Software Requirements Specification

for

Math Runner

Version 1.3 approved

Prepared by Team32

Pye Sone Kyaw (U1920416C)

Le Quang Anh (U1922940J)

Yin Ning (U1923120G)

Cai Xinrui (U1921389A)

Benjamin Goh (U1921615J)

Tran Hien Van (U1920891J)

Mark Tan (U1923902B)

Nanyang Technological University, Singapore

September 10, 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Team32	10/09/22	Initializing SRS	1.0
Team32	24/9/22	Add in Lab 2 Contents	1.1
Team32	7/10/22	Add in Lab 3 Contents	1.2
Team32	24/10/22	Add in Lab 4 and finalize contents	1.3

1. Introduction

1.1 Purpose

This document presents a development plan for Math Runner, an interactive game to gamify and socialize the teaching and learning of mathematical concepts for primary school students. In Math Runner, the student embarks on a race and the objective is to run into the correct answer option that is present on the race track.

The document will include, but is not limited to, a summary of full system functionality, which includes the functional and non-functional requirements, possible use cases, the data dictionary, use-case models, various diagrams such as class diagrams, sequence diagrams, dialogue map, and system architecture diagrams, the user interface of the application, and other appropriate diagrams such as data-flow diagrams.

1.2 Document Conventions

These are the conventions used in this SRS:

- Font type: Arial
- Font size (main header): 18
- Font size (sub header): 16
- Font size (sub-sub header): 14
- Font size (body): 12
- Line spacing: 1.5
- Bold letters represent arbitrary headers/important information in the text body.

In numbering our detailed requirements and use cases, we employ a nested (indented) numbering system where a title numbered 1 has subheadings 1.1, 1.2 and so on.

1.3 Intended Audience and Reading Suggestions

This document is intended for the following types of readers:

1. Project managers
2. Current and future developers
3. Quality Assurance (QA) personnel who are also designing test cases and performing tests on the application
4. Testers
5. Users

The document is organized as follows: Section 1 introduces the structure of the document and provides a brief overview of the software system being developed. Section 2 gives an overview of the product, including its perspectives, functions, target users, operating environment, design and implementation constraints and assumptions. Section 3 gives the interfaces of the product. Section 4 introduces the system features, including the functional requirements, use case model, sequence diagrams, class diagrams, system architecture, design patterns and dialogue map. Section 5 describes the non-functional requirements of the system. Section 6 showcases the architecture of the system. Section 7 presents the subsystem designs. Section 8 presents the various kinds of testing done for the system.

Project managers should read through all sections of the document to get a comprehensive understanding of the project, its overview, requirements, features, system design and acceptance tests involved.

Developers should primarily focus their attention on the overview, system features, non-functional requirements, and system design. They may also read through the tests involved to better target their efforts towards passing the testing done.

For quality assurance personnel, they should focus on the system features, non-functional requirements, and testing to carry out their duties in enforcing a certain level of quality in the code and performance of the system.

Testers should focus on the system features, non-functional requirements, and testing to be aware of the tests that needs to be carried out or have been carried out. They can then figure out which tests can be improved, redone, modified, or added.

Users might be interested in examining the main functionality of the application. As such, users can mainly read the overview and the system features sections of the report.

1.4 Product Scope

Math Runner is a game developed in Unity to gamify and socialize the teaching and learning of mathematical concepts for primary school students and teachers. Math Runner allows for a more interactive and digitalized delivery of teaching and learning deliverables from teachers to students, while at the same time equipping teachers with automated data analysis tools for them to track and gain insights on students' progress, individually or as a group.

Math Runner also fosters a more engaging learning experience for students where they are not only able to play games to complete their assignments but also compete against and challenge one another and see their scores on a leaderboard. Both the assignments and challenges are implemented in the form of custom levels in Math Runner and can be created by either teacher or students.

1.5 References

To aid in the development of the project, the follow documentations are being referenced:

- Unity – https://docs.unity3d.com/Manual/index.html
- Firebase – https://firebase.google.com/docs

2. Overall Description

2.1 Product Perspective

The application is an educational game that aims to teach students mathematical concepts. Math Runner is meant to be used as a supplementary tool for teaching and learning of mathematics and not as a substitute for in-class lessons by a trained mathematics educator.

World-Level

Math Runner will be split into several different worlds, each representing different kinds of mathematical concepts. There are 4 worlds in total: Addition, Substraction, Multiplication and Division.

Within each world, there will be several levels, each of which represents topics from that concept category, which can range from basic to advanced topics. For example, within the addition world, there can be several levels such as ‘addition with numbers up to 10’ or ‘addition with numbers up to 1000’.

Each level itself will have questions of varying difficulties. The difficulties are dynamically adjusted based on real-time analysis of the player’s performance within the level. If the player is answering questions correctly consecutively, then the player is deemed to have good mastery of the content, so the difficulty is increased. The difficulty corresponds to how long the player has to respond to the question. The higher the difficulty, the faster the Game avatar runs, the shorter the time the player must pick an answer to the question.

Modes of Gameplay

There will be two modes of gameplay: single player mode and custom gameplay mode.

Single player mode is where a user plays by themself. The user will select the specific world and specific level, but progress through the difficulties sequentially according to how the game delivers the questions, i.e., no skipping questions or jumping difficulties manually.

Custom gameplay is for any student user to issue challenges to other student users. Teacher users can use this mode to assign assignments to students. As both serve the same purpose and only differ in the account type that is performing the function, they are combined together into the custom gameplay mode. The custom gameplay mode allows users to create custom levels with their own set of custom questions. These custom levels, once created, will generate a seed number that can be shared by teachers and students. For teachers, this custom level can act as an assignment for their students. For students, this custom level can act as a challenge level to compete with other students.

There is also a Telegram bot which can issue notice of new assignments in the form of custom levels to a Telegram group for teachers who have connected their accounts to the bot. This is to facilitate the sharing of the seed and the custom level from the teacher to their class group easily.

Actual Gameplay

In actual gameplay, the player will be controlling a character avatar who is ‘running’ on the screen in a horizontal orientation as seen below.

There will be option bubbles that appear on the running track and are coming towards the avatar. One of these options will be the correct answer to the question presented to the user at the top of the screen. The objective is for the user to move onto the correct lane and collide with the correct option bubble to score points.

2.2 Product Functions

The main functions of the game application are summarized in this section, separated by user classes and major components. Please note that these are not the detailed functional requirements but rather a high-level overview of the functions only.

