28
7.3. Use Case Description.....	29
8. Non-Functional Requirements.....	43
8.1. Performance Requirements.....	43
8.2. Security Requirements.....	44
8.3. Usability Requirements	45
8.4. Reliability Requirements	45
8.5. Scalability Requirements	46
8.6. Compatibility.....	46
9. Analysis Models.....	46
9.1. System Architecture	46
9.2. Context Diagram.....	48
9.3. Data Flow Diagram	49
9.4. Entity Relationship Diagram	50
9.5. Dialog Map.....	51
9.6. Component Diagram.....	52
9.7. Communication Diagram (Game and Quiz Creation)	53
9.8. Communication Diagram (Gameplay).....	54
9.9. Communication Diagram (Facebook Login).....	55
10. Sub System Interface.....	56
11. Test Plan.....	61
11.1. Testing Strategies and Techniques	61
11.2. Blackbox Testing.....	62
11.3. System Testing	68
11.3.1. Functional Testing	68
11.3.2. Performance Testing.....	72

11.3.3. Integration Testing.....	73
11.3.4. Unit Testing	77
Appendix A: Data Dictionary.....	80

2.Team Members

Name	Matriculation No.
Aileen Laksmono Lie	U1920118E
Ashwin Kurup	U1921557H
Bachhas Nikita	U1921630F
Ng Chi Hui	U1922243C
Ong De Quan, Daniel	U1820131B
Rajuravi Vishal Raj	U1822268B
Sim Tian Quan	U1921887D
Song Yu	U1922469E
Teng Sok Ee	U1921479L
Zoe Lim	U1830848H

3.Revision History

Name	Date	Reason For Changes	Version
All members	19/9/2021	Addendum of analysis models and subsystem interface.	1.1
All members	10/10/2021	Addendum of component diagram, activity diagram and test cases for black box testing.	1.2
All members	21/10/2021	Finalise contents to be included in SRS Report	1.3
All members	13/11/2021	Final amendments to contents of SRS Report	1.4

4.Introduction

4.1. Purpose

This document lays out a plan for the development of HUF, a web-based game application by Team HUF. The intended readers of this document are for the current and future stakeholders that are involved in the development of HUF. This document will comprise of, but not restricted to, a summary of the system functionality, functional and non-functional requirements, Use Case models, User-Interface model and other relevant analysis models.

4.2. Document Conventions

This report follows the ‘IEEE Software Requirements Specification’.

This document is written by following the formats as described below:

1. **Section Heading:** ‘Times New Roman’, Bold, 18
2. **Sub-Section Heading:** ‘Times New Roman’, Bold, 16
3. **Body Content:** ‘Times New Roman’, 12

If needed to be written in a hierarchical structure, we will abide to the following rule:
A body content that is part of an existing section will be numbered in accordance with the section it belongs to, followed by its own sequence.

Example as shown:

1. Existing section
 - 1.1. Body content belonging to section 1.
 - 1.1.1.Body content belonging to section 1.1.
2. 2nd body content belonging to section 1.

4.3. Intended Audience & Reading Suggestions

This document is intended for all stakeholders of the system. This can include anyone from business executives to project managers, to system developers and testing teams of the application etc.

The various aspects of the system will be sectioned as follows:

1. Overall Description
2. External Interface Requirement
3. Functional Requirements

4. System Diagrams
5. Non-functional Requirements
6. Appendix

It is recommended that stakeholders first refer to Appendix A: Glossary to understand the terminologies used in the document.

Business Executives, management personnel may wish to refer to section 1.

End users may wish to refer to section 2.

Engineers (software, hardware, system), testing group, maintenance and support team may wish to refer to section 3 onwards.

4.4. Product Scope

HUF will be a web-based application, and the game will come in the form of a set of quizzes. It will be designed with a point system/leader board to keep track of the scores in the game. It will also be accompanied with a dashboard to gain analytical insights specific to each game. This will aid in promoting more interaction and engagement, as students will get to enjoy learning by having fun.

Having such application may also improve overall grades and learning satisfaction/quality of a student as they are given more chances to practice and test their knowledge and understanding of what is being taught through a friendly competition within a stress-free environment in the game, rather than through continuous tests that have permanent effect on their grades. This in turn might have an indirect positive effect on students' overall wellbeing, encouraging them to learn more through our proposed application.

Currently, the project is being developed for students and teaching professors at the School of Computer Science and Engineering at Nanyang Technological University. The goal would be to be able to incorporate this form of learning experience throughout schools in Singapore.

4.5. Quality Attributes

HUF will choose to focus on two main attributes, namely efficiency and availability.

Since it will be a relatively simple quiz application with humble functionalities, as well as having the answers pre-defined by the users who created the game in a multiple-choice

question (MCQ) format, there should be little to no computation involved in that aspect. Therefore, logically, HUF should take up as little system resource as possible.

Being a web-based application also means that we will need to ensure it is available and in a working condition on all existing Operating System and Web browsers that are running on supported versions.

4.6. References

Tools used for game and web development:

Languages:

Python: <https://www.python.org/doc/>

SQL: https://www.w3schools.com/sql/sql_intro.asp

JavaScript: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

CSS: <https://www.w3.org/Style/CSS/Overview.en.html>

Frameworks:

React: <https://reactjs.org/>

Django Framework: <https://docs.djangoproject.com/en/3.2/>

5. Overall Description

5.1. Product Perspective

With the advent of technology, more universities in Singapore are making use of digital tools and online platforms to hold their lessons. While this supports a flexible learning environment, it still does not de-emphasise scores and grades. In fact, remote learning could end up being less engaging due to the distance and digital divide.

Students in Singaporean universities are no stranger to this as they watch the same lectures as their predecessors, but at home, on a screen, and at twice the speed. They absorb less material, interact less with each other, and have a far inferior active learning experience. Furthermore, COVID-19 has brought about many constraints in terms of moving back to in-face learning.

Therefore, the application serves as an alternative online learning platform for students that is interactive, engaging and exciting to participate in. Creating a community among the students and adopting game-based learning will not only motivate students to accomplish a specific goal, but also minimise the stress on grades.

5.2. Product Functions

The major functions for each type of users of this application are as follows:

5.2.1 Game Players:

- 5.1.3.1 Players can view a game's information, including the game name, game description, game tags and number of quizzes. This allows the player to get a better understanding of the game and helps the player decide whether this game is suitable for them.
- 5.1.3.2 Players can easily find a suitable game to play by using the search and filter function that filters games based on their tags that contain games information such as the games' category.
- 5.1.3.3 Players can play the quiz in any order at their will.
- 5.1.3.4 When a player completes a quiz, they will be able to view the leaderboard of that quiz to show how they rank against other players.
- 5.1.3.5 Players can share a game via social platforms such as Facebook to challenge their friends.

5.2.2 Game Creators:

- 5.2.2.1 Creators can create a new game which is open for participation to the community.
- 5.1.3.1 When creating a new game, the creator can specify game information including game tags, game name description, and number of quizzes.
- 5.1.3.2 For each quiz, creator may specify the number of questions, time limit, questions, answer options, and solutions.
- 5.1.3.3 For each created quizzes, the creator can view a dashboard which displays certain statistics and summary of that quiz, such as the number of people participated, and average points. This helps the game creators to have a better understanding of the players' performance in the quiz.
- 5.1.3.4 For each of the games created, the creator will be able to edit the contents of the game.
- 5.2.2.2 The game creator can add new quizzes, edit existing quizzes, and delete existing quizzes.
- 5.2.2.3 For existing quizzes, the creator can add new questions, update existing questions, and delete existing questions.

5.3. User Classes and Characteristics

The two main User Classes for HUF are game creator and game players.

5.3.1 Game Players

Game players are users who plays the games and answers the quizzes in the application. Game players use the platform to learn and pick up relevant skills in an interactive and

engaging manner as compared to traditional educational methods. In addition to the interactivity, HUF gamifies education by providing a leaderboard for each of the quizzes in the platform, as well as allowing game players to challenge their friends through Facebook. This gamification of education can help game players to accelerate their progress in their learning, which meets with HUF's main objective.

5.3.2 Game creator

Game creator uses the platform to educate the game players in the areas they are experts on. The platform allows game creators to set quizzes that contains questions of progressive difficulty levels, allowing game players to pick up knowledge in a step-by-step manner. The game creators will benefit from the platform as it provides an effective and simple way to track the students' progress through the game's dashboard, allowing game creators to understand the students' performance and progress through the different quiz levels and across different games. The students' quizzes will also be automatically checked against the correct answers, eliminating the need for game creators to manually check the answers. By providing these features on the platform, Game creators can facilitate game players' learning in the most effective and engaging way.

5.4. Operating Environment

Operating environment for the game-based learning web application is as listed below.

- Operating System: Windows, MacOS, Linux
- Database: MySQL database
- Platform: JavaScript/Python

5.5. Design and Implementation Constraints

HUF requires a web application written in JavaScript using the React library for the front-end development. The ANT design language has been employed to help create the User Interface (UI) of the system and will have to follow the design standards of the language.

For the back-end development, HUF will be using Python with the help Django framework. APIs will be developed with the Django REST API that helps connect the front and the back end. It also requires maintenance of the user information database via MySQL.

The web application must adhere to the timing and speed performances guidelines stated by the non-functional requirements in this document.

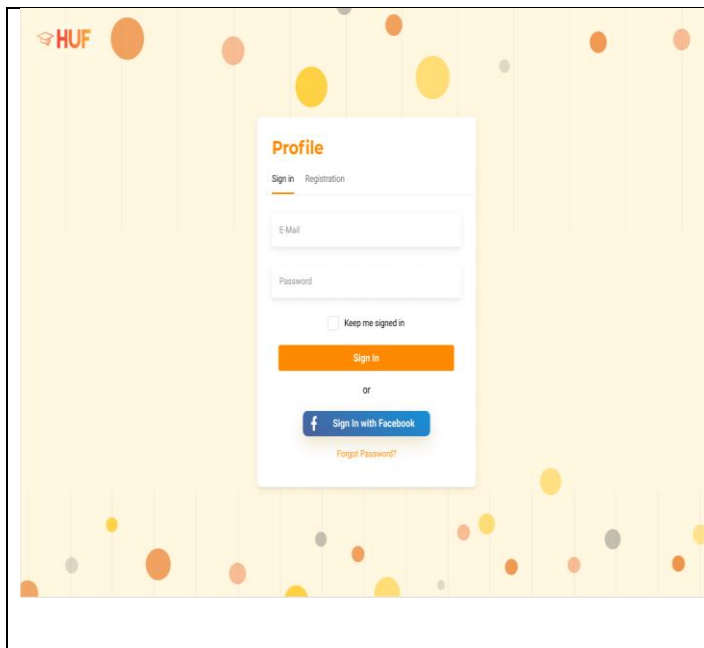
5.6. Assumptions and Dependencies

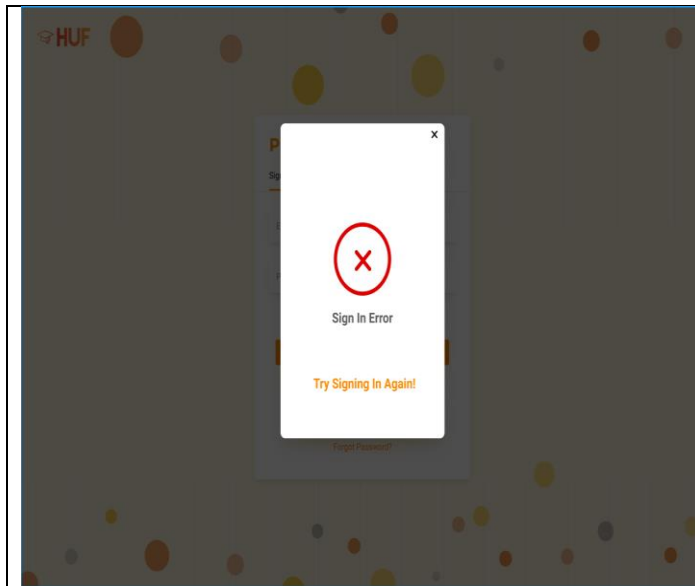
Assumptions

- Users are assumed to possess a good command of English as the main application language is available in English only.
- The web application can be accessed through any device that possess web connectivity.

6. External Interface Requirements

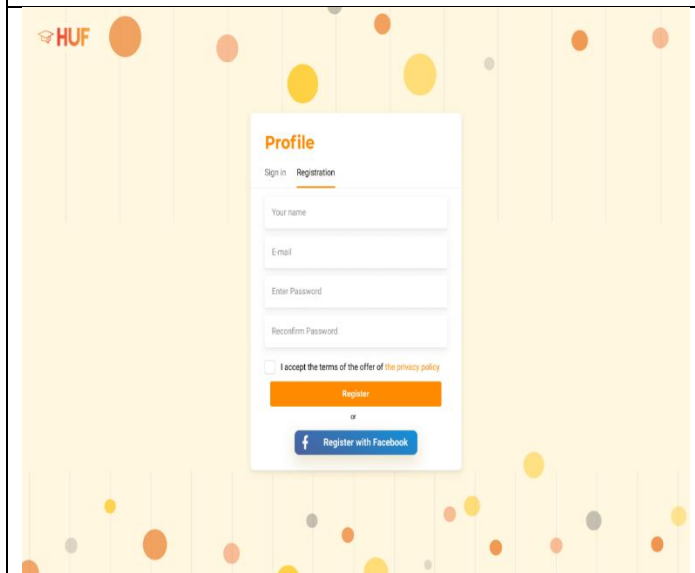
6.1 User Interfaces

	<p>Sign In Page:</p> <p>Users will be directed to this page when they first open the app. They can select sign in or registration. They have an option to keep themselves signed in. They also have an option to sign in with Facebook, and to receive a new password from the system through the "Forgot Password" option.</p>
--	--



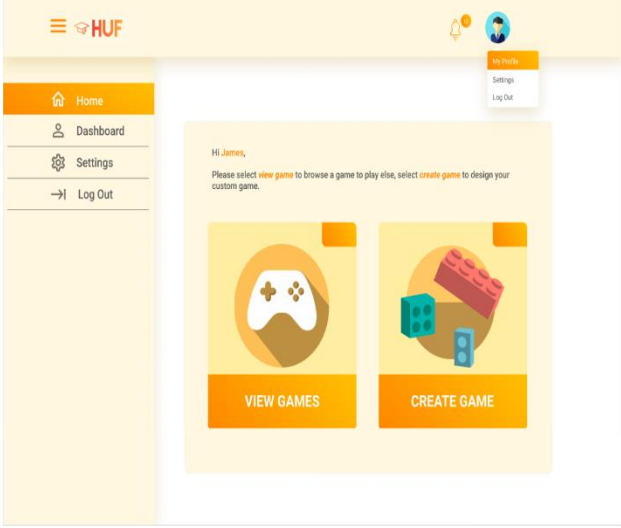
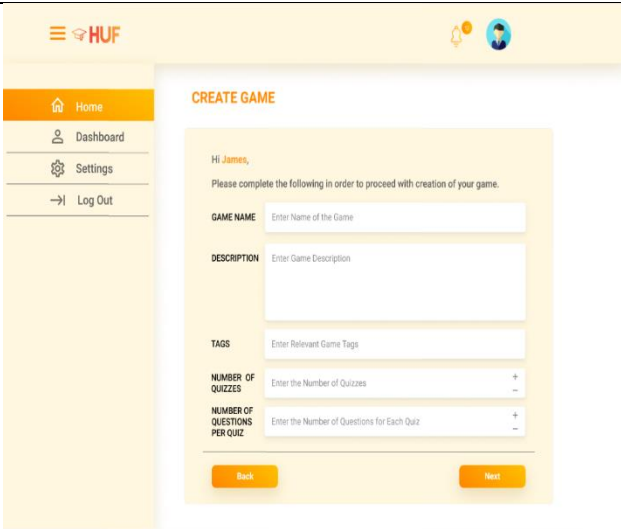
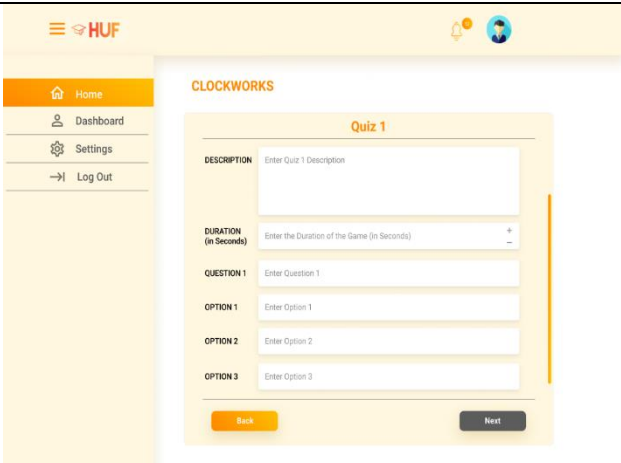
Sign In Error:

This is the page seen by the user if he/she tries to sign in with incorrect information (e.g. incorrect email or password)

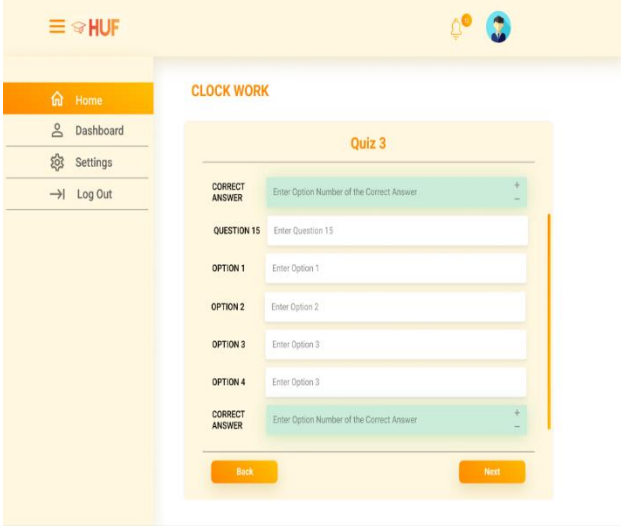
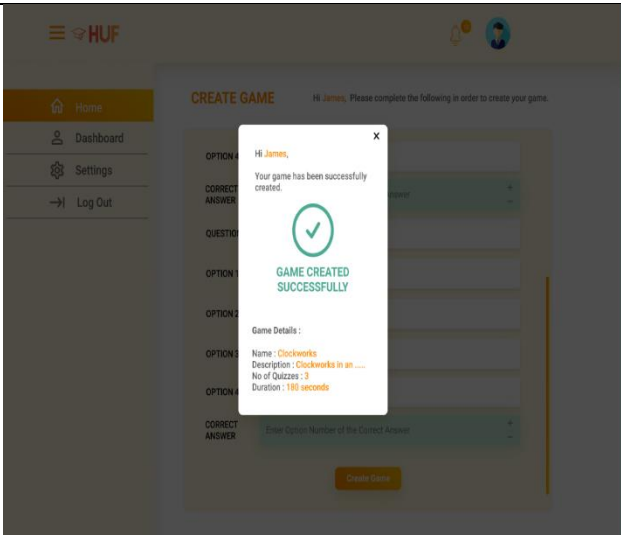


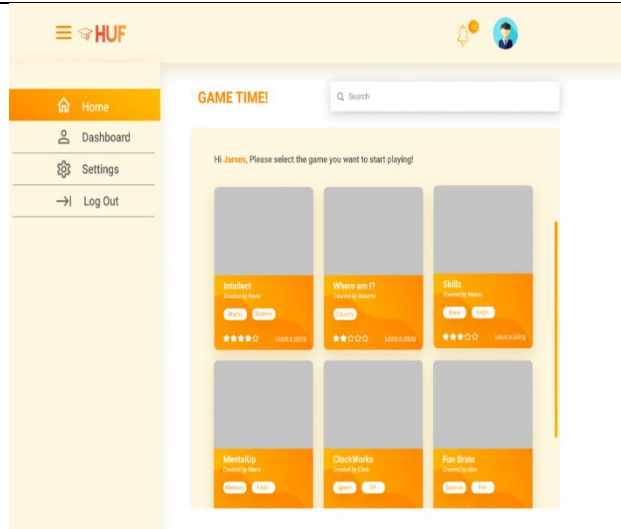
Registration Page:

This is the page where the user will insert particulars to be registered. Optionally, the user can register with their existing Facebook account.

 <p>The Home Page mockup features a sidebar with a hamburger menu icon and the 'HUF' logo. The sidebar contains links for Home, Dashboard, Settings, and Log Out. The main content area has a header with a notification bell and a user profile icon. Below the header, there's a greeting 'Hi James,' and instructions: 'Please select view game to browse a game to play else, select create game to design your custom game.' Two large buttons are displayed: 'VIEW GAMES' with a game controller icon and 'CREATE GAME' with a game controller and building blocks icon.</p>	<p>Home Page:</p> <p>This is a page where the user will be able to select their personal dashboard, view games or create a game. The user is also able to log out from this page. The user is able to click "view games" and select a game of play. The user is able to click "create game" and create a game.</p>
 <p>The Game Creation Page mockup shows a sidebar identical to the Home Page. The main content area has a header with the 'HUF' logo and a notification bell. Below the header, there's a greeting 'Hi James,' and instructions: 'Please complete the following in order to proceed with creation of your game.' The form includes fields for 'GAME NAME', 'DESCRIPTION', 'TAGS', 'NUMBER OF QUIZZES', and 'NUMBER OF QUESTIONS PER QUIZ'. At the bottom, there are 'Back' and 'Next' buttons.</p>	<p>Game Creation:</p> <p>This is where user will be able to input parameters to create a game. Upon clicking “Next” the user would be moved to the quiz question creation page. Upon clicking “Back” the user would be moved back to the home page.</p>
 <p>The Quiz Creation Page mockup shows a sidebar identical to the Home Page. The main content area has a header with the 'HUF' logo and a notification bell. Below the header, there's a greeting 'Hi James,' and instructions: 'Please complete the following in order to proceed with creation of your game.' The form includes fields for 'DESCRIPTION', 'DURATION (in Seconds)', 'QUESTION 1', 'OPTION 1', 'OPTION 2', and 'OPTION 3'. At the bottom, there are 'Back' and 'Next' buttons.</p>	<p>Quiz Creation:</p> <p>This is where the user will be able to input parameters to create a question in a quiz. Before moving to the next question creation page, the user will be prompted to enter the correct answer for this question. The user will have as many questions creation pages as the number of questions he/she created for the quiz. Upon clicking back, the user would be moved back to the previous page.</p>

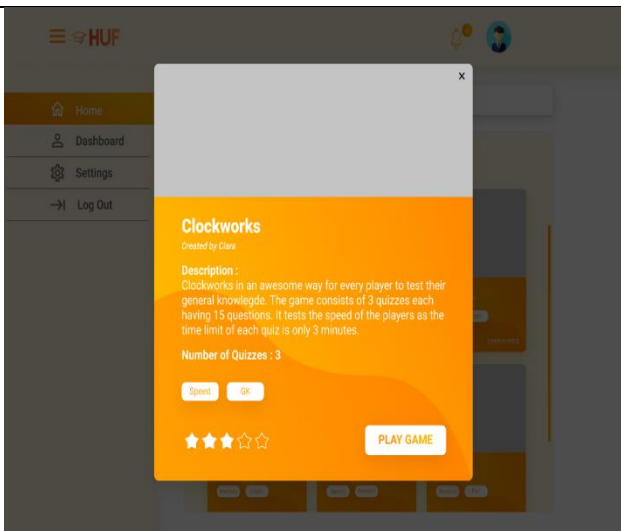
<div data-bbox="240 191 857 720"><div><div>HUF</div><div><div></div><div></div><div></div></div></div><div><div>Home</div><div>Dashboard</div><div>Settings</div><div>Log Out</div></div><div>CLOCKWORKS</div><div><div>Quiz 1</div><div><div>CORRECT ANSWER</div><div>Enter Option Number of the Correct Answer</div></div><div><div>QUESTION 15</div><div>Enter Question 15</div></div><div><div>OPTION 1</div><div>Enter Option 1</div></div><div><div>OPTION 2</div><div>Enter Option 2</div></div><div><div>OPTION 3</div><div>Enter Option 3</div></div><div><div>OPTION 4</div><div>Enter Option 3</div></div><div><div>CORRECT ANSWER</div><div>Enter Option Number of the Correct Answer</div></div><div><div>Back</div><div>Next</div></div></div></div>	<div>Quiz Creation (Quiz 1 Completion)</div>
<div data-bbox="240 814 857 1346"><div><div>HUF</div><div><div></div><div></div><div></div></div></div><div><div>Home</div><div>Dashboard</div><div>Settings</div><div>Log Out</div></div><div>CLOCK WORK</div><div><div>Quiz 2</div><div><div>DESCRIPTION</div><div>Enter Quiz 2 Description</div></div><div><div>DURATION (in Seconds)</div><div>Enter the Duration of the Game (in Seconds)</div></div><div><div>QUESTION 1</div><div>Enter Question 1</div></div><div><div>OPTION 1</div><div>Enter Option 1</div></div><div><div>OPTION 2</div><div>Enter Option 2</div></div><div><div>OPTION 3</div><div>Enter Option 3</div></div><div><div>Back</div><div>Next</div></div></div></div>	<div>Quiz Creation (Quiz 2 Completion)</div>

	<h3>Quiz Creation (Quiz 3 Completion)</h3>
	<h3>Game Creation Successful:</h3> <p>This is the display the user will see upon successful completion of the game. The user can click the “x” to remove the display.</p>



View Games Page:

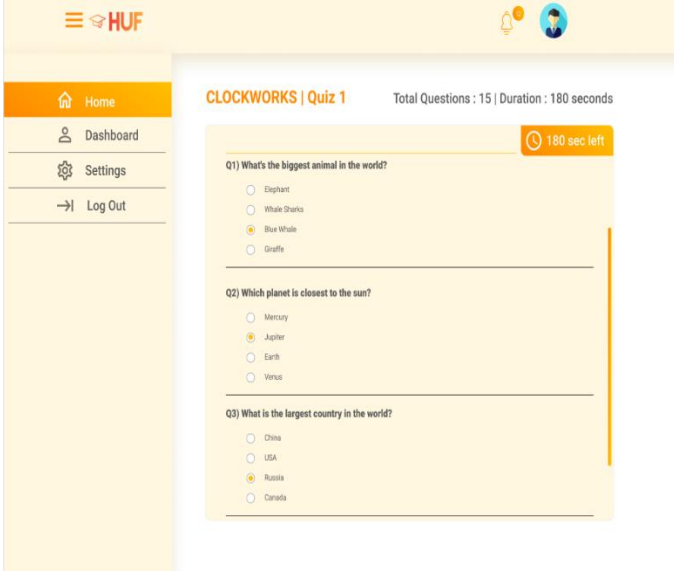
This is where the user can see a list of games that are available to be played. The ratings of the games as well as their tags are also displayed. The user is able to click a game to start it. The user is able to use the search button to filter for games that is most suited to their needs.

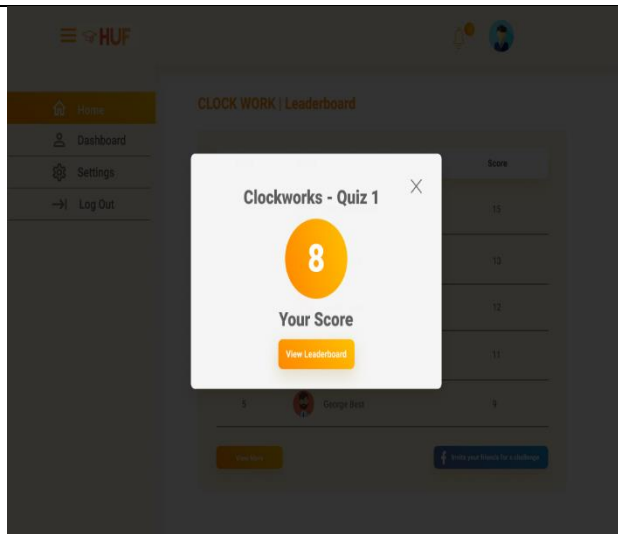


View Game Pop Up:

This is the display the user sees upon selection of a game. The user is shown the name of the game, its creator, its description, the number of quizzes inside, tags of the game and its rating. The user is given an option to play the game and “x” which would allow the user to return to the previous screen.

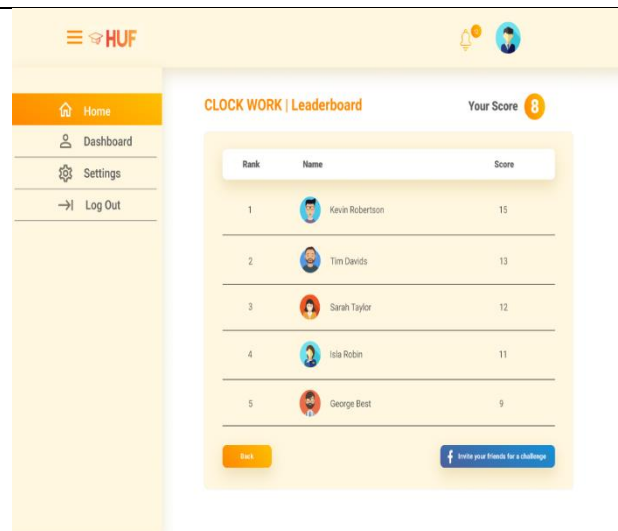
	<p>View Quizzes Page:</p> <p>This is the display the user would see upon clicking “Play Game” in the previous screen. The user would see a brighter display with a green highlight for the quizzes available to him/her. The user would see a darker display with red highlight for the quizzes that haven’t been unlocked. The user would have to complete the previous quiz in order to unlock the quiz that comes next.</p>
	<p>View Quiz Pop Up:</p> <p>This is the display the user sees upon selection of a quiz. The user is shown the name of the quiz, it’s description, the number of questions inside. The user is given an option to play the game and “x” which would allow the user to return to the previous screen. The user can click “Start Quiz” to start the quiz.</p>

	<p>Quiz Gameplay (start):</p> <p>This is the display the user would see when playing the quiz. Each question only allows one answer to be selected. The user would see the total number of questions as well as the time available for the quiz on the top right-hand side of the page. The user would see a timer above showing the time the user has left to complete the quiz. Upon completion of the timer, the quiz would record the score considering the questions the user had answered up till that time. The user would scroll down to see the remaining questions.</p>
	<p>Quiz Gameplay (Completion):</p> <p>This is the display the user would see near the end of the quiz. The user is given an option to click “Finish Quiz” to end the quiz and for the score to be tabulated. The timer above is highlighted in red when the user has a small amount of time left.</p>



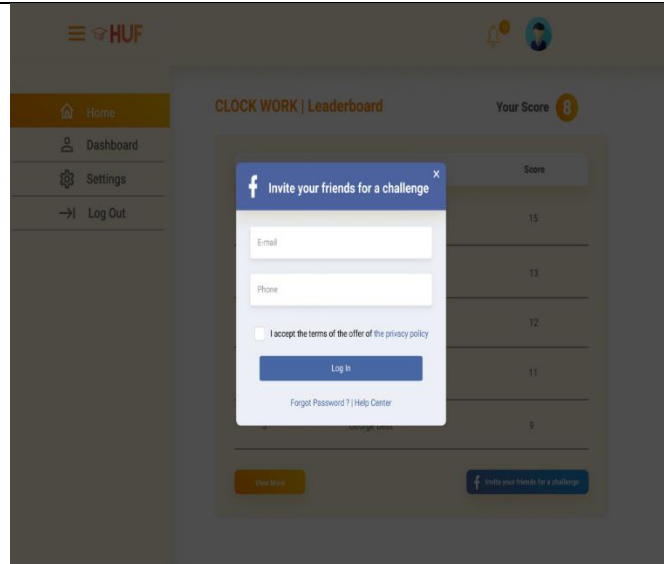
Score:

At this page, the user would see the score they received for the quiz, as well as the name of the quiz and the quiz number. The user is given an option “x” to close the display and return to the previous page. The user can also choose to view the leaderboard.



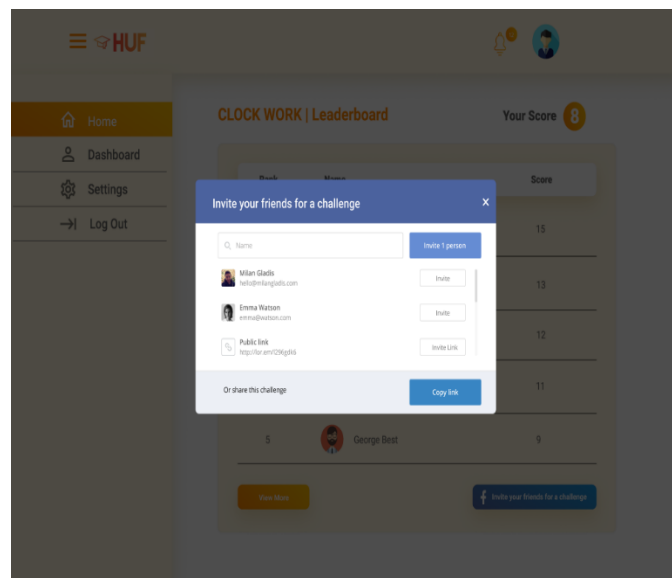
Game LeaderBoard:

At the leaderboard page, the user can see the 5 top scorers for each game, as well as their rank and name. The user’s score is also displayed at the top left-hand corner of the page. The user is given and option “Back” to return to the home page. User is also given an option to invite friends to challenge them to a game.



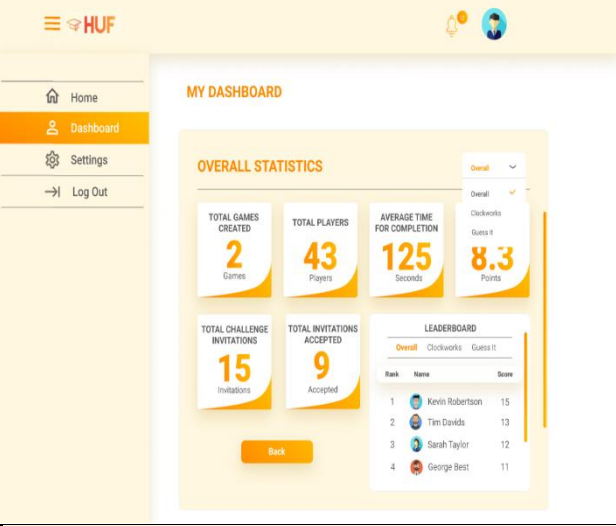
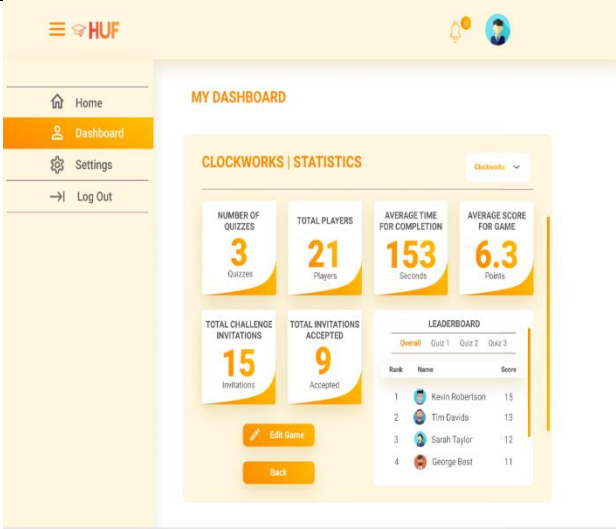
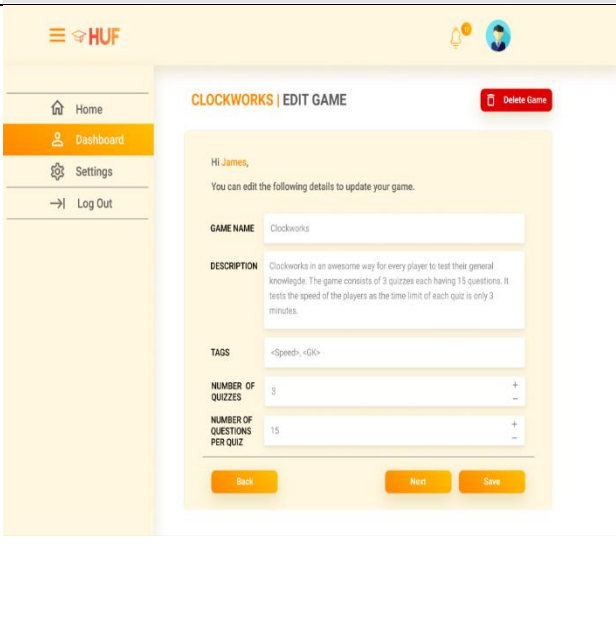
Challenge Friend (Facebook Log In):

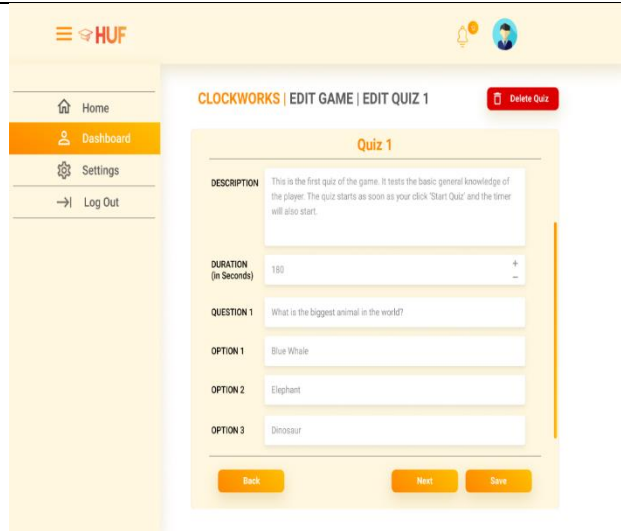
At this page, the user must log in to Facebook first before being shown the list of friends to invite. The user clicks “x” to return to the leaderboard screen.