Overall game

- The game will have several worlds, each corresponding to a concept category such as ‘addition’ and ‘subtraction’
- Each world will have several levels, each corresponding to a specific topic under the concept category for that world, such as ‘addition for numbers less than 10’
- Each level will have questions of different difficulties. Difficulty within the level is determined by how long the player has to respond to the question. Difficulty is increased when the player answers questions correctly continuously, and difficulty is reset when the player answers a question wrongly.
- There will be two modes of gameplay: single player gameplay mode, and custom gameplay mode

All General Users

- All users can register an account for the game
- All users can login to the game

User Group 1: Students

- Students can advance through the game by answering questions correctly
- Students can play the game on their own
- Students can complete assignments assigned by teachers
- Students can see his/her place on the leaderboard

- Students can play custom levels made by another student
- Students can choose avatars before the game starts to customize their character

User Group 2: Teachers

- Teachers can assess the performance of players via analytics page
- Teachers can track the progress of all students in a one-page view
- Teachers can play games in all worlds and all levels and custom level games
- Teachers can see the leaderboard

Gameplay

- Student's character will be running in the game (hence Math Runner)
- Question will be displayed on top section of screen
- Three options to answer the question, one of which is the correct option, will come up on the screen
- Student's character must collide with the correct answer option that comes up on the lane (see user interface mockup section for more) to score points and move up a progress bar
- Student must answer a certain number of questions to reach the end of the progress bar
- Student must answer a certain number of questions correctly for the game to consider the level to be completed and to unlock the next level

Leaderboard

- Users can see the overall leaderboard
- Users can see world-specific leaderboard
- Users can see level-specific leaderboard
- Users can see custom level-specific leaderboard

Data Analysis

- The game will analyze each student's playing history to determine a student's understanding of concepts in real-time

- The difficulty of the game is personalized according to the real-time data analysis done by the game, to challenge the student at a higher intellectual level.
- The system will output a report for teachers that will reflect the level of understanding for items on a list of pre-defined concepts for different students

Custom Levels

- Users can create custom levels with custom questions of their own creation.
- Users will be presented with a seed code generated after saving their custom levels and questions.
- Users can send the seed code to students to play this custom level, which constitutes as an assignment if given by the teacher, and a challenge if given by a student

2.3 User Classes and Characteristics

The two main user classes are ‘student’ and ‘teacher.’

2.3.1 Student

A ‘student’ refers to a player user in this game and has a ‘student’ account created. The student is expected to be within the primary school educational level of knowledge and age range.

Students are using Math Runner to learn mathematical concepts in an interactive and engaging manner that would in turn have result in better knowledge absorption and application compared to traditional learning methods.

They are also able to create custom levels and share them with other students, play custom levels from teachers and other students, and look at the leaderboard to see their ranking amongst all users playing the game.

2.3.2 Teacher

A ‘teacher’ refers to a user who has a ‘teacher’ account. The teacher account is akin to an administrator role in the game and can track student accounts in terms of their progress and

performance. The teacher account would be the highest privileged user account in the system, and teachers are expected to know how to interpret the data analysis report presented to them.

Through tracking the students' activities, they are also able to determine students' understanding of the content in terms of a list of pre-defined concepts, so that in-person classes can be adapted accordingly to focus on certain topics or concepts that students are weak in.

2.4 Operating Environment

- Operating platforms for game applications
    - Development platform: Unity
    - Operating system platform: Windows and Mac OS
- Database service
    - Firebase

2.5 Design and Implementation Constraints

- As there is no funding provided for the development of this application, developers will have to use either free services or self-fund services, such as databases and hosting platforms, if any.
- The game engine used for development is Unity, which uses C# for development.
- Unity version used for development is 2021.3.8f1.
- Firebase is used for backend components such as database and authentication
- The database schema is constructed to allow for easy addition of new components and features to the game without requiring a massive overhaul.
- The version of Firebase SDK for Unity used is 8.2.0.

2.6 User Documentation

User documentation, specifically user manuals and troubleshooting steps will be provided along with the software. Documentation will be accessible for download through the product's website. There will be separate user manuals for teachers and students so as to prevent potential abuse of the system by students with knowledge of the teacher accounts' capabilities.

The user manuals would expound on the features of the product and provide detailed step-by-step instructions on setting up the product for both teachers and students. There would also be a frequently asked questions (FAQ) section, and a troubleshooting section in the manuals.

Telegram bot for teachers will be setup during onboarding of the teacher onto the game during a physical in-person training session.

2.7 Assumptions and Dependencies

Connectivity

- The application assumes that users always have access to the internet while playing the game. Internet connectivity is required for the login, leaderboard, and custom level sharing features of the game. Should players be disconnected from the game before completing a game, the progress of that game will be lost.

User Device Hardware Capability

- Users should have a Windows computer capable of running the game. The game is expected to not be resource intensive, so any modern Windows computer would be capable of running the game smoothly.

Authentication

- The game would require users to log in using an email and a password tied to each user's account. The assumption is that users possess an email address for password recovery and had kept a record of their account IDs.

Digital and Language Literacy

- Users must be digitally literate and know how to operate a digital device without assistance. Users must be able to comprehend English, as the language for the game is English.

3. External Interface Requirements

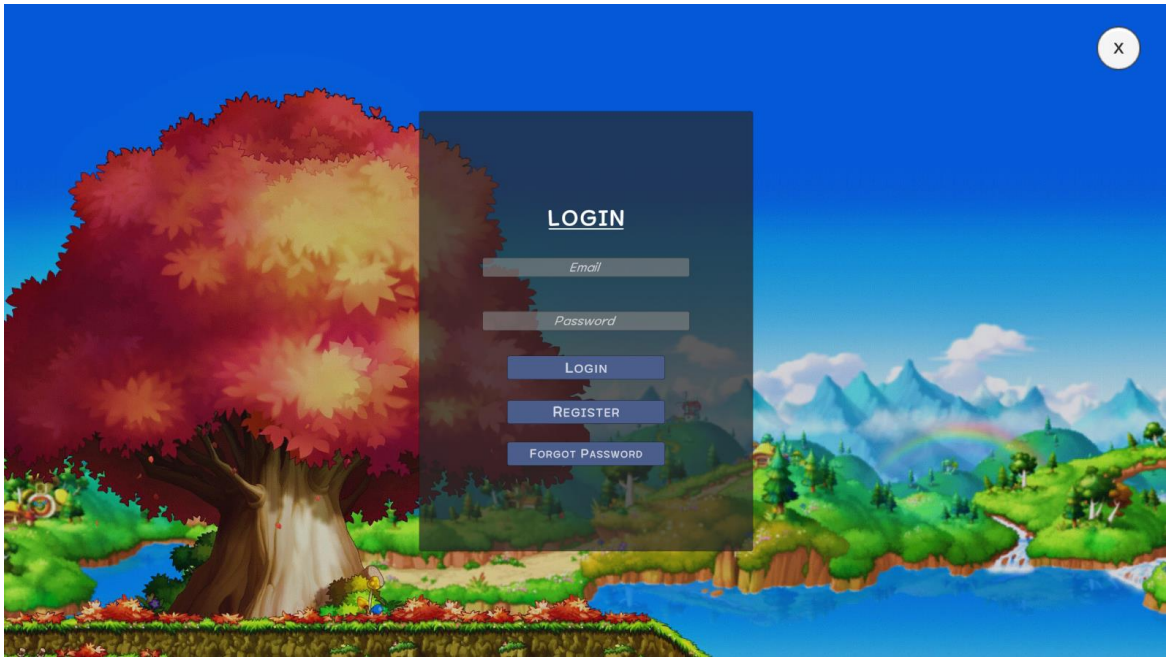
3.1 User Interfaces

The following sections in this chapter will contain the user interface of the game, albeit at an early stage of development. The purposes of these screenshots is to facilitate the development of the game and aid in understanding the flow of the team for readers.

All artwork and assets used for the game were obtained from copyright-free, royalty-free sources.