Challenge Friend (Facebook Invite):

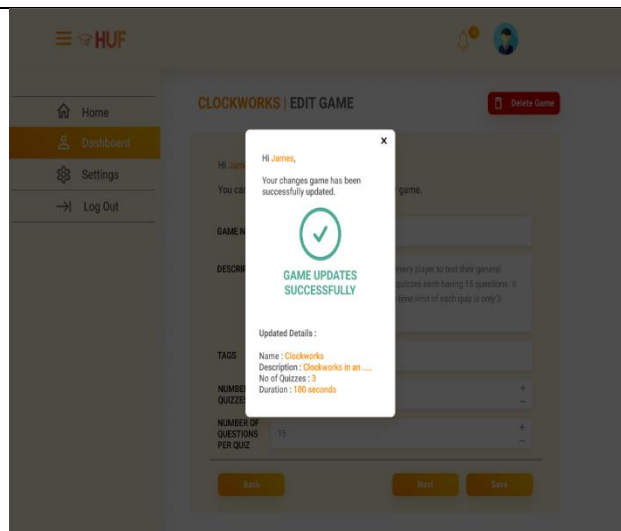
At this page, the user can scroll down a list of friends. The user can click “invite” next to the name of a friend to send the invite link through Facebook to the friend. The user is also given an option to the copy the invite link. The user clicks “x” to return to the leaderboard screen.

	<p>Dashboard (Overall):</p> <p>At this page, the user can view the leaderboard as well as other statistics for all of their games. The user can click ‘Back’ to return to the previous screen. The user can select, at the top left-hand corner, to view statistics for specific games. At the leaderboard display, the user can select the view the leaderboard for different games.</p>
	<p>Dashboard (Game):</p> <p>The user can view the leaderboard as well as other statistics for all of the quizzes within the game. The user can click ‘Back’ to return to the previous screen. The display at the top left-hand corner shows the name of the current game the user is viewing. At the leaderboard display, the user can select the view the leaderboard for different quizzes within the game.</p>
	<p>Edit Game:</p> <p>The user is able to edit the parameters of the game here. The user can click “save” to save edited parameters of the game. The user given an option to Delete Game, which would delete all the data of the game.</p>



Edit Quiz:

The user is able to edit the parameters of the Quiz here. The user can click “save” to save edited parameters of the Quiz. The user given an option to Delete Quiz, which would delete all the data of the Quiz. The user can click “Next” to view the edit screen for the next quiz in the game.



Updated Game Successfully:

The user is showed the details that were updated for the game. User can click “x” to return to the previous screen.

6.2 Hardware Interface

A wireless Network Interface Card (WNIC) or Network Interface Card (NIC) with LAN port is necessary.

6.3 Software Interface

The minimum system requirement for the user is stated in the table below:

Desktop	Minimum Requirements
Operating Systems	Windows 7 SP1+ macOS 10.12+ Ubuntu 16.04+

The web application is required to run on all frequently used browsers such as Google Chrome, Mozilla Firefox, Safari and Microsoft Edge.

6.4 Communications Interface

Huf requires an internet connection to access the MySQL database that is deployed on Amazon Relational Database Service (Amazon RDS). This includes updating data or querying for data which requires internet connection. Users also need internet connection to connect to the website to play the game.

7. Functional Requirements

7.1 System Functionality to be Performed

7.1.1. Account Registration

Description: Users must be able to create an account for the system.

Priority: High

7.1.1.1. The user must be able to access the registration page from the login page (default page upon opening the website), using a button, 'Register'.

7.1.1.2. The system must prompt the user for his/her credentials during registration.

7.1.1.2.1. User shall be able to input his/her name.

7.1.1.2.2. User shall be able to input their email address when registering for an account.

- 7.1.1.2.3. User shall be able to input in their desired password combination when registering for an account.
- 7.1.1.2.4. User shall be able to input their confirmed password during account registration.
- 7.1.1.3. The system must be able to validate that all the fields have been filled up.
 - 7.1.1.3.1. System must validate the user's email address is filled up.
 - 7.1.1.3.2. System must validate the user's password is filled up.
 - 7.1.1.3.3. System must validate the user's full name is filled up.
 - 7.1.1.3.4. If any of the corresponding fields are not filled, a corresponding error message shall be displayed by the system.
- 1.1.1.2. The system must be able to check the validity of the fields entered by the user.
 - 7.1.1.4.1. The system must validate that the password entered by the user is at least 8 characters long.
 - 7.1.1.4.2. The system must be able to validate that the email entered is not linked to an existing account.
 - 7.1.1.4.3. If validation fails, an error message must be displayed to inform the user of the corresponding error.
- 7.1.1.4. The system must notify the user upon successful account creation.
 - 7.1.1.5.1. The system must display an in-app message on the website to inform the user of successful account creation.
- 7.1.1.5. The system must allow the user to register an account with their Facebook account.
 - 7.1.1.6.1. The email address used to register the Facebook account must not be the same as an existing email in the database.

7.1.2. Login

Description: Users must be able to log in to the system.

Priority: High

- 7.1.2.1. The system must provide a login entry for all users.
- 7.1.2.2. The system must prompt the user for his/her credentials during login.
 - 7.1.2.2.1. The system shall provide an input field for the user to enter his/her username.
 - 7.1.2.2.2. The system shall provide an input field for the user to enter his/her password.
- 7.1.2.3. The system must be able to validate the credentials entered by the users.
 - 7.1.2.3.1. The system must validate that the username field is filled.
 - 7.1.2.3.2. The system must validate that the password field is filled.
 - 7.1.2.3.3. If any of the corresponding fields are not filled, a corresponding error message shall be displayed by the system to inform the user that either of the fields must not be left empty.

- 7.1.2.4. The system must be able to validate the username and the password against the database.
 - 7.1.2.4.1. The system shall validate that the username entered is a registered user.
 - 7.1.2.4.1.1. If the username entered is not valid, the application must display an error message.
 - 7.1.2.4.2. The system must validate if the password entered matches the username entered in the database.
 - 7.1.2.4.2.1. If the password entered is not valid, an error message shall inform the user that the password entered is wrong.

7.1.3. Forget Password

Description: Users must be able to recover their account if they forget their passwords.

Priority: Low

- 7.1.3.1. The user must be able to access the forget password page from the login page (default page upon open the website) using a button, 'Forget Password'.
- 7.1.3.2. User shall be able to recover their password into the setting page.
- 7.1.3.3. The system shall automatically generate a new password for the user and save it in the database.
 - 7.1.3.2.1. User shall be able to receive new password sent automatically by our system through their registered email address.
 - 7.1.3.2.2. User shall be able to log in with the newly received password

7.1.4. Display Home Page

Description: Displays all games available and a search bar where users can search or filter for their desired games.

Priority: High

- 7.1.4.1. The system shall display 2 main functions/ buttons displayed side by side (i.e., 2x2 grid)– View Game and Create Game.
 - 7.1.4.1.1. The system must allow the user to navigate to the respective screens when the respective buttons are created.
 - 7.1.4.1.2. In the first function, which is the 'View Games' button, the system shall load the View Game age when the user taps on the view game button
 - 7.1.4.1.3. In the second function, which is the 'Create a Game' button, the system shall load the Create Game page when the user taps on the create game button.
- 7.1.4.2. The system shall have a logout button which can be found in the navigation header that will log the user out when clicked.

7.1.5. Game Creation

Description: User can create a game as well as the quizzes within the game.

Priority: High

- 7.1.5.1. The system must allow the user to create a game.
- 7.1.5.2. The system must allow the user to create up to 5 quizzes in a ranked order (Easiest to Hardest).
- 7.1.5.3. The system must allow the user to specify the following in his game:
 - 7.1.5.3.1. Game name
 - 7.1.5.3.2. Game description
 - 7.1.5.3.3. Number of quizzes
 - 7.1.5.3.4. Number of questions for each quiz
 - 7.1.5.3.5. Game Tag
 - 7.1.5.3.6. The system must ensure that if any of the corresponding fields are not filled, a corresponding error message shall be displayed by the system and the user shall not be able to proceed until all fields have been filled in.
- 7.1.5.4. Upon successful game creation, the system shall redirect the users to the quiz creation page to create the quizzes.
- 7.1.5.5. The system must display the game in the main lobby of the system
- 7.1.5.6. The system must allow all users to access the game.

7.1.6. Quiz Creation

Description: User can create up to 5 quizzes.

Priority: High

- 7.1.6.1. The system must allow the user to create the respective number of quizzes and questions based on the users input during game creation.
- 7.1.6.2. The system must allow the users to create questions, its answer options (MCQ), and input its respective answer/solution.
 - 7.1.6.2.1. The answers can be stored in the database.
- 7.1.6.3. The system must allow users to set a time limitation for each quiz.
 - 7.1.6.3.1. The system must display the time in a MM: SS format.
- 7.1.6.4. The system must allow users to select a difficulty level for the quiz depending on the difficulty level of their previous quiz (i.e., Quizzes must be created in a ranked order – Easiest to Hardest)
- 7.1.6.5. The system must allow the user to post the created questions for other users in the game to try.

7.1.7. View Games Page

Description: Displays all relevant information about all the available games.

Priority: High

- 7.1.7.1. The system shall load the view games page when the user taps on the View Game button in the Homepage.
 - 1.1.1.1.1.1. The system shall display a grid view of all the available games on the platform. The user shall be able to select the game they would like to play upon tapping into one of the game buttons.
 - 1.1.1.1.1. The system shall display a pop up which contains the details of the game when one of the game buttons is being tapped on.
 - 7.1.7.1.2.1. The selected game page shall display the following information:
 - Creator
 - Game name
 - Game tag
 - Game description
 - Number of quizzes
 - 1.1.1.1.1.2. The pop up of the selected game shall display the 'Play' button where the user could tap on the 'Play' button.
 - 1.1.1.1.1.2.1. Upon tapping the 'Play' button, the system shall redirect the user to the View Quiz Page.
- 7.1.7.2. The view game page shall contain a search bar that allows users to search for a game.
 - 1.1.1.2.1. The user must be able to search for the game created using keywords or tags associated with the game, this will show a grid view of all relevant tagged games.

7.1.8. View Quiz Page

Description: Displays the quiz as a pop up selected by the user.

Priority: High

- 7.1.8.1. The system shall display an array of all the quizzes available in the selected game.
- 7.1.8.2. The system shall load the selected quiz as a pop up to the user.
 - 7.1.8.1.1. The system must display a 'Start Quiz' button to trigger the start of the quiz.
 - 7.1.8.1.1.1. The user must be able to tap on the 'Start Quiz' button to trigger the quiz gameplay.
 - 7.1.8.1.2. The system must display 4 main information related to the selected quiz as follows:

- a. Name of Quiz
- b. Description of Quiz
- c. Number of Questions
- d. Time limit of Quiz

7.1.9. Quiz Gameplay

Description: Describes how the player interacts with the selected quiz.

Priority: High

- 7.1.9.1. The system shall start the selected quiz with the pre-set time limit determined by the creator upon a player clicking on the 'Start Quiz' button.
 - 7.1.9.1.1. The system must accurately display each question followed by the multiple-choice set of answers to the player during the gameplay.
 - 7.1.9.1.2. The system must consider that the game has ended if the user clicks on the "Finish Quiz" button at the bottom after the last question.
- 7.1.9.2. The system must consider that the game has ended if the time limit expires.
- 7.1.9.3. The system shall load the leader board page at the end of the quiz gameplay and show user's score.
 - 7.1.9.3.1. The system must allow the user to be redirected back to the game page after user clicks on 'Back' button at the leader board page.
- 7.1.9.4. The system must allow users to exit the quiz at any time if they choose to, by pressing the "Back" button.
- 7.1.9.5. For users who have finished the quiz, if they click on the same quiz again and tries to submit the quiz again, they will receive a 'You have already completed this quiz.' message and get redirected back to the games page.

7.1.10. View Leader Board

Description: All players who have finished the quiz must be able to visually view the top 5 players of a quiz and their own score.

Priority: High

- 7.1.10.1. System must display a leader board unique to each quiz.
 - 7.1.10.1.1. System will display the players' score.
 - 7.1.10.1.2. The system will calculate the player's score based on the time they take to answer each quiz question correctly.
 - 7.1.10.1.3. System must display the scores of the top 5 players.
 - 7.1.10.1.4. System must display an 'Back' button that allows user to return to the previous screen.
- 7.1.10.2. The system must show a 'Challenge' button.

7.1.11. Challenge

Description: Players can send a challenge invite to their friends on Facebook.

Priority: Medium

- 7.1.11.1. System must navigate player to a pop-up tab login screen of player's Facebook account.
- 7.1.11.2. System must allow player to log in into Facebook account.
 - 7.1.11.2.1. System must navigate player to drop down list of Facebook friends.
 - 7.1.11.2.2. System must allow player to select multiple friends and click 'Send Invite.'
 - 7.1.11.2.3. System must send invites to selected friends.
- 7.1.11.3. System must provide a "Back" button for player to return to previous screen.

7.1.12. View Dashboard

Description: Creators will be able to view the overall statistics of the games they created as well as edit the content of the game from the 'Dashboard' tab at the side.

Priority: High

- 7.1.12.1. System must display a dashboard.
- 7.1.12.2. The dashboard must show all the games the creator has created.
- 7.1.12.3. The system shall load the selected game dashboard upon the selection of the specified game by the creator.
 - 7.1.12.3.1. Dashboard must contain statistics about the game and all quizzes inside.
 - a. Complete list of scores for all individuals who played the game.
 - b. Average time taken to complete quiz.
 - c. Average score.
 - 7.1.12.3.2. Dashboard must display an 'Edit Game' button.
 - 7.1.12.3.3. Dashboard must display an 'Edit Quiz' button.
 - 7.1.12.3.4. Dashboard must display a 'Back' button that allows creator to return to the 'Dashboard' page.

7.1.13. Edit Game

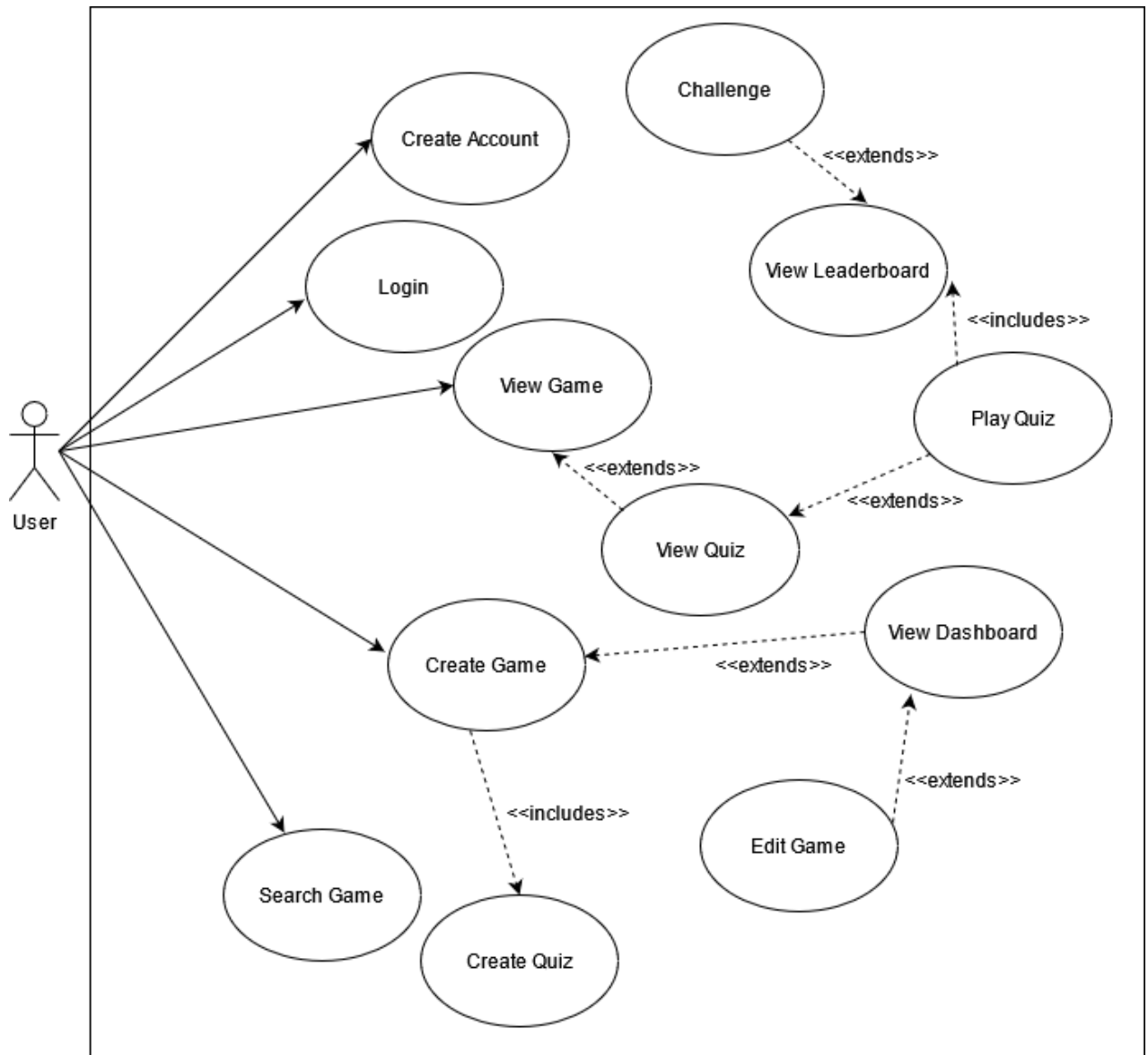
Description: Creator can modify elements of the game, such as name, description, and number of quizzes or delete the game or delete it.

Priority: Medium

- 7.1.13.1. The system must allow the creator to edit the game once the game is created.

- 7.1.13.1.1. The system must allow the option for the creator to edit the game name and description.
- 7.1.13.1.2. The system must allow the option for the creator to delete the game.
- 7.1.13.2. The system must allow the creator to edit the quizzes within the game.
 - 7.1.13.2.1. The system must allow the option for the creator to add or delete quizzes.
 - 7.1.13.2.2. The system must allow the option for the creator to edit existing quizzes.
 - 7.1.13.2.2.1. The system must allow the creator to change the original questions and answers.
 - 7.1.13.2.2.2. The system must allow the creator to add additional questions in the quiz.
 - 7.1.13.2.2.3. The system must allow the creator to change the time limit of the quiz.
- 7.1.13.3. The system must display an “Updated Game Successfully” message detailing the changes the creator made to the game.