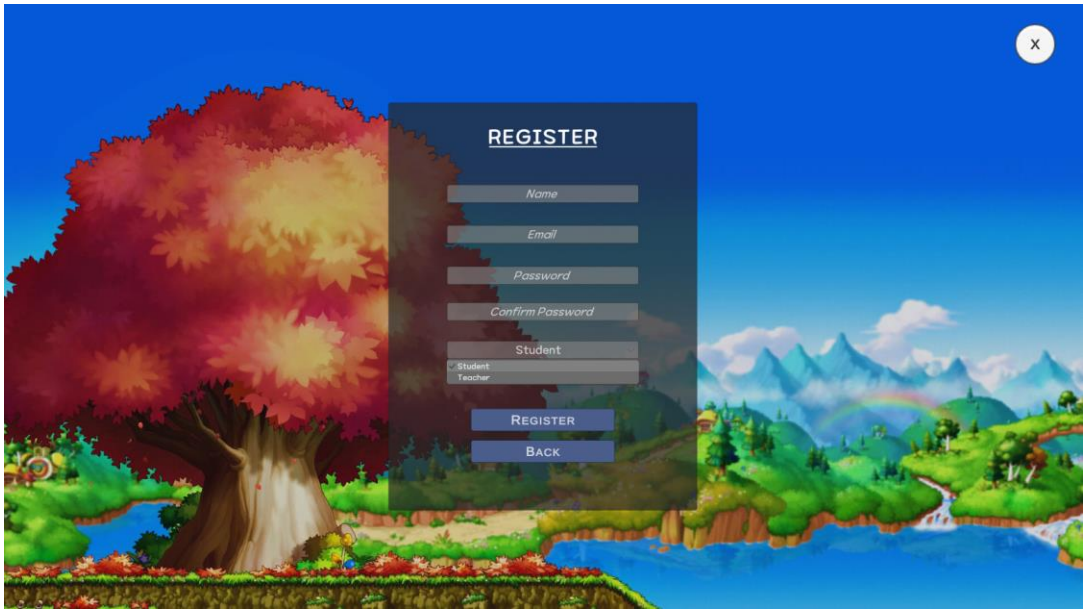
3.1.1 Landing/Login Page

The landing page allows users to choose to either login or create an account.



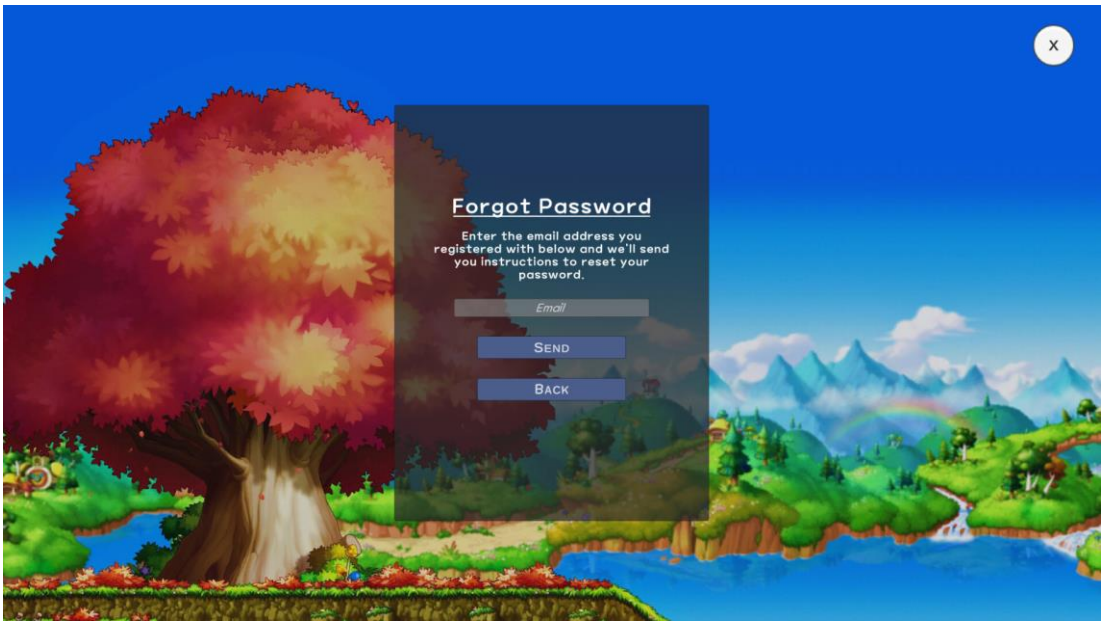
3.1.2 Register Account Page

The create account page allows users to create an account using their email, password, and access code for teachers



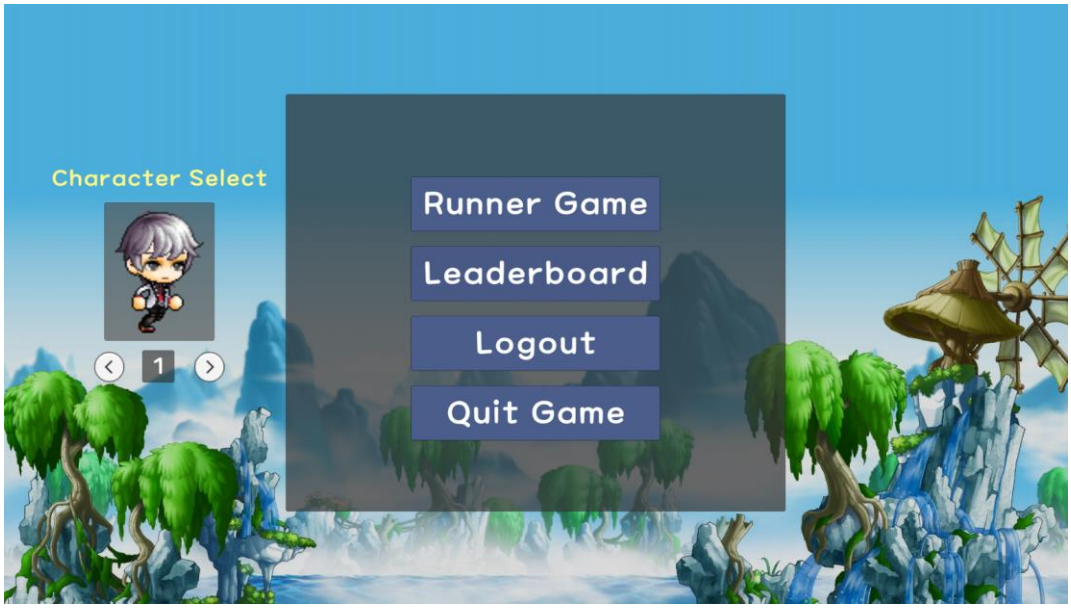
3.1.3 Forgot Password Page

The forgot password page allows users to reset their passwords if they have forgotten them.