7.2 Use Case Diagram



7.3 Use Case Description

Use Case ID:	1		
Use Case Name:	Create Account		
Created By:	Ng Chi Hui	Last Updated By:	Ng Chi Hui
Date Created:	02/09/2021	Date Last Updated:	02/09/2021

Actor:	User (Initiating Actor)
Description:	First time user registers for a new account.
Preconditions:	1. User device must be connected to the Internet.
Postconditions:	1. Successful account creation will redirect users to the login page. 2. The system must give access to user who successfully created an account.
Priority:	High
Frequency of Use:	Very Low – user will only need to create an account once.
Flow of Events:	1. User opens the website. 2. User clicks on the button ‘Register’. 3. Player keys in their credentials – email and password. 4. Player successfully registers for an account. 5. User redirected to the login page for login.
Alternative Flows:	N/A
Exceptions:	EX 1: User’s password is not strong enough. 1. System prompts user with the specific password requirements (e.g., Alphanumeric, special characters etc).
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Use Case ID:	2		
Use Case Name:	Login		
Created By:	Teng Sok Ee	Last Updated By:	Teng Sok Ee
Date Created:	29/08/2021	Date Last Updated:	29/08/2021

Actor:	User (Initiating Actor)
Description:	<p>The user can login to the website with their valid username and password at the login page to gain access to the website materials.</p> <p>This use case is a precondition for all the other use cases.</p>
Preconditions:	<ol style="list-style-type: none"> 1. User must have a registered account. 2. Users' device is connected to the internet.
Postconditions:	System must give access to user who successfully login or deny access to user who has an unsuccessful login. Successful login will redirect users to the homepage while unsuccessful login will not be redirected.
Priority:	High
Frequency of Use:	High
Flow of Events:	<ol style="list-style-type: none"> 1. User goes to login page. 2. User enters their registered email address and password. 3. System retrieves data using email address provided from database. 4. System validates the email address and password. 5. Account is verified and the user is granted access to the application. 6. System redirects users to the homepage.
Alternative Flows:	<p>AF - S4. If user enters incorrect email address/password.</p> <ol style="list-style-type: none"> 1. The system will show the message "Try Signing in Again". 2. Return to step 2.
Exceptions:	N/A
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Use Case ID:	3		
Use Case Name:	Create Game		
Created By:	Zoe	Last Updated By:	Ashwin
Date Created:	02/09/2021	Date Last Updated:	22/10/2021

Actor:	User (Initiating Actor)
Description:	System will create a new game based on user's input.
Preconditions:	<ol style="list-style-type: none"> 1. User must be logged in. 2. Users' device must be connected to the internet.
Postconditions:	System will create a game under the user's name.
Priority:	High
Frequency of Use:	High
Flow of Events:	<ol style="list-style-type: none"> 1. User clicks on the 'Create Game' button on the Game page. 2. Users will be navigated to the game creation page. 3. User will fill up information of the game: <ul style="list-style-type: none"> - Game name - Game description - Number of Questions per quiz - Number of quizzes - Game tag 4. User will need to add their own quizzes into the game. 5. User clicks on 'Next', and the game will be created.
Alternative Flows:	AF-S3: <ol style="list-style-type: none"> 1. User clicks the "Back" button 2. User is directed to the home page
Exceptions:	EX 1: Missing fields or invalid inputs for the game creation form. <ol style="list-style-type: none"> 1. System prompts user with the specific input field requirements (e.g., Alphanumeric, special characters etc).
Includes:	Use Case 12: 'View Dashboard'
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Use Case ID:	4		
Use Case Name:	Edit Game		
Created By:	Teng Sok Ee	Last Updated By:	Sim Tian Quan
Date Created:	04/09/2021	Date Last Updated:	10/11/2021

Actor:	User (Initiating Actor)
Description:	User can edit the details of the game.
Preconditions:	<ol style="list-style-type: none"> 1. User needs to be logged in. 2. User needs to be connected to the internet. 3. There must exist at least 1 game created in the system. 4. User must be the creator of a game.
Postconditions:	System will successfully update the edited game.
Priority:	Low
Frequency of Use:	Medium
Flow of Events:	<ol style="list-style-type: none"> 1. User will click on 'Edit Game' from the statistics page that is shown after clicking on a particular game from 'Dashboard' page. 2. System will redirect user to a page where user is allowed to make changes to the selected game. 3. User can carry out the following actions: <ul style="list-style-type: none"> - Edit name of the game - Edit description of the game - Edit Number of Quizzes in the game - Edit Number of Questions per quiz in the Game - Delete game - Edit Quiz - Add Quiz - Delete Quiz 4. After editing, user can click on 'Confirm changes' button. 5. System will show a message to confirm changes made. 6. User clicks 'Confirm'. 7. System will update the game.
Alternative Flows:	AF – S3: <ol style="list-style-type: none"> 1. User clicks on 'Back' button. 2. The system will not record the changes made. 3. System will redirect user back to the previous page.
Exceptions:	EX 1: Missing fields or invalid inputs for the game creation form. <ol style="list-style-type: none"> 1. System prompts user with the specific input field requirements (e.g., Alphanumeric, special characters etc).

Includes:	Use Case 12: ‘View Dashboard’
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Use Case ID:	5		
Use Case Name:	Create Quiz		
Created By:	Song Yu	Last Updated By:	Nikita
Date Created:	29/08/2021	Date Last Updated:	22/10/2021

Actor:	Creator (Initiating Actor)
Description:	When the 'Create Quiz' button on the Create Game page is clicked, the creator can create a new quiz in that game.
Preconditions:	<ol style="list-style-type: none"> 1. User needs to be logged in. 2. Users' devices must be connected to the internet. 3. User must be the creator of the game which this quiz is in.
Postconditions:	System will create a quiz under the creator's name. Quiz that is successfully created will be shown on the Game page.
Priority:	High
Frequency of Use:	Medium
Flow of Events:	<ol style="list-style-type: none"> 1. Creators click on the 'Next' button in the Game creation page. 2. Creators will be navigated to the quiz creation page. 3. Creators will fill out the basic information of the quiz, including quiz title, quiz description, difficulty, time limit. 4. Creators click 'Next' and will be navigated to question creation page. 5. Creators can add multiple choice questions, answer options, and solution/correct answer to each question. 6. Creators click on 'Confirm'. 7. System will create the new quiz in the current game successfully. 8. Creators will be redirected to the home page.
Alternative Flows:	<p>AF-S3:</p> <ol style="list-style-type: none"> 1. Creators click on the 'back' button. 2. Creators will be redirected to the Game page. <p>AF-S4:</p> <ol style="list-style-type: none"> 1. Creator does not fill out all the required information. 2. System will prompt creator to fill in the empty inputs.
Exceptions:	N/A
Includes:	Use Case 3: 'Create Game'

Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Use Case ID:	6		
Use Case Name:	View Game		
Created By:	Nikita	Last Updated By:	Teng Sok Ee
Date Created:	04/09/2021	Date Last Updated:	13/11/2021

Actor:	Player (Initiating Actor)
Description:	Displays all available games to the player.
Preconditions:	<ol style="list-style-type: none"> 1. player must be logged in. 2. Player's device must be connected to the internet.
Postconditions:	System will display all the available games to the player.
Priority:	High
Frequency of Use:	High
Flow of Events:	<ol style="list-style-type: none"> 1. Player will click on 'Play Game' button in the Home Page. 2. System will show all games available 3. User clicks on a game, system will show a pop up of the game description 4. Player clicks on 'Play Game' 5. User will be redirected to the 'Games Page' where all available games are displayed to the user.
Alternative Flows:	AF-S3: <ol style="list-style-type: none"> 1. Player clicks on the cross ("x"), prompting the player to go back to the page where all the games are displayed.
Exceptions:	N/A
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Use Case ID:	7		
Use Case Name:	View Quiz		
Created By:	Teng Sok Ee	Last Updated By:	Teng Sok Ee
Date Created:	04/09/2021	Date Last Updated:	04/09/2021

Actor:	User (Initiating Actor)
Description:	Displays all relevant information of the quiz such as name, description, number of questions, and time limit.
Preconditions:	<ol style="list-style-type: none"> 1. User must be logged in. 2. Users' device must be connected to the internet.
Postconditions:	System will open the quiz information that the user has selected and display all relevant information regarding the quiz as a pop up.
Priority:	High
Frequency of Use:	High
Flow of Events:	<ol style="list-style-type: none"> 1. User clicks on the specific quiz with name "XXX". 2. System will show a pop up showing the information of the quiz. <ul style="list-style-type: none"> - Quiz Name - Quiz Description - Number of questions - Time limit 3. User can click on 'Start Quiz'. 4. User can also click on 'View Leaderboard'
Alternative Flows:	AF – S3: User close or click outside of the pop-up message. <ol style="list-style-type: none"> 1. The pop-up message will close, and user will remain in the game page which shows all the quizzes.
Exceptions:	N/A
Includes:	Use Case 6: 'View Game'
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Use Case ID:	8		
Use Case Name:	Play Quiz		
Created By:	Teng Sok Ee	Last Updated By:	Teng Sok Ee
Date Created:	03/09/2021	Date Last Updated:	03/09/2021

Actor:	User (Initiating Actor)
Description:	Users select a quiz in a game to play.
Preconditions:	<ol style="list-style-type: none"> 1. User needs to be logged in. 2. Users' devices must be connected to the internet. 3. User must clear lower difficulty quizzes before attempting higher difficulty quizzes (if applicable).
Postconditions:	System will allow users to play the quiz they have selected.
Priority:	High
Frequency of Use:	High
Flow of Events:	<ol style="list-style-type: none"> 1. User starts the quiz. 2. System shows the questions from the quiz. 3. User answers all the questions. 4. User clicks on 'Finish Quiz.' 5. System will be redirected to leaderboard page and shown users' score, the leader board of the quiz, "Invite your friends for challenge" button, and top 5 scorers for the quiz. 6. User clicks on 'Back' to return to game page.
Alternative Flows:	<p>AF – S3: User clicks on 'Back' button.</p> <ol style="list-style-type: none"> 1. User will be redirected back to game page. <p>AF – S3: Time limit is reached</p> <ol style="list-style-type: none"> 1. Pop up message will show 'Times Up!' 2. User will be redirected to the leaderboard page
Exceptions:	N/A
Includes:	Use Case 7: 'View Quiz'
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Use Case ID:	9		
Use Case Name:	View Leader Board		
Created By:	Ashwin	Last Updated By:	Tian Quan
Date Created:	04/09/2021	Date Last Updated:	22/10/2021

Actor:	User (Initiating Actor)
Description:	The players will be able to view the top 5 scoring individuals after the end of the quiz.
Preconditions:	<ol style="list-style-type: none"> 1. User has internet connection. 2. User is logged in. 3. User has just completed a quiz or clicked the ‘View leaderboard button in the quiz description pop up’.
Postconditions:	System will display top 5 players for that quiz.
Priority:	High
Frequency of Use:	High
Flow of Events:	<ol style="list-style-type: none"> 1. Player has just finished the quiz or clicked the ‘View leaderboard button in the quiz description pop up’. 2. System calculates the player’s score (if any). 3. System retrieves leader board. 4. System displays leader board of top 5 players 5. System displays “Challenge” button.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	Use Case 8: ‘Play Quiz’
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Use Case ID:	10		
Use Case Name:	Challenge		
Created By:	Vishal Raj	Last Updated By:	Tian Quan
Date Created:	04/09/2021	Date Last Updated:	22/10/2021

Actor:	User (Initiating Actor)
Description:	User selects to send an invitation to challenge a friend through Facebook.
Preconditions:	<ol style="list-style-type: none"> 1. Users' devices must be connected to the internet. 2. User needs to be logged in. 3. User needs to complete a quiz under a selected game.
Postconditions:	System allows the users to send invite link to challenge friends for the quiz through Facebook.
Priority:	High
Frequency of Use:	High
Flow of Events:	<ol style="list-style-type: none"> 1. User clicks on the 'Challenge' button in the 'Leader Board' page. 2. User will be navigated to a pop-up modal tab. 3. User login into Facebook account. 4. User will select friends to send challenge invitation. 5. User clicks on 'Send Invite'. 6. System will send invitations successfully. 7. User will be redirected back to 'Leader Board' page.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	Use Case 9: 'View Leader Board'
Special Requirements:	N/A
Assumptions:	<ol style="list-style-type: none"> 1. User's account is not created using Facebook. 2. The user that is being challenged has a registered account on the platform. 3. The user has an existing Facebook account.
Notes and Issues:	N/A

Use Case ID:	11		
Use Case Name:	Search Game		
Created By:	Daniel	Last Updated By:	Daniel
Date Created:	03/09/2021	Date Last Updated:	03/09/2021

Actor:	User (Initiating Actor)
Description:	Users can search for existing games based on name or game tags.
Preconditions:	<ol style="list-style-type: none"> 1. User is in the 'View Games' page. 2. User needs to be logged in. 3. User needs to be connected to the internet.
Postconditions:	System will display the results of the searched name or game tag.
Priority:	Low
Frequency of Use:	Medium
Flow of Events:	<ol style="list-style-type: none"> 1. User types in name or tag of the game in the search bar. 2. User hits on 'Enter'. 3. System shows the result of the searched name or game tag.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Use Case ID:	12		
Use Case Name:	View Dashboard		
Created By:	Tian Quan	Last Updated By:	Sim Tian Quan
Date Created:	04/09/2021	Date Last Updated:	10/11/2021

Actor:	User (Initiating Actor)
Description:	User can view the details from a created game.
Preconditions:	<ol style="list-style-type: none"> 1. User needs to be logged in. 2. User needs to be connected to the internet.
Postconditions:	System will display the details of the game to user.
Priority:	Medium
Frequency of Use:	High
Flow of Events:	<ol style="list-style-type: none"> 1. User clicks on 'Dashboard'. 2. System will display the game that was created by the user. 3. User can select a game, and the related statistics for that game will be shown to the user. 4. System will give the User options to "Edit Game" or "Edit Quiz".
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	Use Case 3: 'Create Game' Use Case 4: 'Edit Game'
Special Requirements:	N/A
Assumptions:	There must exist at least 1 game created by the user.
Notes and Issues:	N/A

8. Non-Functional Requirements

8.1 Performance Requirements

Performance requirements specify how well the system performs certain functions under specific conditions, its resource allocation to the memory for usage of algorithms and its response time to the user.

8.1.1. Create account

- 8.1.1.1. The website must be able to load the login page within 3 seconds after opening the website.
- 8.1.1.2. When the email entered for creating an account is invalid, the field must be marked in red within 3 seconds after clicking the create account button.
- 8.1.1.3. When creating an account, if all account details are correctly entered, the account needs to be created and the user needs to land on the login page within 3 seconds.

8.1.2. Login

- 8.1.2.1. When the login details are correctly entered, the user must be able to reach the home page within 3 seconds after submitting the login details.
- 8.1.2.2. When there are missing fields in the login details, the field must be highlighted in red within 3 seconds after the user submits the login details.
- 8.1.2.3. When there is a failed login attempt, the error message "Incorrect email/password" must pop out within 3 seconds after the user submits the login details.
- 8.1.2.4. On the homepage, the assets must be displayed within 2 seconds as the user scrolls through the homepage.

8.1.3. Game Creation

- 8.1.3.1. When entering incomplete or invalid information, the problematic field must be marked in red within 3 seconds after submitting game creation request.
- 8.1.3.2. After submitting the request to create a game with all valid details entered, the game should appear on the game page within 5 seconds.
- 8.1.3.3. After submitting the request to create a game with all valid details entered, user will be redirected to quiz creation page within 3 seconds.

8.1.4. Quiz Creation

- 8.1.4.1. When entering incomplete or invalid information, the problematic field must be marked in red within 3 seconds after submitting quiz creation request.
- 8.1.4.2. After submitting the request to create a quiz with all valid details entered, the quiz should appear on the quiz page of the game within 5 seconds.
- 8.1.4.3. After submitting the request to create a quiz with all valid details entered, users should be redirected to quiz page of the game within 3 seconds.

8.1.5. Playing the quiz

- 8.1.5.1 The questions and options must be displayed within 3 seconds after starting the quiz.
- 8.1.5.2 The Quiz timer must be displayed within 3 seconds after starting the quiz.
- 8.1.5.3 When the timer runs out, the 'Times Up!' pop out message should appear within 1 second after the time runs out.

8.1.6. Viewing Leader Board

- 8.1.6.1. When the user clicks to view the leader board, the leader board showing the top 5 users and their score must appear within 3 seconds.

8.1.7. Searching for games

- 8.1.7.1. When the user searches the game title, the relevant search results need to be displayed within 3 seconds on the game page.
- 8.1.7.2. When the user searches games by using tags, games with the same tag needs to be displayed within 3 seconds on the game page.

8.1.8. Display Dashboard

- 8.1.8.1. When the user clicks to open the dashboard, the dashboard information needs to be loaded within 3 seconds.

8.2 Security Requirements

- 8.2.1. No personal and sensitive information such as the students' complete matriculation number, phone number, password etc. is shared within the website.
- 8.2.2. Minimum password strength of 8 characters is needed during account creation so that the account is more secure.

8.3 Usability Requirements

Usability shows how users can navigate through the game application without receiving any guidance. As this game application allows users to create games, play games and monitor people who plays your quiz's progress, it is important for the game application to be simple and convenient to navigate around. This helps students to learn and teachers to monitor the student's learning progress.

- 8.3.1. A user without a registered account must be able to register an account and play quizzes available.
 - 8.3.1.1. A user must be able to create an account under 2 minutes.
 - 8.3.1.2. A logged-in user must be able to navigate to a game of their choice within 1 minute.
 - 8.3.1.3. User must be able to navigate to the quiz available to them in the game within a minute.
 - 8.3.1.4. User must be able to learn to play the quiz without any guidance within 5 minutes.
- 8.3.2. A minimum font size of 12pt and a readable font style such as 'Calibri' is used for the website.
- 8.3.3. The game creator who created the game can delete the game as well. This allows the creator to revert their action when a wrong game is created.
- 8.3.4. When deleting or creating a game or a quiz, a confirmation message will pop out to ensure that it is the creator's intention to carry out these actions. This prevent games being deleted accidentally.
- 8.3.5. The system allows the creator to edit the questions asked in a quiz, in case if any questions are phrased ambiguously, allowing easy reversal of actions.
- 8.3.6. The structure and colour scheme of the UI are to be consistent so that user can navigate through the website more easily.

8.4 Reliability Requirements

Reliability specifies how the game application can handle increasing workload without failure while maintaining its performance throughput and the accuracy of calculations. It is also determined by following the security protocol used to prevent security breaches by hackers and being compatibility friendly at the same time.

- 8.4.1. The game application must not have a downtime of less than 5 minutes at any point of time.
- 8.4.2. The game application must not allow non authorised users to make SQL injections into the database.
- 8.4.3. The database used by the game application must be able to support any commercial product using MySQL workbench.
- 8.4.4. Creators cannot alter the scores in the game application at any point of time.

- 8.4.5. The calculation logic, such as adding up the users' score must be 100% correct.
- 8.4.6. The players' score must be preserved after the user logs out of the system and logs in again.
- 8.4.7. The questions that are uploaded by the game creator must be preserved after the game creator logs out of the system.

8.5 Scalability Requirements

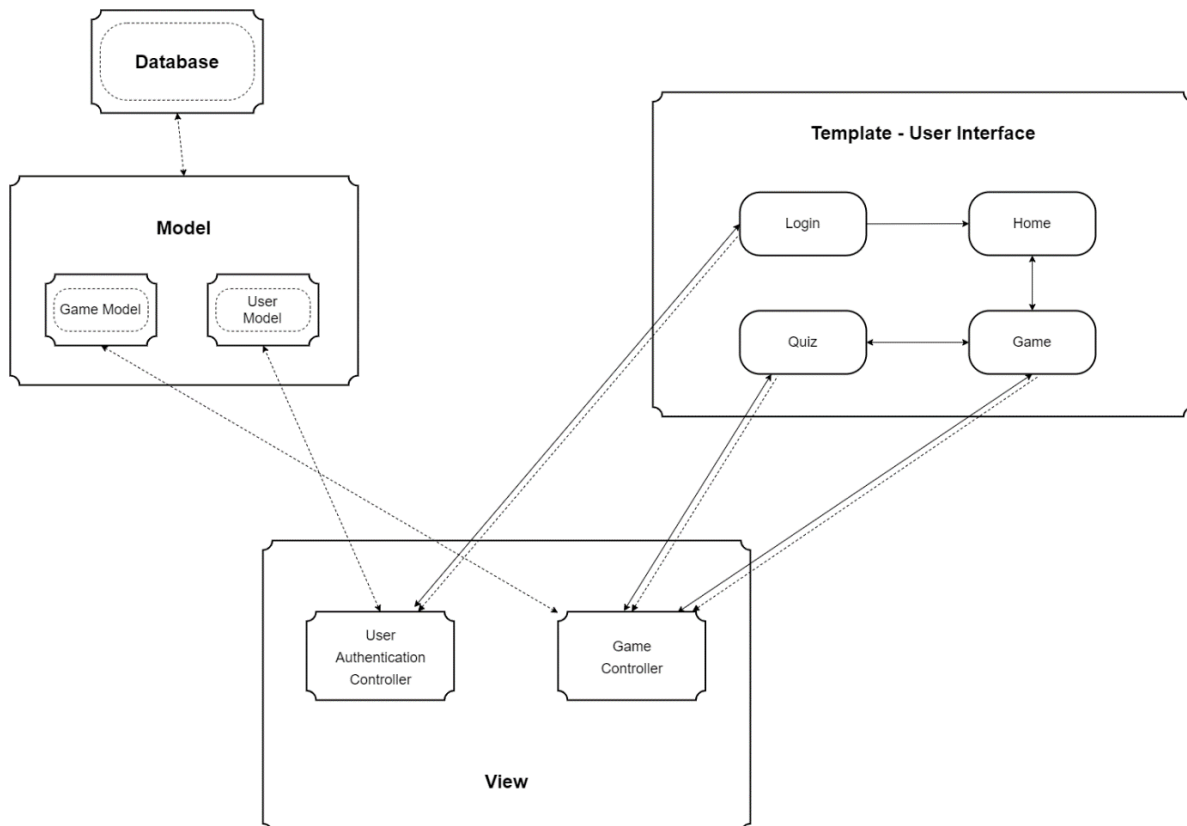
- 8.5.1. The game can fulfil its performance requirements even if the number of accounts registered on the website increases.
- 8.5.2. The website allows adding of new features and functionalities without breaking the current existing features and functions.

8.6 Compatibility

- 8.6.1. The website must be accessible using the latest versions of commonly used web browsers such as Google Chrome and Mozilla Firefox.

9. Analysis Models

9.1. System Architecture



Reason for Architecture:

For our project "HUF", we have chosen to adopt the Model-View-Template architectural style, as it merges best with the Django framework.

The MVT architecture separates the program into the Model, View and Template layers. Since each layer controls its own set of classes and functions, this allows for easy maintenance of the program. Communications between the layers (Model, View, Template) are in the forms of HTTP requests and responses. The Template layer is the user interface of our program, containing all the classes which the user is allowed to interact with and view, such as the leader board page. It also receives input from the user.

When the user logs into the application, control is passed from the login page to the home page in the app. Control can then be passed between the Home page, Game page and Quiz page to render each page as the user navigates between them.

Control and user data are passed from Login to User Authentication Controller. User Authentication Controller will then send user data to the UserModel, which is an Object Relational Model that maps to an SQL table in the database. UserModel will then permanently write user data to the database.

For user data to be rendered in Login Page, the UserModel first retrieves user data from the database. User Authentication Controller then collects data from the user model and renders the user information in Login.

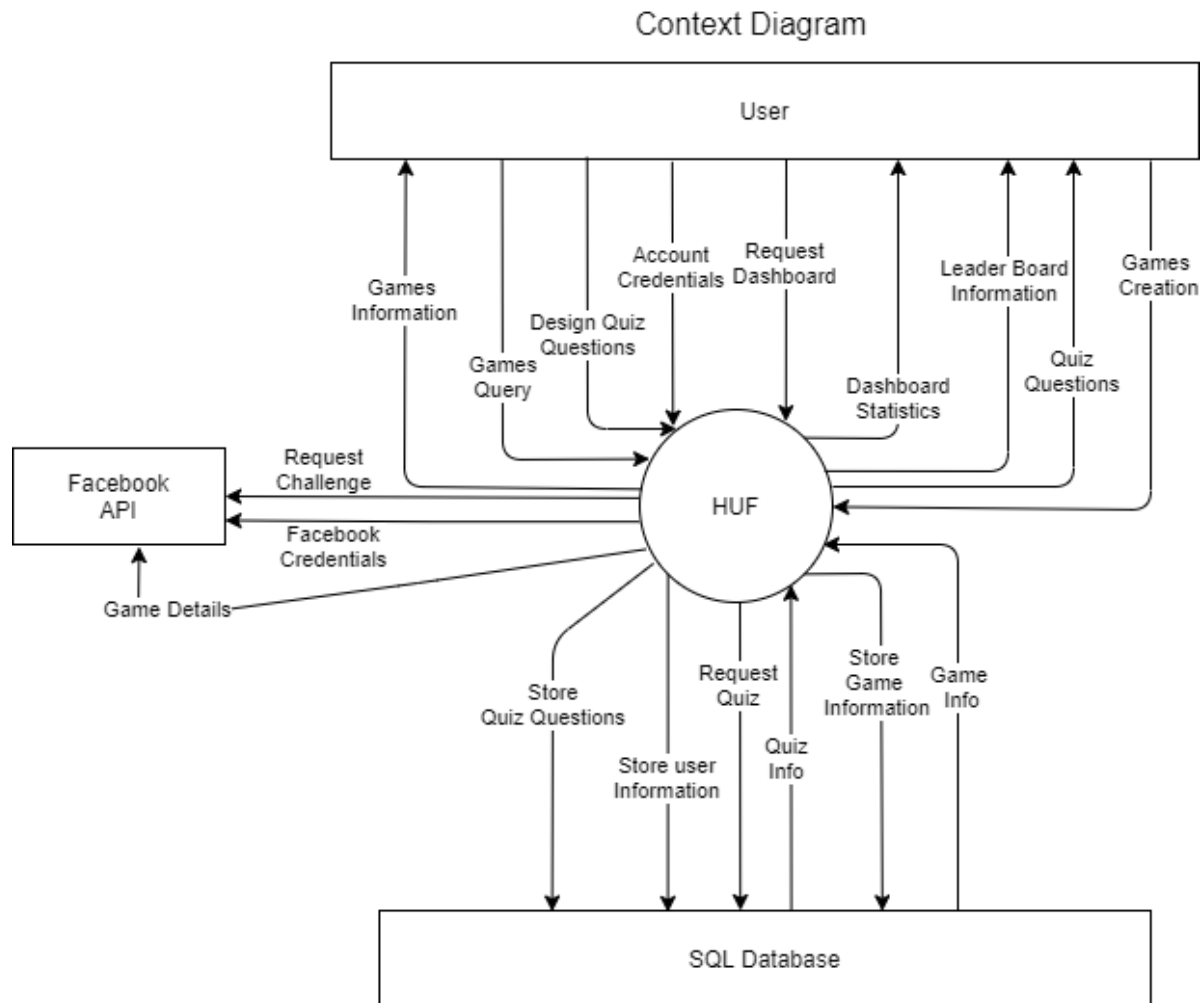
Control and user data are passed from Game and Quiz to Game Controller. Game Controller will then send user data to the GameModel, which will update game data to the database.

For game data to be rendered in Game and Quiz Pages, first GameModel retrieves game data from the database. Game Controller then collects data from the Game model and renders the user information in Game and Quiz pages.

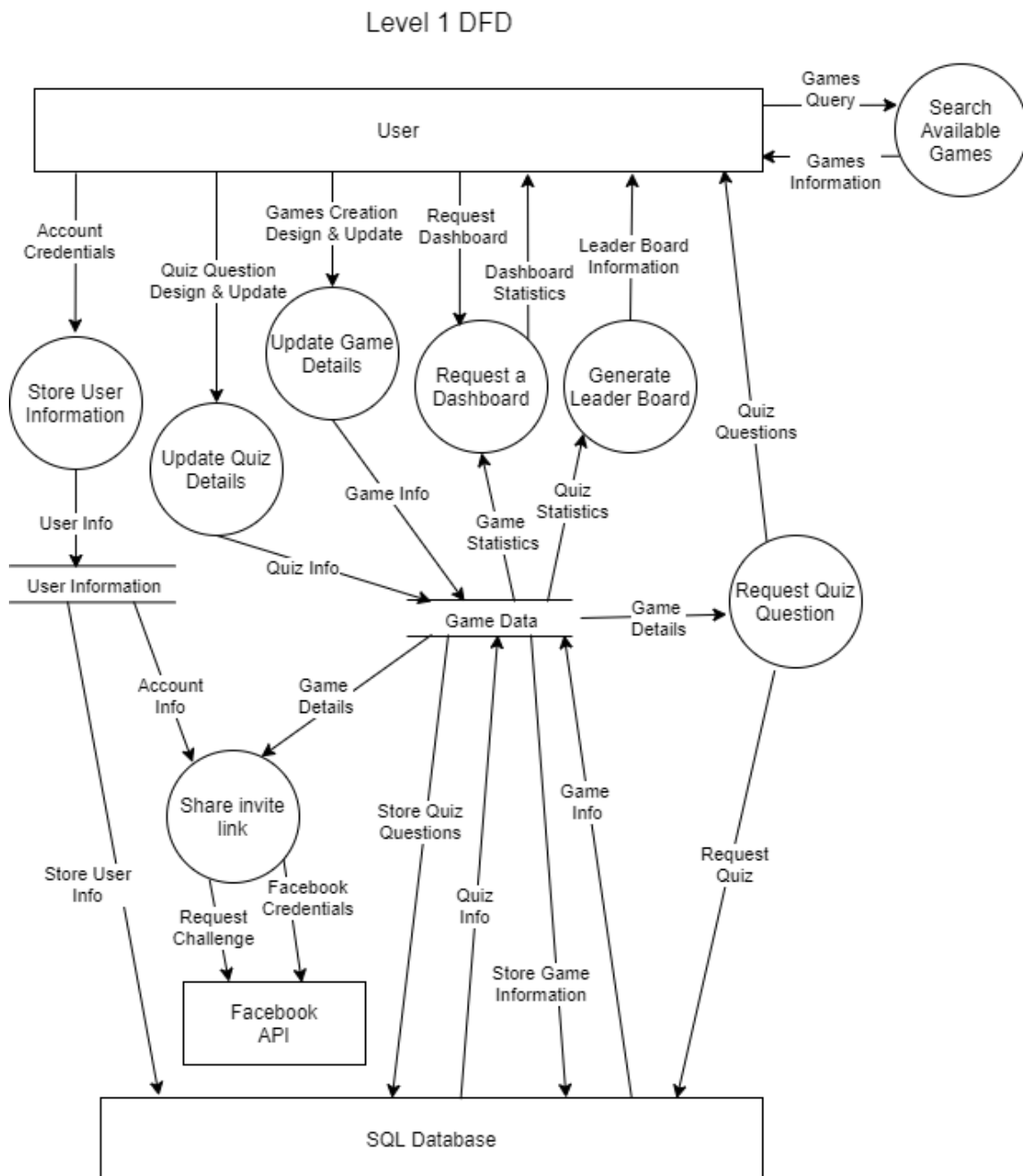
Another architecture style we could have used is the layered architecture. However, we rejected the layered architecture as basic modifications would lead to a complete redeployment of the application.

We have chosen the Model View Template Style over the Model View Controller Style and over other architectural styles because modifications are easier in MVT (tables can be created in Models in Django without needing to write complex SQL code) and communication between the View layer, Model and Template layers is handled by the Django framework, allowing for rapid development. As a result, modifications to the platform will come at low cost in the future.