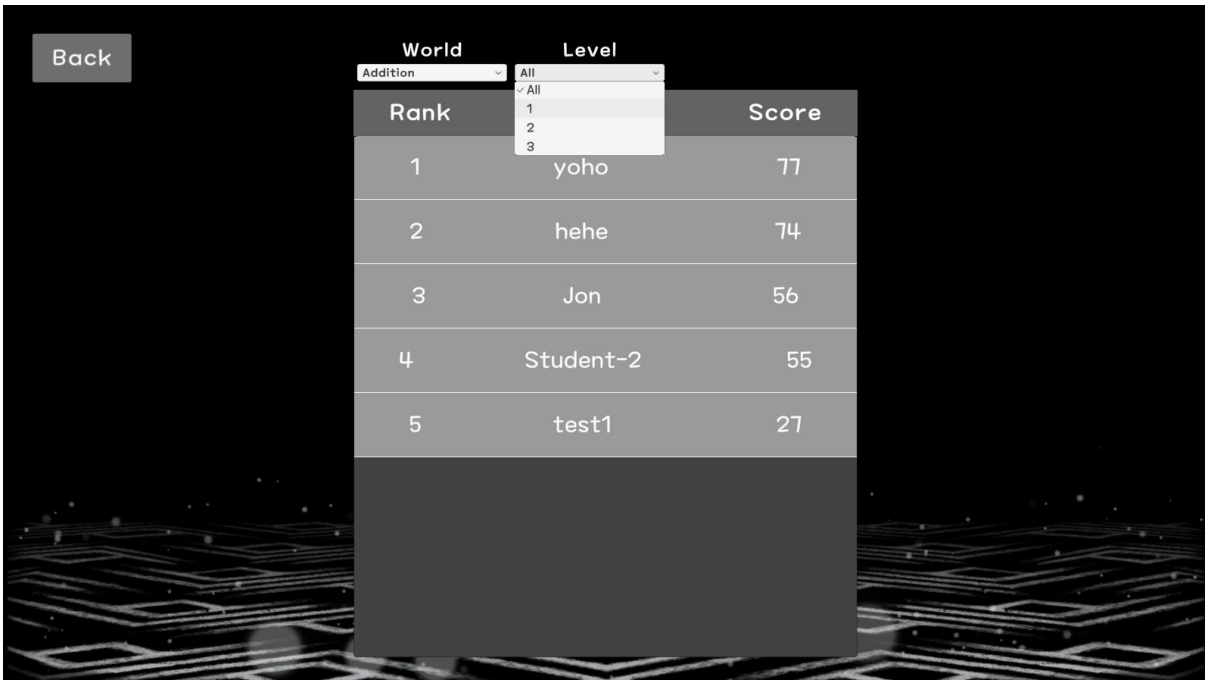
3.1.4 Student – Main Menu Page

The student main menu page allows students to play the game modes, view leaderboards, edit their profile or settings, logout, or quit the game. It also allows the user to select the desired character/avatar for the game.



3.1.5 Leaderboard Page

The leaderboard page allows users to view the leaderboards for various worlds/levels.

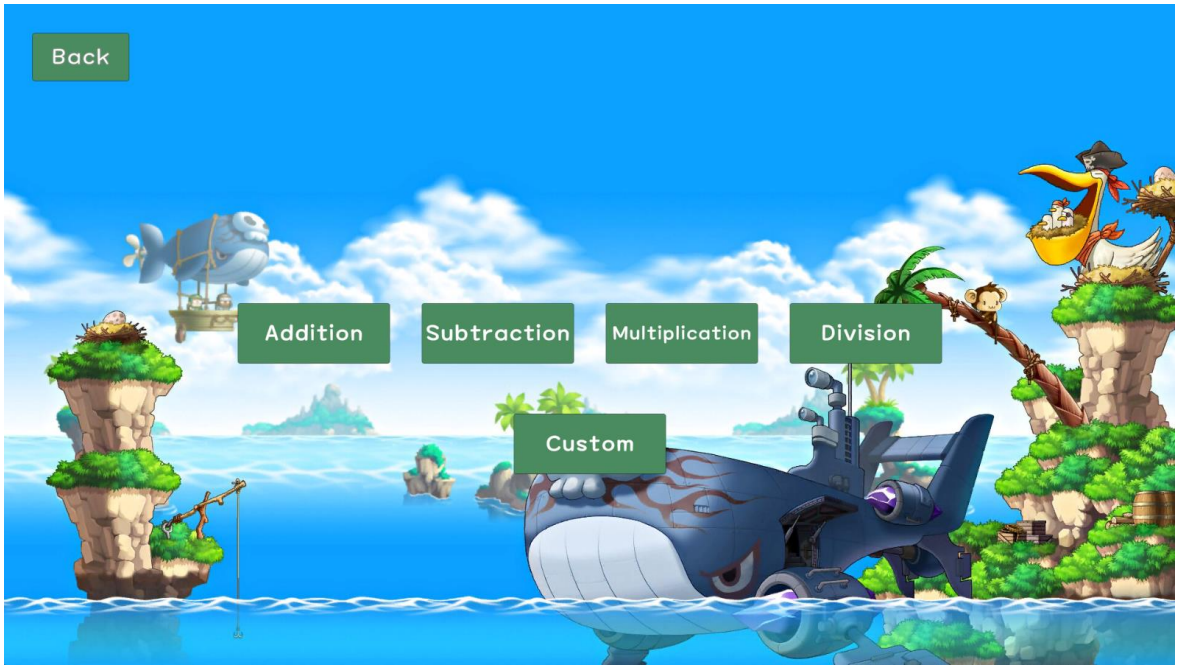


The leaderboard page also allows users to view the leaderboards for the various custom levels. Teachers can use this to track who has done the custom level assignments, together with the data analytics function.

The score displayed on the leaderboard corresponds to the total score of the level or the world, depending on what filter is set by the user.

3.1.6 Student – Single Player Mode – World Selection Page

The world selection page allows students to choose which world they wish to play



3.1.7 Student – Single Player Mode – Level Selection Page

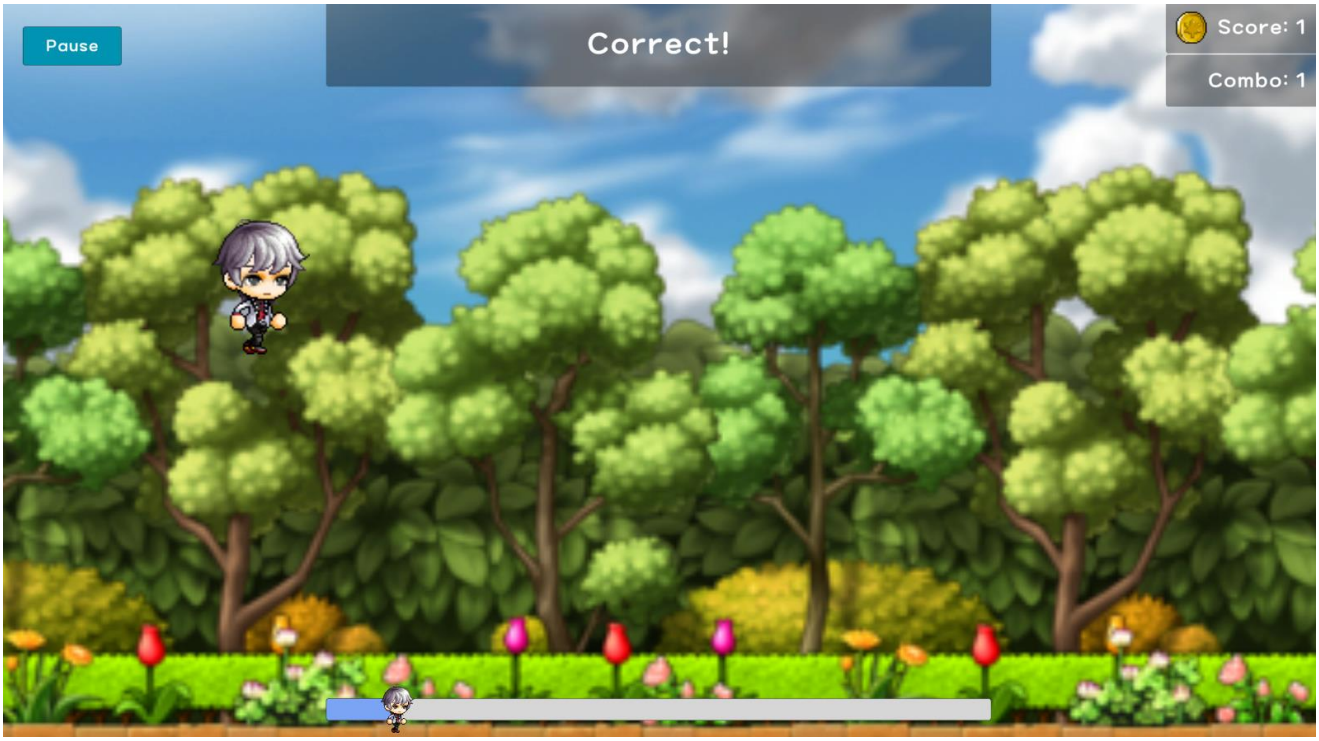
The level selection page allows users to choose which level they wish to play. Levels are unlocked (translucent green to solid green) when the user finishes prerequisites levels