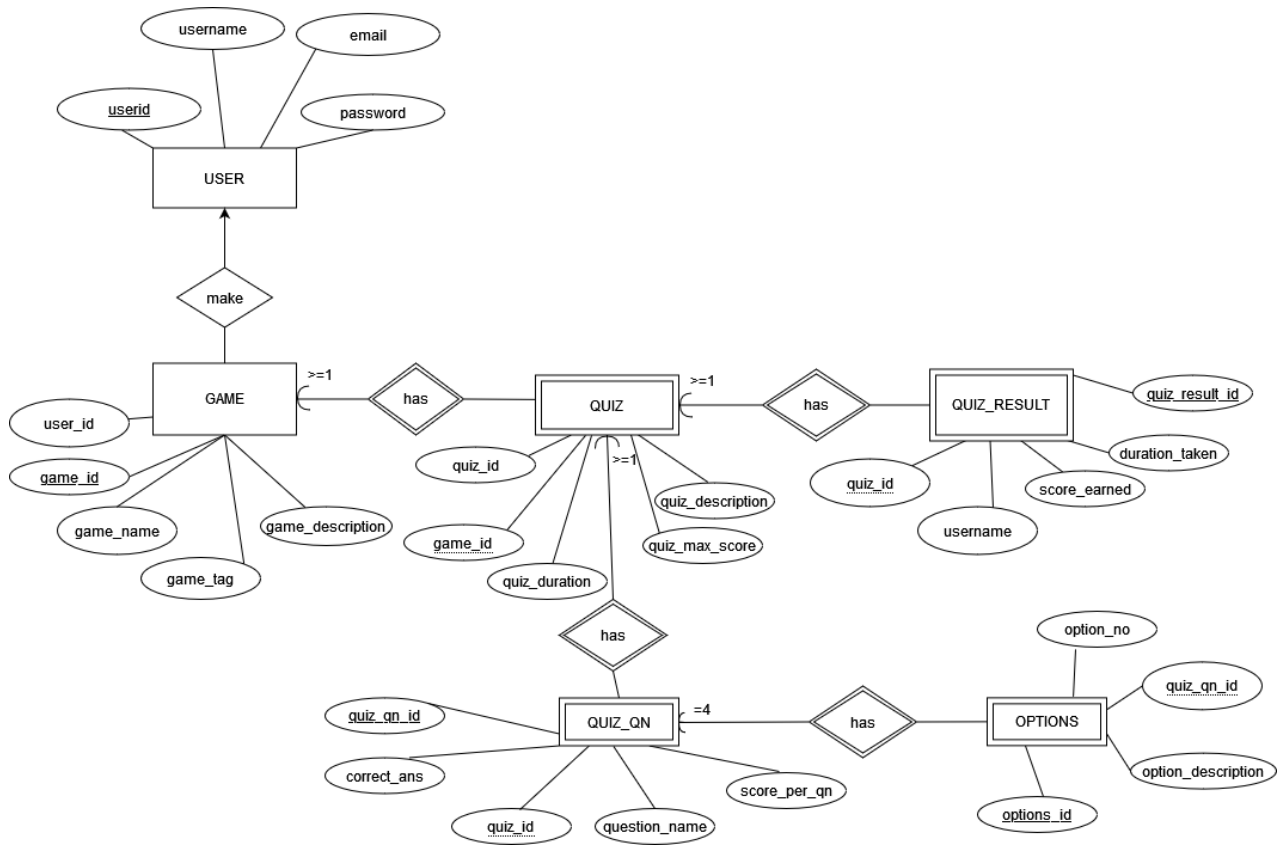
9.2.Context Diagram



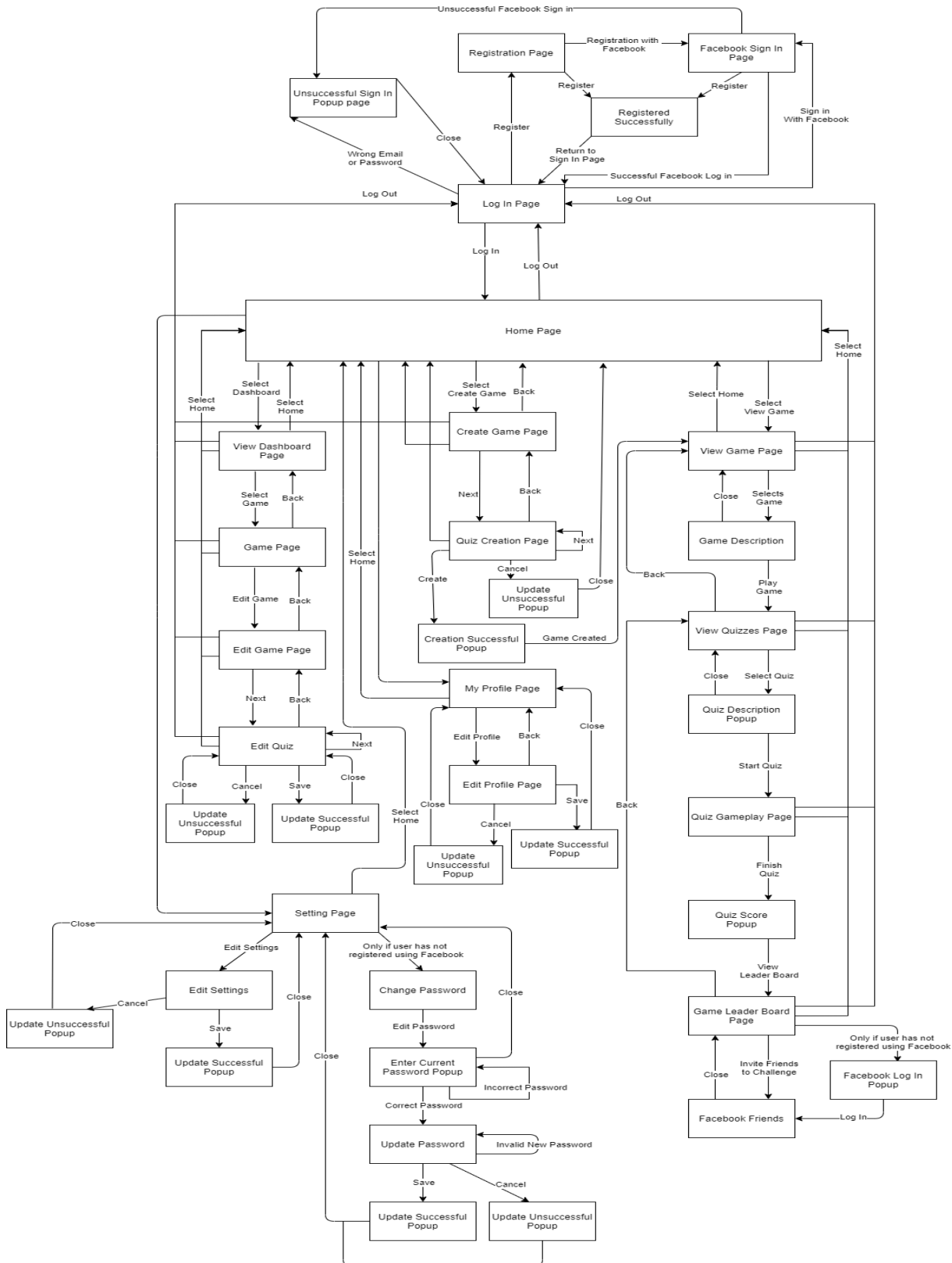
9.3.Data Flow Diagram



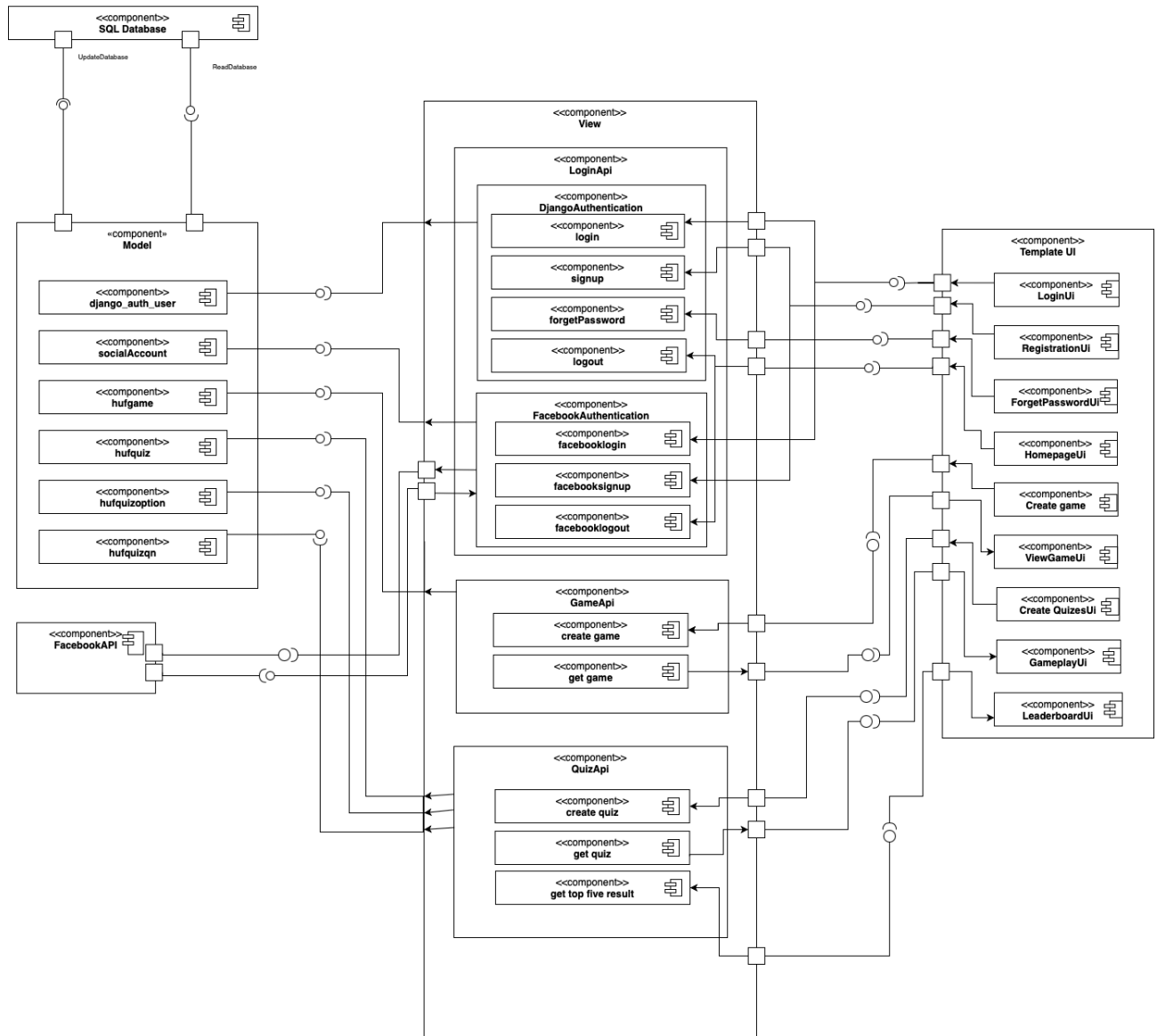
9.4.Entity Relationship Diagram



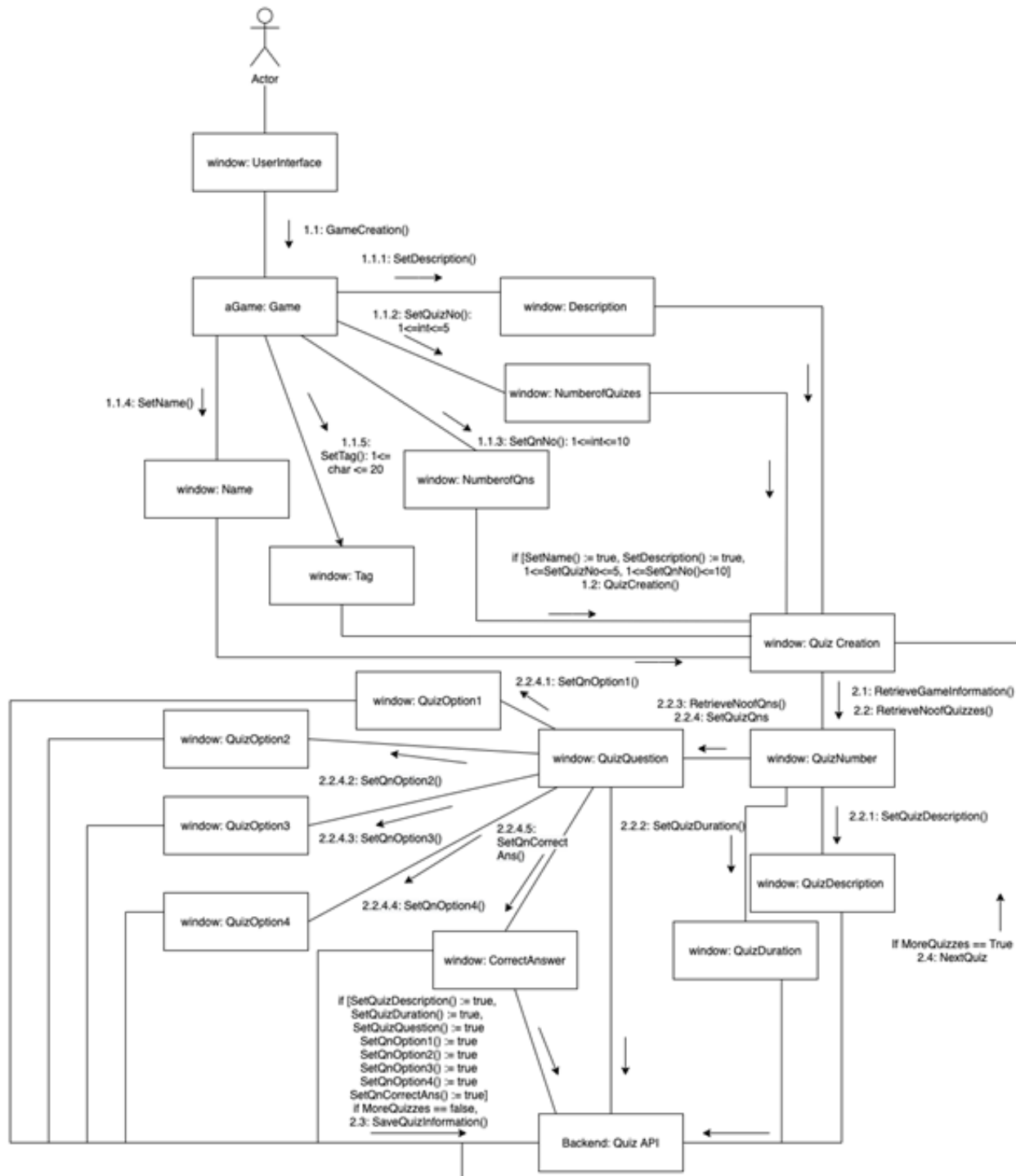
9.5.Dialog Map



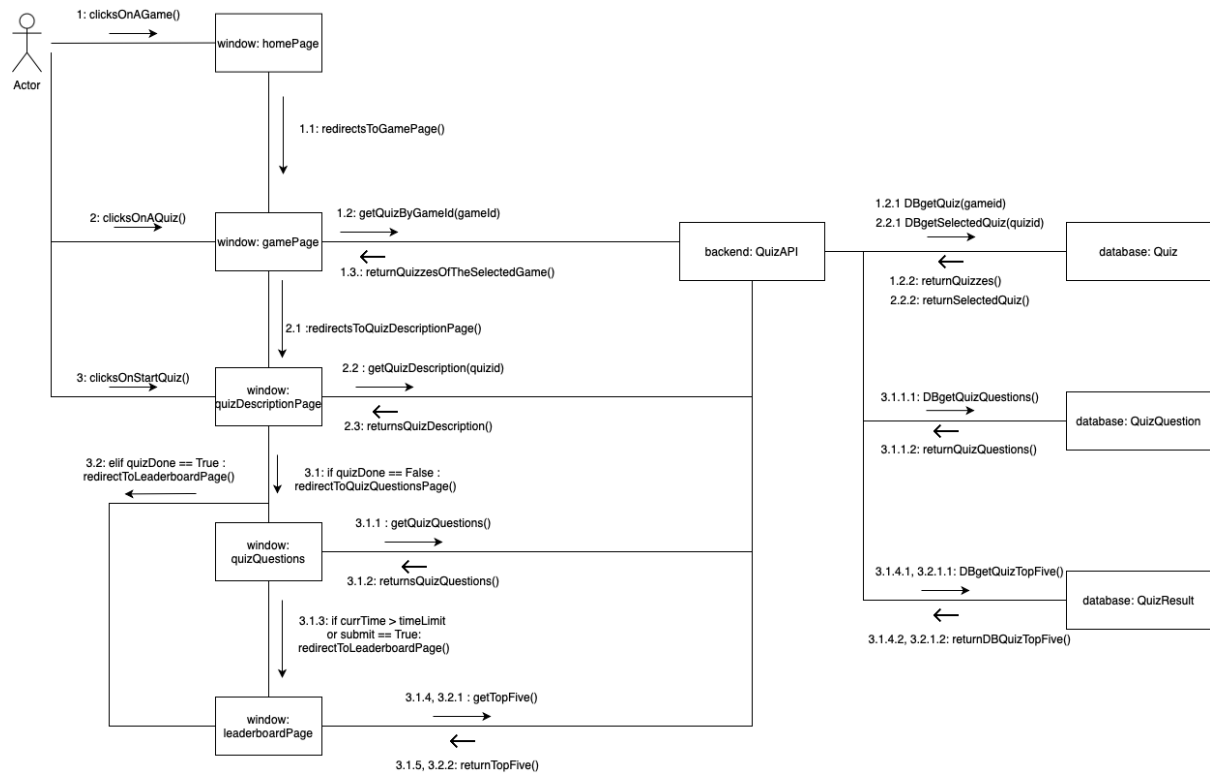
9.6.Component Diagram



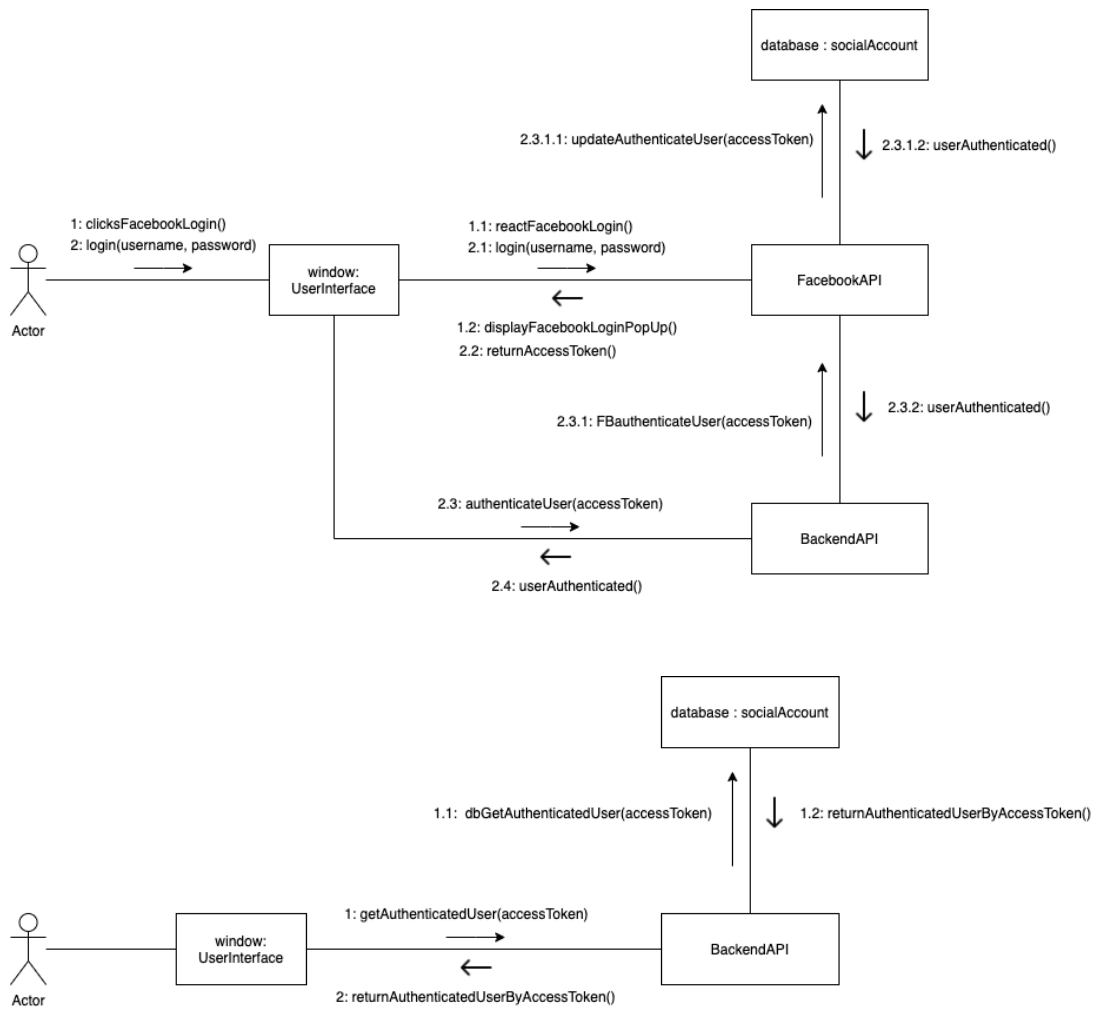
9.7.Communication Diagram (Game and Quiz Creation)



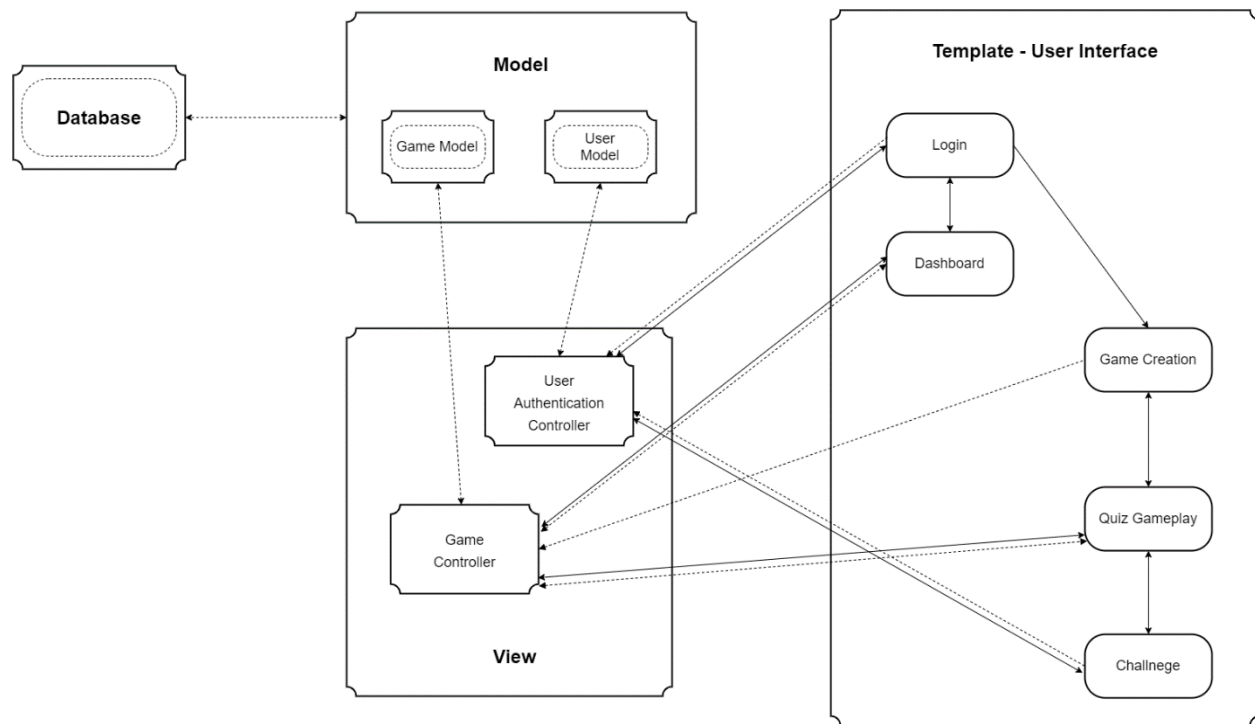
9.8.Communication Diagram (Gameplay)



9.9.Communication Diagram (Facebook Login)



10. Sub System Interface



Considering the messages that we require as well as how these messages are being passed around, we have divided up the whole system based on the functional requirements and designed 5 subsystems that is able to enact the key system usage scenarios. Some functions and parameters that are required for the subsystem to function are also listed down along with some descriptions.

1. **Name:** Log In

Functionality: Gets the user data from database through an API with the use of useEffect Hook in React. Validates the user and log them into the system.

1.1. Public void logIn(String username, String password)

Input Parameters:

- username: username of registered user.
- password: password of registered user.

Return: Returns account data respective to the user.

Pre-condition: User must already have a registered account.

Usage Scenario: User would like to access the system.

1.2. Public void forgetPassword(String email)

Input Parameters:

- email: email address of registered user.

Return: Returns a http response “your new password has been sent to your email”

Pre-condition: User must already have a registered account

Usage Scenario: User has forgotten about the password and would like to get access to the system using the registered email address.

1.3. Public void registerAccount(String username, String email, String password)

Input Parameters:

- username: username of registered user.
- email: email address to register user.
- Password: password of user

Return: Returns a new account data created

Pre-condition: Email address must not be tied to an existing registered account.

Usage Scenario: User would like to create an account in the system.

2. Name: Game creation

Functionality: Takes in the user input as the information needed to create a game which will host all the quiz and its questions for other users to solve. Information will be stored in the back-end database.

2.1. Public void gameCreation(String gameName, String gameDescription, int noOfQuiz, int noOfQuestionsPerQuiz, String gameTag, String userName)

Input Parameters:

- gameName: The name of the game.
- gameDescription: The description of the game.
- noQuiz: User input number of quizzes within the game.
- noOfQuestionsPerQuiz: the number of questions under each quiz (max of 5).
- gameTag: Tag relating to game. Eg. Geography, Mathematics, etc.
- userName: The name of the game creator.

Return: none

Pre-condition: User must be logged into the system.

Usage Scenario: User would like to create his/her own game.

2.2. Public void quizCreation(String userName, int gameId, String quizName, int quizMaxScore, String quizDescription, int noOfQuestionsPerQuiz, int duration)

Input Parameters:

- userName: The name of the game creator.
- quizName: name of Quiz.

- quizMaxScore: the maximum score for the quiz
- quizDescription: the description of the quiz
- noOfQuestionsPerQuiz: number of questions within quiz.
- duration: time limit set for quiz completion.

Return: none

Pre-condition: A game must be created by the same user.

Usage Scenario: User would like to create quizzes for other users to solve.

3. **Name:** QuizGamePlay

Functionality: When user starts a quiz, the system will use an API to retrieve the quizzes that belong to the game. After which, the system will award and record the scores for the players as they progress in the game.

3.1. Public ArrayList<int> fetchQuiz(int gameId, int quizId)

Input Parameter:

- gameId: Id of the game. QuizId : Id of the quid.

Return: Retrieves the quiz questions, options and answer of the selected quiz of the selected game.

Pre-condition: The quiz must be created. The quiz must either be a quiz of the first level or the user must have completed quizzes of all previous levels.

Usage Scenario: An existing player that is not the creator is interested in playing the game.

3.2. Public int SetUserAnswers (Int ans)

Input Parameter:

- ans: The ans that the user inputs

Return: Inputs the user's answer for the selected question of the selected quiz of the selected game.

Pre-condition: The quiz must be created. The quiz must either be a quiz of the first level or the user must have completed quizzes of all previous levels.

Usage Scenario: An existing player that is not the creator is interested in playing the game.

3.3. Public void submitQuiz(ArrayList<int> answers)

Input Parameter:

- answers: a list of the user's corresponding answers for the quiz.

Return: none

Pre-condition: The game must be created, and the player must answer all the questions in the quiz.

Usage Scenario: When the player has completed answering the quiz.

3.4. `Public int checkAllAnswers(int gameId, int quizId, int QuizQuestionId, ArrayList<int> answers))`

Input Parameter:

- gameId: ID of game
- quizId: ID of the quiz.
- quizQuestionID: ID of each question within the quiz.
- answers: a list of the user's corresponding answers for the quiz.

Return: Checks if answers are correct and returns total score for the quiz.

Pre-condition: Targeted quiz must already be completed/answered by the player and answers retrieved.

Usage Scenario: When player completes the quiz, and the system would return the score of the player.

4. **Name:** Dashboard

Functionality: To allow the creator of the game to view the statistics relating to his game. It will also allow the creator to modify the selected game and quizzes.

4.1. `Public void showStats (String userName, int gameId, int Score)`

Input Parameters:

- userName: The name of the game creator.
- gameId: ID of the game.

Return: Retrieves and Displays all the games the user has created, all the quizzes the user has created, total number of players, average time taken by the user to complete the test, the average game score, challenges sent, challenges accepted

Pre-condition: The game and the quizzes under the game must be created. At least one user must have played the game.

Usage Scenario: When the creator would like to gain access to statistical insights on the game that was created or would like to modify the game or quizzes.

4.2. `Public void editGame(String userName, int gameId, String gameDescription)`

Input Parameters:

- userName: The name of the game creator
- gameId: ID of the game.

Return: The edited game is stored back into the data store. If a game is deleted, the data store discards the game_id and all related information.

Pre-condition: The game must be created.

Usage Scenario: When the creator would like to modify the game.

4.3. Public void editQuiz(String userName, int gameId, int quizId, String quizDescription, String quizQuestion, String options, int answers)

Input Parameters:

- userName: The name of the game creator
- gameId: ID of the game.
- quizID: ID of the quiz.
- quizDescription: Description of the quiz
- quizQuestion: Quiz Question
- Options: Choices for user selection
- Answers: Correct answer for then quiz

Return: The edited quiz is stored back into the database. If a quiz is deleted, the data base discards of the quizId and all related information.

Pre-condition: At least one quiz must be created.

Usage Scenario: When the creator would like to modify the quiz/quizzes.

5. **Name:** Challenge

Functionality: Using Facebook API the player is able to send an invite to his/her Facebook friends that also has a registered account with the system.

5.1. Public void sendInvite(String userName, int gameId)

Input Parameters:

- userName: The name of the game creator
- gameId: ID of the game to be shared

Return: Challenge send confirmation pop-up.

Pre-condition: The game must be created and the player sending the invite must already complete the quiz.

Usage Scenario: When the player completes the game and would like to send an invite to his/her friends to compete with them.

11. Test Plan

11.1. Testing Strategies and Techniques

Our testing strategy uses Black Box Testing, System Testing and Integration Testing.

Manual testing is chosen for its ease-of-use and convenience.

Black Box testing was used to test functions like Create Game where different inputs can be tested easily for correctness. We entered a series of different inputs into the system and checked that the expected outcome was achieved. We also utilised unit testing to test individual modules in our software to check if their outputs were correct, achieved through the inbuilt python and Django testing modules.

For integration testing, we automated tests to investigate integration of Models in the Django Framework.

Systems testing was used to test systems such as Game Creation and Registration. We decided to carry out manual systems testing as our app is small sized but complex, therefore manual testing would allow us to test more scenarios under a shorter period of time to verify correctness.

11.2. Blackbox Testing

For this part of the testing, we will be doing testing on the major functions of the application based on as much scenarios as possible during a normal session of use.

11.2.1. Log In

S/N	Test Scenario	Expected Outcome	Actual Results
1	Login with valid Username and password.	The system redirects the user to the home page.	The system redirects the user to the home page.
2	Wrong Username, Valid Input for Password.	The system displays an Error message: "Login failed. Please check your credentials and try again.".	The system displays an Error message: "Login failed. Please check your credentials and try again.".
3	Valid Username, Wrong Password.	The system displays an error message: "Login failed. Please check your credentials and try again.".	The system displays an Error message: "Login failed. Please check your credentials and try again.".
4	Empty Username field, Valid Input for Password.	The system displays a warning text: "Please input your username!".	The system displays a warning text: "Please input your username!".
5	Valid Input Username field, Empty Password field.	The system displays a warning text: "Please input your Password!".	The system displays a warning text: "Please input your Password!".
6	Empty Username and Password field.	The system displays both warning text: "Please input your username!" & "Please input your password!".	The system displays both warning text: "Please input your username!" & "Please input your password!".

11.2.2. Register

S/N	Test Scenario	Expected Outcome	Actual Results
1	Register with all fields empty.	The system displays warning texts for each individual field and prompts the user to fill in the empty fields.	The system displays warning texts for each individual field and prompts the user to fill in the empty fields.
2	Invalid email address format.	Error message display: "This is not a valid email address."	Error message display: "This is not a valid email address."
3	Mismatching Passwords.	The system displays a warning text: "The two passwords entered do not match!"	The system displays a warning text: "The two passwords entered do not match!"
4	Password has insufficient characters.	The system displays a warning text: "Password must have at least 8 characters."	The system displays a warning text: "Password must have at least 8 characters."
5	Username already taken by an existing user.	The system displays a warning text: "A user with that username already exists".	The system displays a warning text: "A user with that username already exists".
6	Register with empty Username field	The system displays a warning text: "Please enter your name!".	The system displays a warning text: "Please enter your name!".
7	Register with empty E-mail field	The system displays a warning text: "Please enter your email address!".	The system displays a warning text: "Please enter your email address!".
8	Register with empty Password field	The system displays a warning text: "Please enter your Password!".	The system displays a warning text: "Please enter your Password!".
9	Register without confirming Password	The system displays a warning text: "Reconfirm your Password!".	The system displays a warning text: "Reconfirm your Password!".

11.2.3. Forget Password

S/N	Test Scenario	Expected Outcome	Actual Results
1	User enters a valid email address that has already been registered with the system	<p>The system displays a message: “Success! Your new password has been sent to your email.”</p> <p>The system will send a new password to the email of the user that the user will be able to use to log in.</p>	<p>The system displays a message: “Success! Your new password has been sent to your email.”</p> <p>The system will send a new password to the email of the user that the user will be able to use to log in.</p>
2	User enters an invalid email address that has not been registered with the system.	The system displays a warning text: “Please ensure that you have entered a valid email address registered with HUF.”	The system displays a warning text: “Please ensure that you have entered a valid email address registered with HUF.”
3	User presses the ‘Back’ button at the Forget Password page without filling anything in the textbox.	User is directed to the login page.	User is returned to the login page.
4	User presses the ‘Back’ button at the Forget Password page with textbox filled.	User will receive a prompt that the page has details that is filled and confirms if user wants to go back. Upon clicking ‘OK’, the user is directed to the login page.	User will receive a prompt that the page has details that is filled and confirms if user wants to go back. Upon clicking ‘OK’, the user is directed to the login page.

11.2.4. Create Game

S/N	Test Scenario	Expected Outcome	Actual Results
1	Game Name field is left empty.	The system displays a warning text: "Please input the Game Name".	The system displays a warning text: "Please input the Game Name".
	Game Description field is left empty.	The system displays a warning text: "Please input the Game Description".	The system displays a warning text: "Please input the Game Description".
	Game Tag field is left empty.	The system displays a warning text: "Please input the Game Tag".	The system displays a warning text: "Please input the Game Tag".
3	Creator clicks on 'Back' button.	Creator is directed to the 'Home page'.	Creator is directed to the 'Home page'.
5	Creator has filled in "Game name", "Game Description", "Number of quizzes", "Game tag" with correct inputs and clicks "Next" button.	The system displays a message: "You have successfully created a new game."	The system displays a message: "You have successfully created a new game."

11.2.5. Create Quiz

S/N	Test Scenario	Expected Outcome	Actual Results
1	Creator leaves input empty for quiz description	Error message display at corresponding input field: "Please input quiz description!"	Error message display at corresponding input field: "Please input quiz description!"
2	Creator tries to enter a duration less than 0 sec.	Input field will automatically select minimum value (1 sec) as duration.	Input field will automatically select minimum value (1 sec) as duration.
	Creator tries to enter a duration more than 3600 sec.	Input field will automatically select maximum value (3600 sec) as duration.	Input field will automatically select maximum value (3600 sec) as duration.
3	Creator leaves one or more "Question" input empty.	Error message display at corresponding input field(s): "Please input question!"	Error message display at corresponding input field: "Please input Question!"
4	Creator leaves one or more "Options" input empty.	Error message display at corresponding input field(s): "Please input <i>"first"</i> * option!"	Error message display at corresponding input field(s): "Please input <i>"first"</i> * option!"
5	Creator tries to set the numbering option for the correct answer to be out of option boundary (1-4).	Input field will automatically select the minimum option of 1 or maximum option of 4 depending on the input entered by the user.	Input field will automatically select the minimum option of 1 or maximum option of 4 depending on the input entered by the user.
6	Creator tries to set the score obtained for any specific question to be out of score boundary (1-5).	Input field will automatically select the minimum option of 1 or maximum option of 5 depending on the input entered by the user.	Input field will automatically select the minimum option of 1 or maximum option of 5 depending on the input entered by the user.
7	Creator has filled in all necessary fields with the correct inputs.	The system displays a message: "You have successfully created quizzes for game <i>Game Name</i> ".	The system displays a message: "You have successfully created quizzes for game <i>Game Name</i> ".

**It can be any option out of first, second, third and fourth*

11.2.6. Quiz Gameplay

S/N	Test Scenario	Expected Outcome	Actual Results
1	Player runs out of time where Timer goes to 0:00.	The system displays an error message display: "Times Up!" player's score will be calculated and redirect player to the leader board.	The system displays an error message display: "Times Up!" player's score will be calculated and redirect player to the leader board.
2	Options input not all are filled and player clicks on 'Finish Quiz'.	The system displays an error message display: "Please select an answer!" at the option with an empty field.	The system displays an error message display: "Please select an answer!" at the option with an empty field.
3	When player answers all questions and clicks on 'Finish Quiz'.	The system displays a message: "Completed". Players' answers will be submitted, and player will be redirected to the leader board page.	The system displays a message: "Completed". Players' answers will be submitted, and player will be redirected to the leader board page.
4	player clicks 'Back' whilst playing quiz.	Player will be redirected back to the quiz page.	Player will be redirected back to the quiz page.
5	Player tries to submit the quiz when they have done it before.	System will show a message 'You have already completed the quiz.' after the player completes the quiz and will be redirected back to games page.	System will show a message 'You have already completed the quiz.' after the player completes the quiz and will be redirected back to games page.

11.3. System Testing

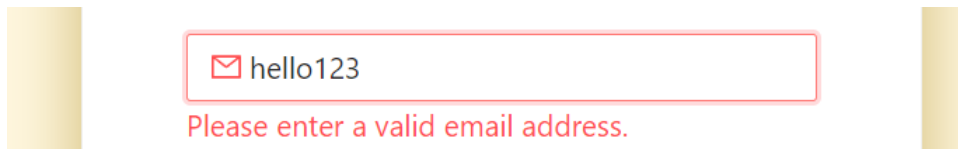
This is a validation test that is performed to ensure that the requirements of the system matches the functional and non-functional requirements. This testing will help to surface limitations that the software has.

11.3.1. Functional Testing

Manual testing of the following common usage scenarios will be performed:

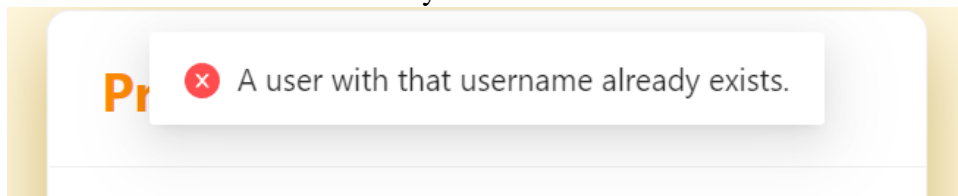
11.3.1.1. Registration

- Invalid Email
 - If user tries to enter an invalid email



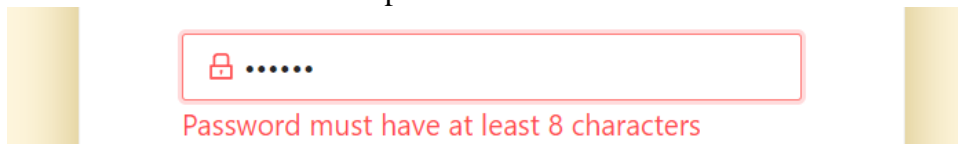
A screenshot of a registration form. It shows a text input field with a red border and a red envelope icon on the left. The text inside the field is "hello123". Below the field, there is a red error message: "Please enter a valid email address."

- Username already in database



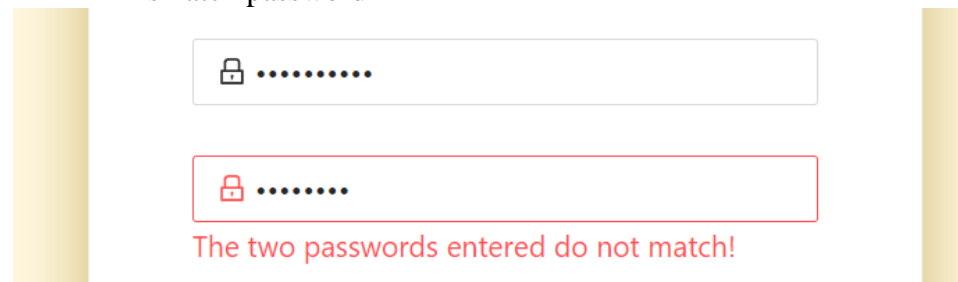
A screenshot of a registration form showing a success message. The message is displayed in a white box with a red border and a red 'x' icon. The text reads: "A user with that username already exists."

- Invalid Password
 - Weak password - Password has less than 8 characters



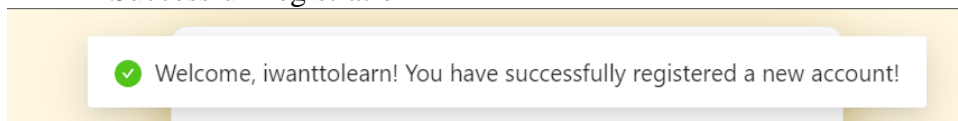
A screenshot of a registration form. It shows a text input field with a red border and a red padlock icon on the left. The text inside the field is ".....". Below the field, there is a red error message: "Password must have at least 8 characters"

- Mismatch password



A screenshot of a registration form showing two password input fields. The top field has a red border and a red padlock icon, with the text ".....". The bottom field has a red border and a red padlock icon, with the text ".....". Below the fields, there is a red error message: "The two passwords entered do not match!"

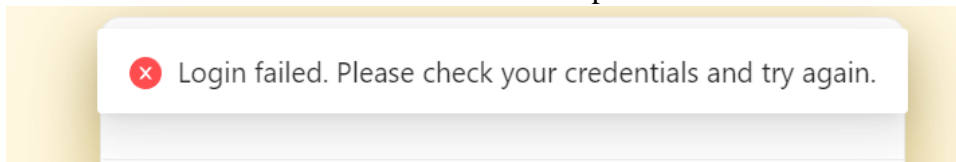
- Successful Registration



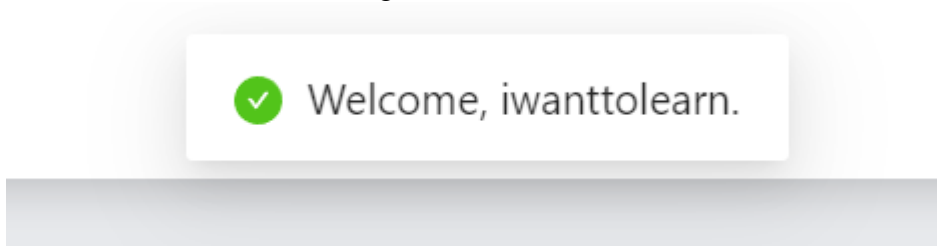
A screenshot of a registration form showing a success message. The message is displayed in a white box with a red border and a green checkmark icon. The text reads: "Welcome, iwanttolearn! You have successfully registered a new account!"

11.3.1.2. Login

- Invalid Username or incorrect password

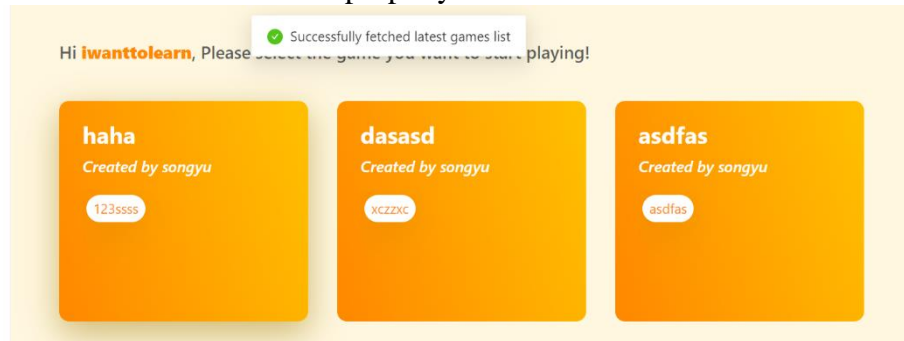


- Successful Login

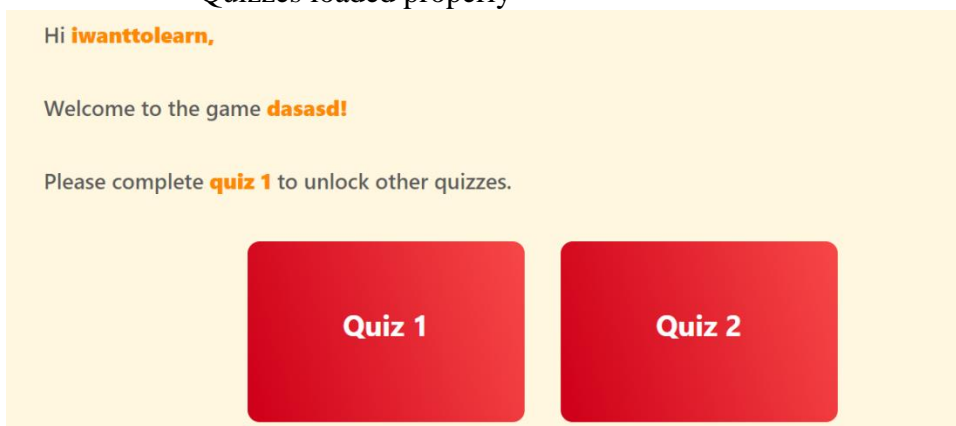


11.3.1.3. Gameplay

- Games loaded properly



- Quizzes loaded properly



- All questions and options successfully loaded

Back

dasasd | Quiz 1

Time Left: 00:05:52

Question 1) hello

☐ saf
☐ fasgsa
☐ asfasf
☐ asgsdh

Question 2) my name

☐ tkyu
☐ lyuf
☐ wet
☐ 541g56

Finish Quiz

- User successfully completed quiz, and score posted.