3.1.8 Student – Single Player Mode – Gameplay Page

The gameplay page is where students actually play the game, which is a horizontal runner game. Students are supposed to run into one of the options (9, 15, 12 in the image above) which they deem to be the correct answer (“9” in this case) to the question presented to them (“What is 5 + 4?” in this case). The students can move the avatar on the screen using the arrow keys and pause the game by pressing the pause button on the top left side of the screen.

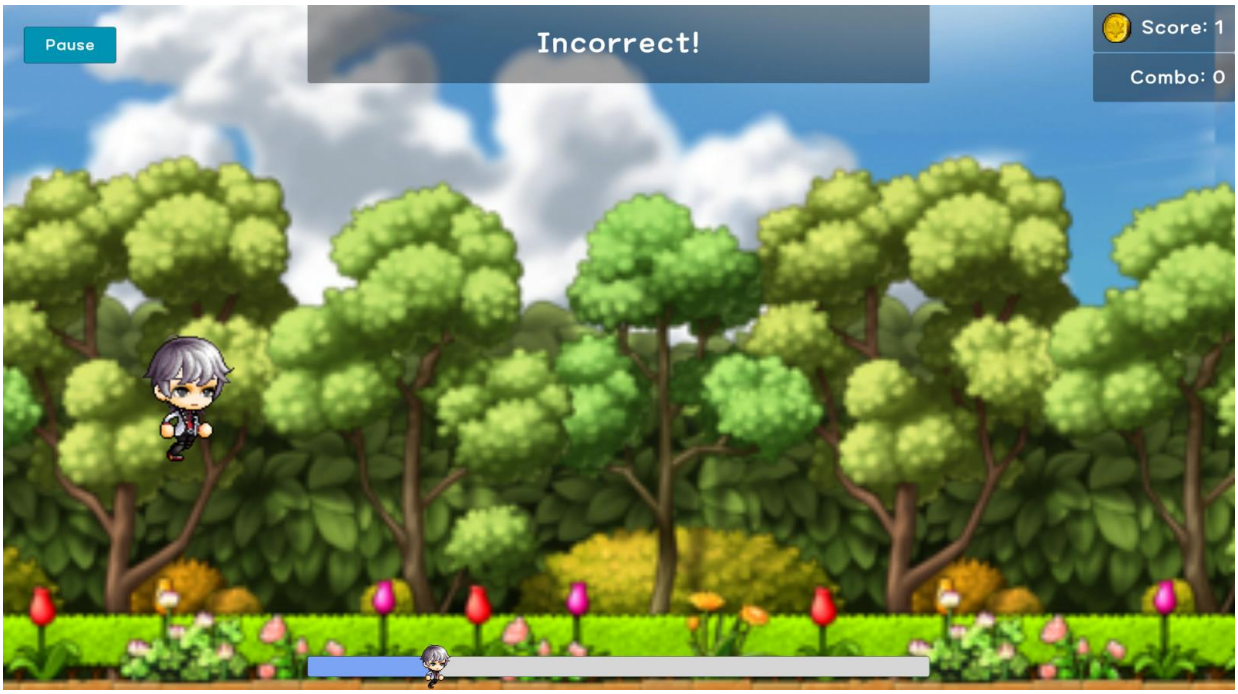
Upon answering questions correctly, the user’s avatar will move to the right on the progress bar, signifying progress through the level. The progress bar can be found at the bottom of the gameplay page. By answering questions correctly, the combo value also increases. The score of the level for the player is incremented by the combo value each time the player answers a question correctly. These game mechanics can be seen in the screenshot on the next page.

The speed of the game/horizontal scroll will also increase according to the combo value, which signifies the dynamic difficulty ‘levels’ of the game, and the real-time performance analysis aspect of the game.



The eventual end goal is the reach the end of the progress bar which will signify completion of the level. When they complete the level, they will be shown the success message as seen below.

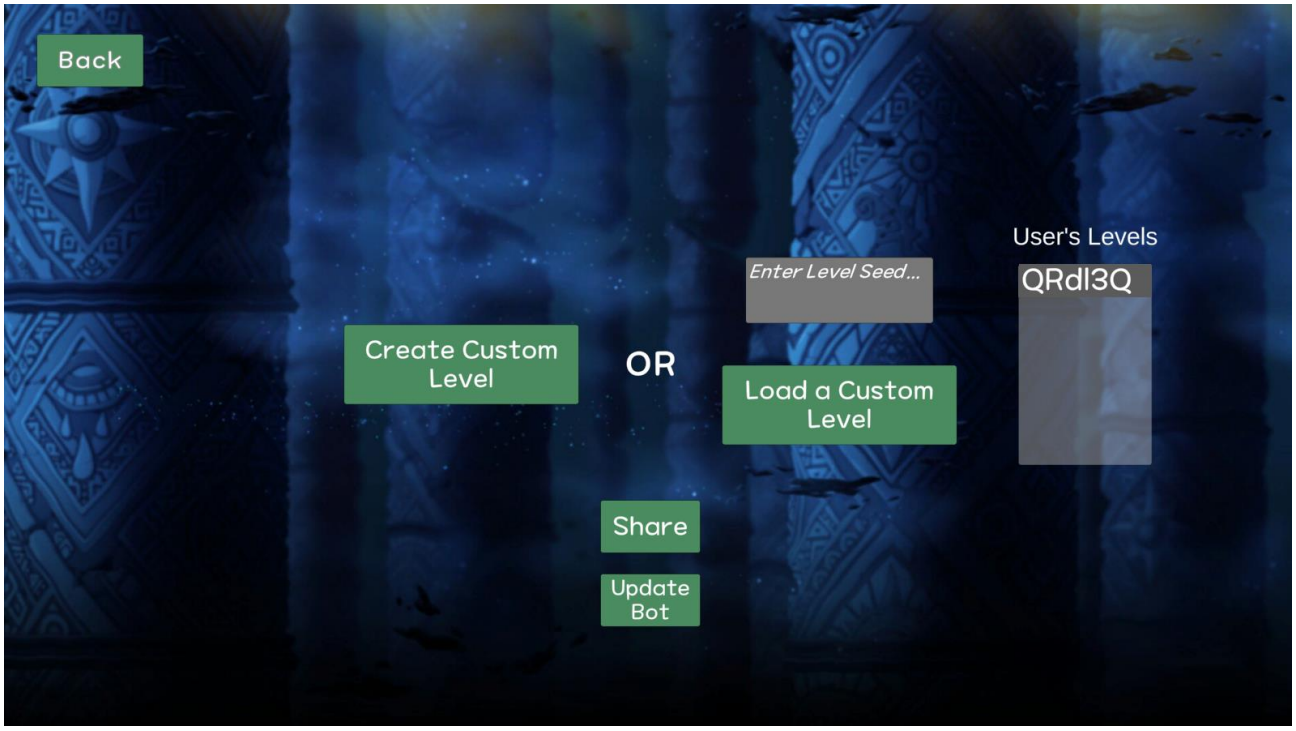
In the event of an incorrect answer, the combo value will be reset to 0, and the speed/difficulty will also reset to 0, as seen below.