DASASD | Leaderboard

Your Score **8**

Rank	Name	Score
1	<div>S</div> ssss	65
2	<div>A</div> aileen	53
3	<div>A</div> aileenlaksmono1	53
4	<div>H</div> hufadmin	15

11.3.1.4. Game and Quiz Creation

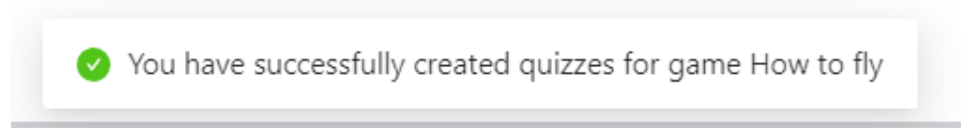
- Game Creation – 1 field empty
 - This is the example for empty ‘Game Description’ field, similar error message will appear for other inputs that are blank.

* Game Description:

Enter Game Description

Please input the Game Description

- Quiz Creation – All fields filled



- Quiz Creation – 1 field empty
 - This is the example for empty ‘Option 3’ field, similar error message will appear for other inputs that are blank.

* Option 3:

Please input option 3!

- Game Creation – All fields filled

✓ You have successfully created a new game How to fly

11.3.1.5. Gameplay

- Leaving a question blank and user attempts to submit quiz

Question 1) hello

☐ saf

☐ fasgsa

☐ asfasf

☐ asgsdh

Please select an answer!

- Timer runs out of time.
 - User’s score will be calculated, posted into the database and redirected to leaderboard page.

Times Up!

Score is being calculated!

You will be directed to the leaderboard page.

[Close](#)

- User successfully completes quiz in time.
 - User’s score will be calculated, posted into the database and redirected to leaderboard page.

Completed.

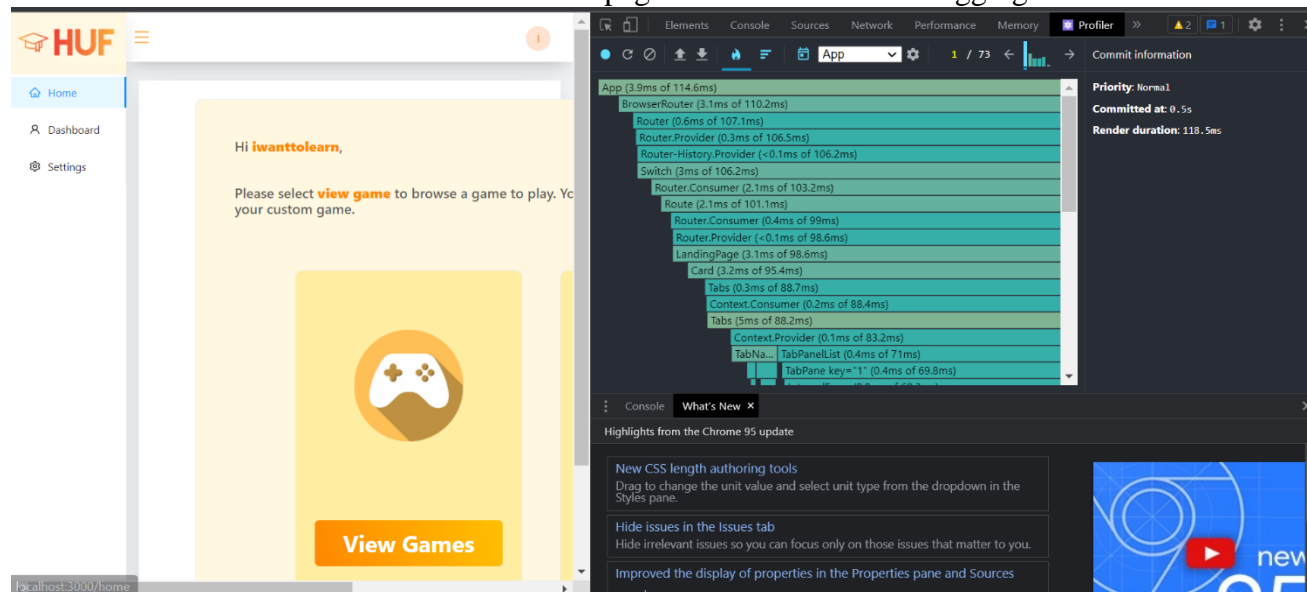
DASASD | Leaderboard Your Score **8**

Rank	Name	Score
1	T testing	16
2	H hufadmin	2

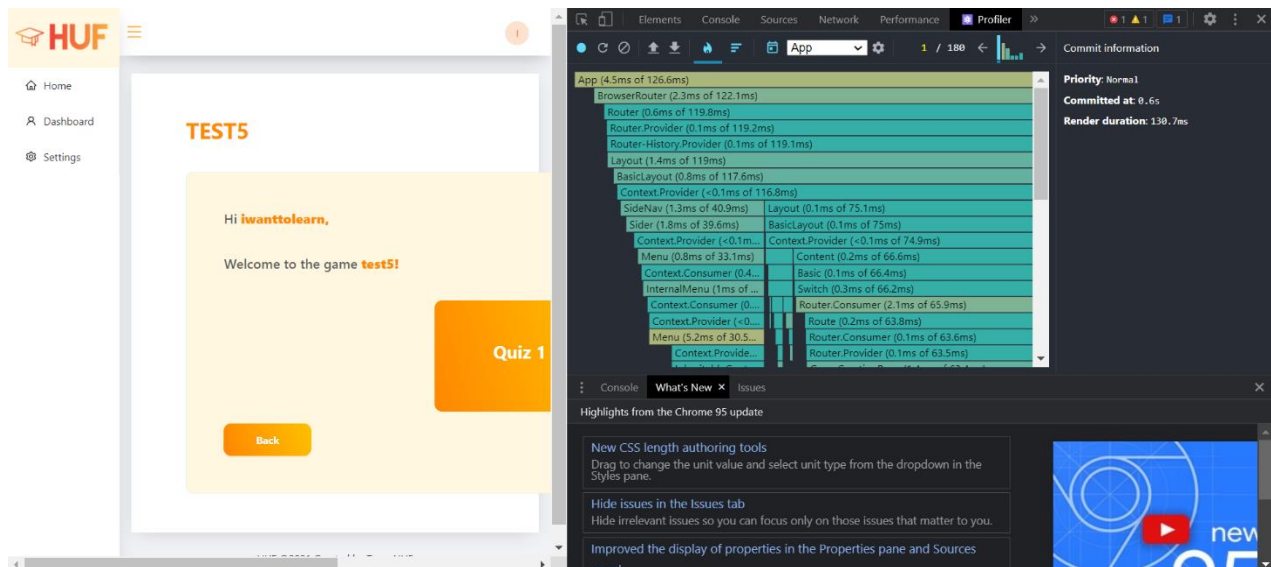
11.3.2. Performance Testing

To test the efficiency of the system, we have performed load testing. Here we refer to some of the non-functional performance requirements that were established in the SRS. We used React developer tools to get the render time for the pages. Some of the non-functional requirements that we have tested are:

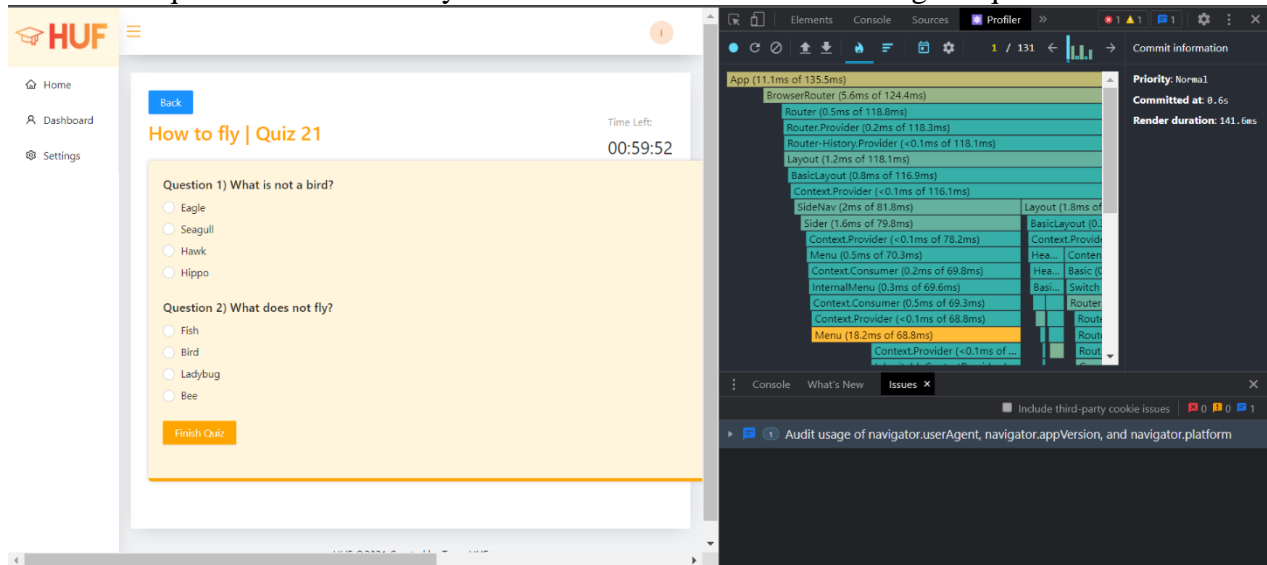
- The user must be able to see the home page within 3 seconds of logging in.



- The game that the user creates is successfully created and updated onto the games page within 10s of the users submitting all information regarding the games and the quizzes within the game.



- The quiz loads successfully within 5 seconds of the user starting the quiz



We can see for all 3 tests that it renders the page in less than the required time.

11.3.3. Integration Testing

The purpose of our integration testing was to investigate how well different modules in our system interact together. In our integration tests, we test how the different tables in our database link together. This is done in testing the interactions between

1. Quiz Creation and Game Creation
2. Quiz Question Creation and Quiz Creation
3. Pairwise Integration between Question Option and Quiz Question

11.3.3.1. Pairwise Integration between Quiz Creation and Game Creation

Usage Scenario: Creating a quiz under a specific game

Code Snippet:

```
User = get_user_model()
class GameTestCast(TestCase):

    def setUp(self):_# Python's builtin unittest
        user_a = User(username='cfe', email='cfe@invalid.com')
        user_a.is_staff = True
        user_a.is_superuser = True
        user_a.save()
        user_a_pw = '12345678'
        self.user_a_pw = user_a_pw
        self.user_a = user_a
        user_a.set_password(user_a_pw)
        HufGame.objects.create(game_id=__1, username=__user_a, game_name=__"johns game",
                               game_tag=__"tag1", no_of_quiz=__5, game_description=__"test game",
                               total_no_qn=__5)

        HufQuiz.objects.create(quiz_id=__1, game_id=__HufGame.objects.get(game_id=__1),
                               quiz_duration=__10, quiz_max_score=__10, quiz_description=__"test quiz",
                               no_of_qn=__5)

    def test_quiz_game_link(self):
        """Test if Quiz and Game are correctly linked"""
        self.assertEqual(HufGame.objects.get(game_id=__1), HufQuiz.objects.get(quiz_id=__1).game_id)
```

Explanation: For this test, we are trying to check if we can create an instance of a Quiz that would be part of a Game, and whether our database would show that they are correctly linked. A HufGame and a HufQuiz instance are created, where the HufQuiz instance is initiated with a game_id of the HufGame instance. In the test_quiz_game_link function, we check if the HufQuiz instance contains the game_id corresponding to the HufGame game_id. Since the output returns OK, this means that the Quiz instance that we created does indeed contain the game_id as a foreign key of the HufGame instance we created.

Output:

```
Ran 1 test in 3.376s

OK
Destroying test database for alias 'default'...
```

11.3.3.2. Pairwise Integration between Quiz Question Creation and Quiz Creation

Usage Scenario: Creating a quiz question under a specific quiz

Code Snippet:

```
def setUp(self): # Python's builtin unittest
    user_a = User(username='cfe', email='cfe@invalid.com')
    user_a.is_staff = True
    user_a.is_superuser = True
    user_a.save()
    user_a_pw = '12345678'
    self.user_a_pw = user_a_pw
    self.user_a = user_a
    user_a.set_password(user_a_pw)
    HufGame.objects.create(game_id=__1, username=__user_a, game_name=__"johns game",
                           game_tag=__"tag1", no_of_quiz=__5, game_description=__"test game",
                           total_no_qn=__5)

    HufQuiz.objects.create(quiz_id=__1, game_id=__HufGame.objects.get(game_id=__1),
                           quiz_duration=__10, quiz_max_score=__10, quiz_description=__"test quiz",
                           no_of_qn=__5)

    HufQuizQn.objects.create(quiz_qn_id=1, quiz_id=HufQuiz.objects.get(quiz_id=1), correct_ans=3,
                             question_name="test question", score_per_qn=5)

def test_quizqn_quiz_link(self):
    """ tests if quiz and quiz qn are correctly linked """
    self.assertEqual(HufQuiz.objects.get(quiz_id=1), HufQuizQn.objects.get(quiz_qn_id=1).quiz_id)
```

Explanation: For this test, we see if we can create a QuizQn object that would be part of a Quiz. A HufQuizQn instance is created with the same quiz_id as the HufQuiz instance. In the test function test_quizqn_quiz_link, we test that the HufQuizQn can instance we created contains the foreign key of the HufQuiz instance in our database. Since the output returns OK, this means that HufQuizQn and HufQuiz are correctly linked.

Output:

```
Ran 1 test in 3.598s

OK
Destroying test database for alias 'default'...
```

11.3.3.3. Pairwise Integration between Question Option and Quiz Question

```
def setUp(self):_# Python's builtin unittest
    user_a = User(username='cfe', email='cfe@invalid.com')
    user_a.is_staff = True
    user_a.is_superuser = True
    user_a.save()
    user_a_pw = '12345678'
    self.user_a_pw = user_a_pw
    self.user_a = user_a
    user_a.set_password(user_a_pw)
    HufGame.objects.create(game_id=_1, username=_user_a, game_name=_ "johns game",
                           game_tag=_ "tag1", no_of_quiz=_5, game_description=_ "test game",
                           total_no_qn=_5)

    HufQuiz.objects.create(quiz_id=_1, game_id=_HufGame.objects.get(game_id=_1),
                           quiz_duration=_10, quiz_max_score=_10, quiz_description=_ "test quiz",
                           no_of_qn=_5)

    HufQuizQn.objects.create(quiz_qn_id=1, quiz_id=HufQuiz.objects.get(quiz_id=1), correct_ans=3,
                             question_name="test question", score_per_qn=5)

    HufQuizOption(quiz_qn_id=_HufQuizQn.objects.get(quiz_qn_id=1), option_id=_1, option_description=_ "test descrip")

def test_quizresult_quiz_game_link(self):
    """ tests if quiz option and quiz question are correctly linked """
    self.assertEqual(HufQuizQn.objects.get(quiz_qn_id=1), HufQuizOption.objects.get(option_id=1).quiz_qn_id)
```

Explanation: For this test, we check if we can create a Quiz Option that would be properly linked under a Quiz Question in our database. We create separate instances of HufQuizOption and HufQuizQn, where the HufQuizOption model would be instantiated with the quiz_qn_id of the HufQuizOption we created. In the test_quizop_quizqn_link function, we test that the instance of HufQuizOption that we created would correctly be linked to the HufQuizQuestion we created in our database. Since the test passed as seen below, question options can be correctly linked to questions in our database.

Output:

```
Ran 1 test in 4.591s

OK
Destroying test database for alias 'default'...
```

11.3.4. Unit Testing

We also performed automated unit tests to test the functionality of individual modules in our system.

11.3.4.1 Registration Unit Test

To check if a user could register in our system, we created a user_a with arbitrary credentials and initialised user_a as a User within the Huf system. Following that, in the test_user_exists function, Django creates an empty test database with the same fields as our database and adds in user_a. The function then checks if user_a is correctly recorded in the database. Since the output of this test was OK, users can be recorded accurately in our database

Test Script 1:

```
User = get_user_model()
class UserTestCast(TestCase):

    def setUp(self):
        user_a = User(username='cfe', email='cfe@gmail.com')
        user_a.is_staff = True
        user_a.is_superuser = True
        user_a_pw = '12345678'
        user_a.set_password(user_a_pw)
        self.user_a_pw = user_a_pw
        self.user_a = user_a
        user_a.save()

    def test_user_exists(self):
        """ Testing if the system can record a new user """
        user_count = User.objects.all().count()
        self.assertEqual(user_count, 1)
        self.assertNotEqual(user_count, 0)
```

Output:

```
Ran 1 test in 2.116s

OK
Destroying test database for alias 'default'...
```

11.3.4.2 Password Unit Test

For this unit test, we were testing if the `check_password` function in our system accurately allows in users with the correct password and disallows users with the incorrect password.

Test Script 2:

Correct password script:

```
User = get_user_model()
class UserTestCast(TestCase):

    def setUp(self):
        user_a = User(username='cfe', email='cfe@gmail.com')
        user_a.is_staff = True
        user_a.is_superuser = True
        user_a_pw = '12345678'
        user_a.set_password(user_a_pw)
        self.user_a_pw = user_a_pw
        self.user_a = user_a
        user_a.save()

    def test_user_password(self):
        """ Testing if system records users' passwords correctly """
        self.assertTrue(
            self.user_a.check_password('12345678'))
```

As you can see above Django created a replica of our database and checked the result of a user trying to log in with the correct password. As can be seen below, since the output for this test is OK, user with the correct password are allowed access to our system.

Correct Password Output:

```
Ran 2 tests in 3.454s

OK
Destroying test database for alias 'default'...
```

Incorrect password script:

```
def test_user_password(self):  
    """ Testing if system records users' passwords correctly """  
    self.assertTrue(  
        self.user_a.check_password('WrongPassword'))
```

Above, the test_user_password function tests the output of a user trying to log in with the wrong password. Since the output for the test is a Fail, a user with the wrong password is not allowed access to our system.

Incorrect password output:

```
.F  
=====  
FAIL: test_user_password (loginapi.tests.UserTestCast)  
-----  
Traceback (most recent call last):  
  File "C:\Users\rohit_kurup\PycharmProjects\pythonProject5\CZ3003-HUF\loginapi\tests.py", line 20, in test_user_password  
    self.assertTrue(  
AssertionError: False is not true  
  
-----  
Ran 2 tests in 3.413s  
  
FAILED (failures=1)  
Destroying test database for alias 'default'...
```


Appendix A: Data Dictionary

Terms	Definition
User	A person with a registered account that is interacting with the system.
Creator	A user that creates a game.
Player	A user that is playing a game.
Quiz	A set of questions that is to be answered.
Game	Contains a set of quizzes that is created by a user.
Dashboard	A display of the details unique to each quiz.
Leader Board	A list to keep track of the top 5 scoring users.