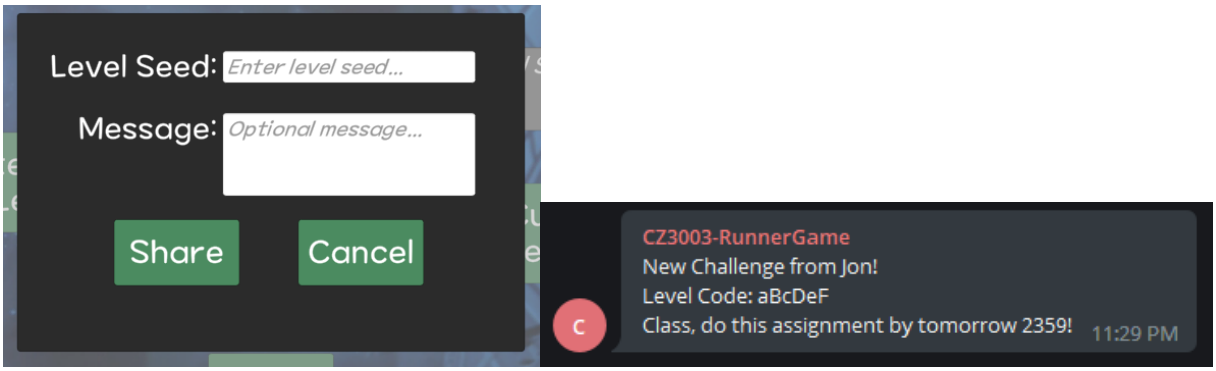
The next level of the world will only be unlocked if the player answers at least half the questions correctly.

3.1.9 Custom Level Mode – Landing Page

Custom mode landing page for users allows them create a custom level or load a custom level by entering its seed. Through this loading of custom level, the user can complete assignments given by their teachers or challenge levels made by their peers.



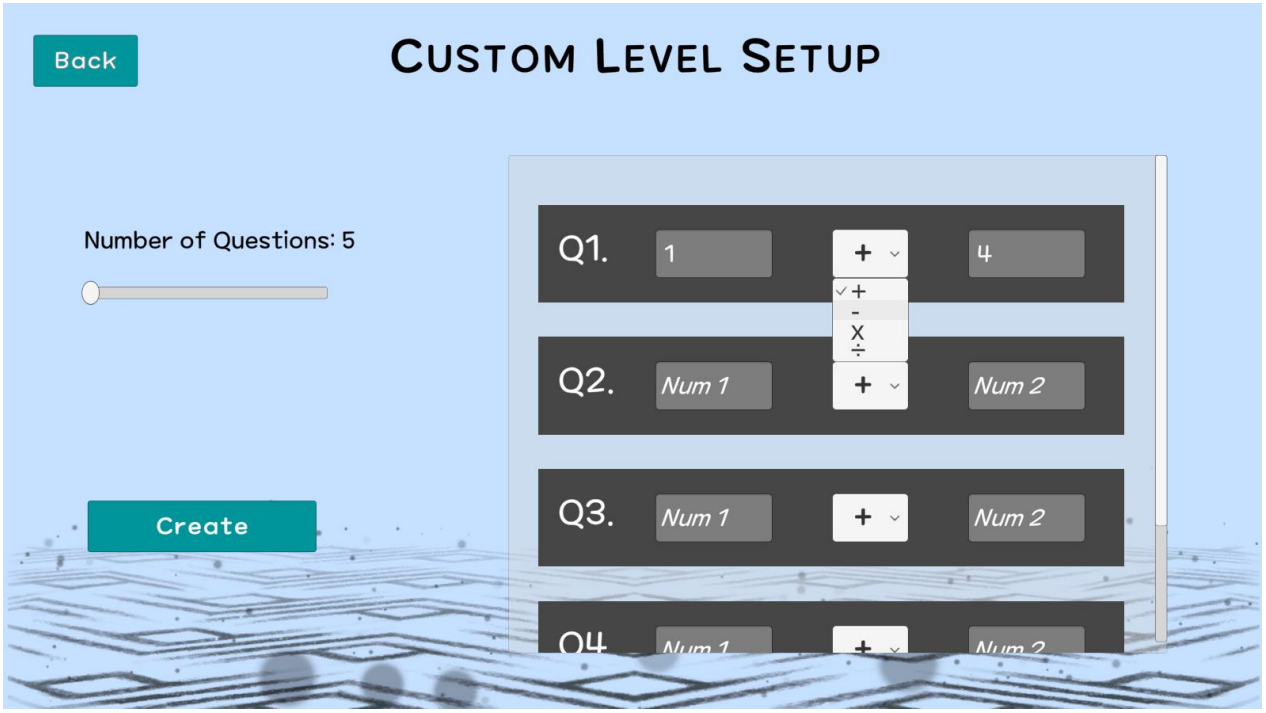
Users can see the seeds of the previous custom levels that they created on the right and clicking on the level will copy the seed to the clipboard for easy sharing. Users can also share the custom level's seed by clicking the share button, which will show the popup dialog as seen below. The user needs to enter a valid seed and can optionally enter a message to be shared along with the seed.



This message will be shared via the Telegram bot as seen above.

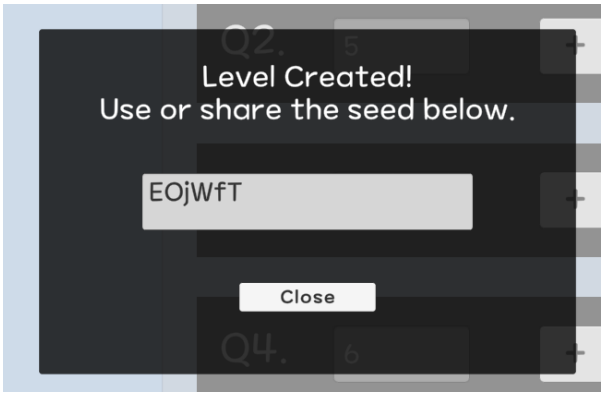
3.1.10 Custom Level Setup Page

The custom level setup page is where the user provides the number of questions in the level, which will populate the scrollable question list on the right. The user can then edit each of the questions by choosing the mathematical operation and the operands of the question. The options for the question will be generated by the game itself. This functionality is to aid in usability of the custom level maker function, as users might enter incorrect or invalid answer options if it is left modifiable to the user.



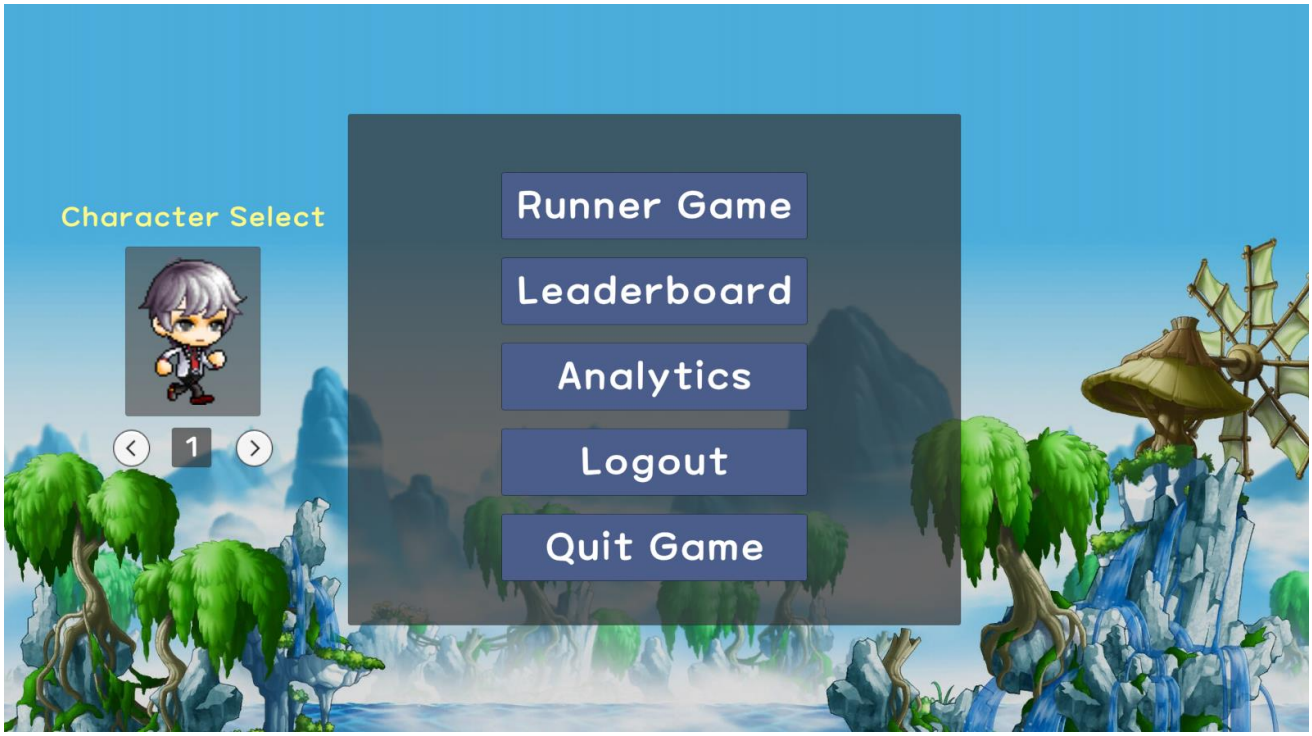
3.1.11 Custom Level – Confirmation Popup

This page allows users to take note of the custom level seed generated.



3.1.12 Teacher – Main Menu Page

The page allows teachers to view data analytics reports, make, track or send out assignments, view leaderboards, edit settings, or logout

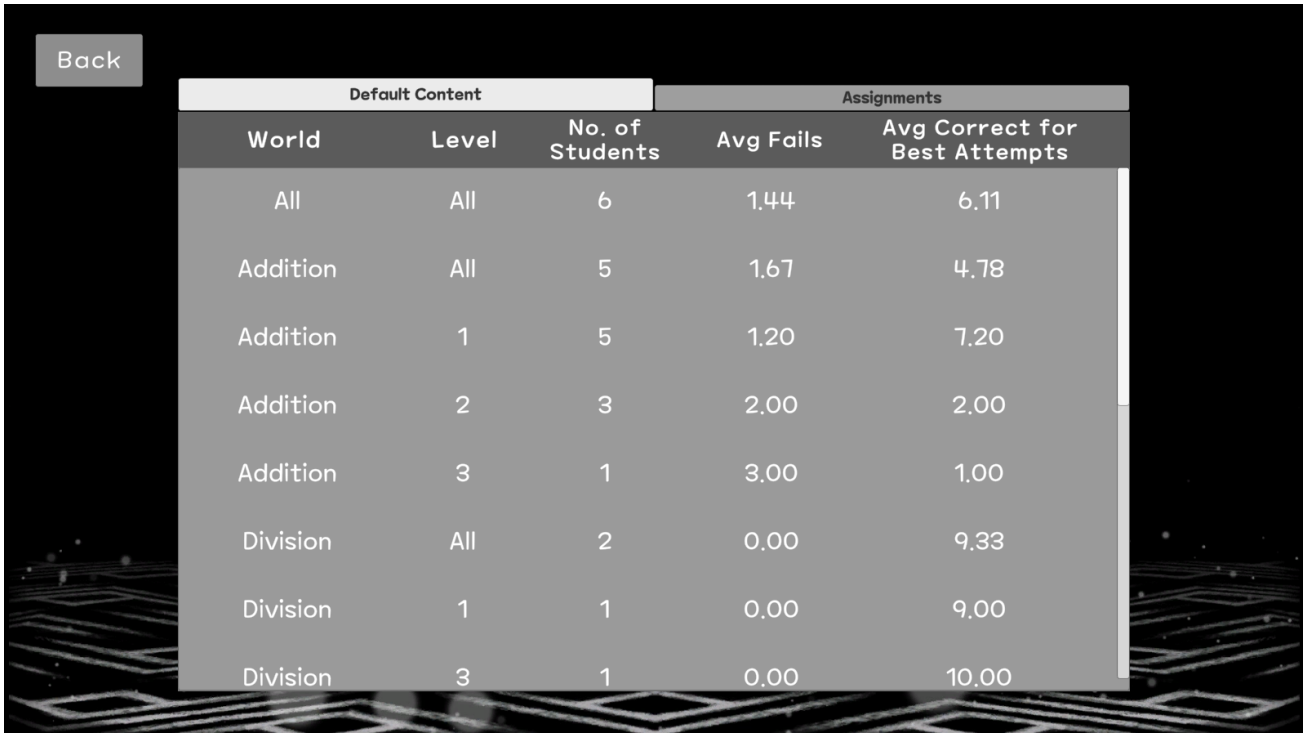


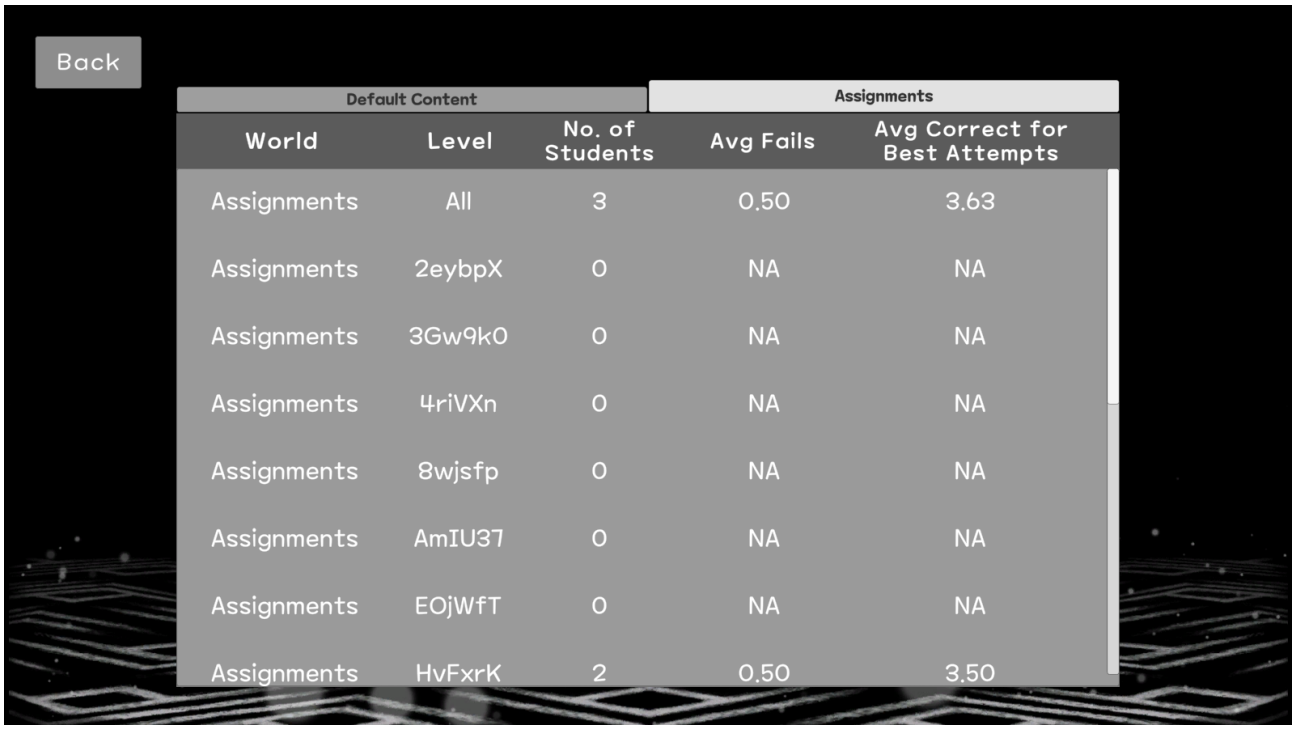
3.1.13 Teacher – Analytics Landing Page

The data analytics page allows the teacher to view/track progress/mastery of the students for various worlds/levels/custom levels at one shot. The statistics displayed are as such:

- Number of students who have attempted the world/level/custom level
- The number of average fails for that world/level/custom level – Fail corresponds to how many did not achieve a passing score of answering more than half the questions correctly
- The average number of questions answered correctly for that world/level/custom level

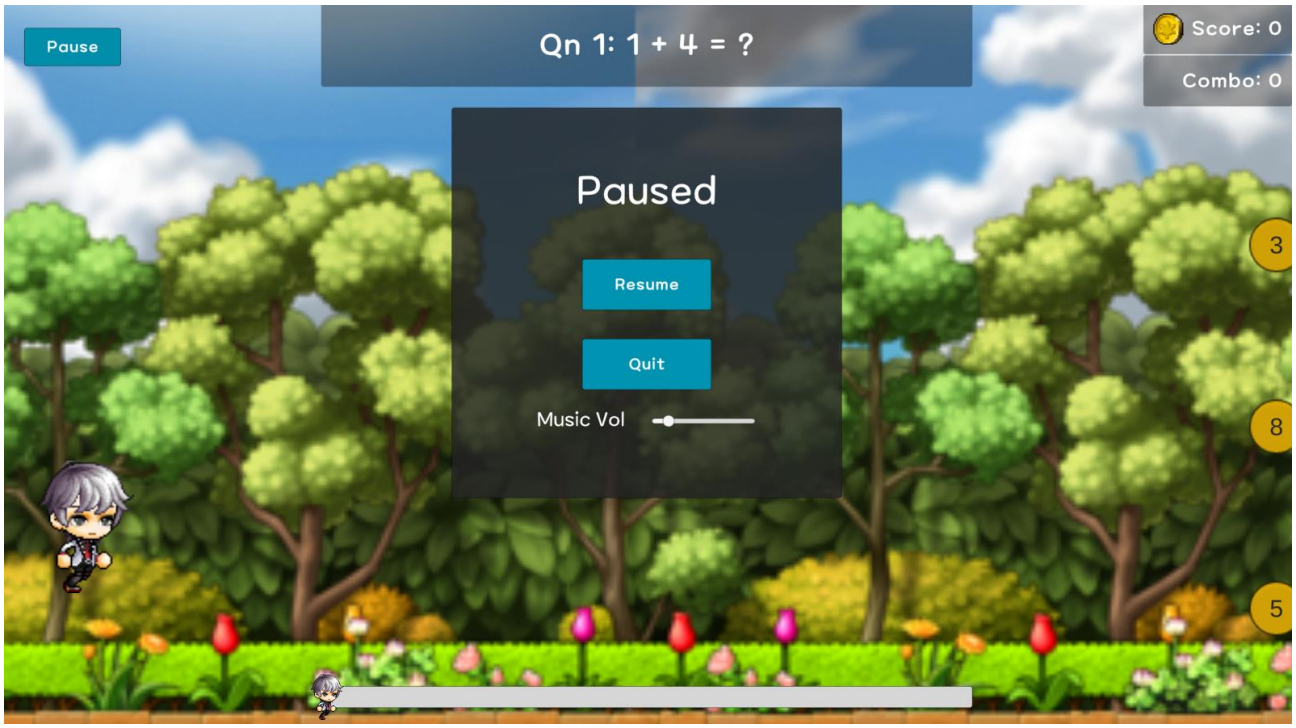
Together with this page and the leaderboard page, the teacher can make sense of the data and decide on the next course of action for the class.


Back
Default Content
Assignments
World
Level
No. of Students
Avg Fails
Avg Correct for Best Attempts
All
All
6
1.44
6.11
Addition
All
5
1.67
4.78
Addition
1
5
1.20
7.20
Addition
2
3
2.00
2.00
Addition
3
1
3.00
1.00
Division
All
2
0.00
9.33
Division
1
1
0.00
9.00
Division
3
1
0.00
10.00


Back
Default Content
Assignments
World
Level
No. of Students
Avg Fails
Avg Correct for Best Attempts
Assignments
All
3
0.50
3.63
Assignments
2eybpX
0
NA
NA
Assignments
3Gw9k0
0
NA
NA
Assignments
4riVXn
0
NA
NA
Assignments
8wjsfp
0
NA
NA
Assignments
AmIU37
0
NA
NA
Assignments
EOjWfT
0
NA
NA
Assignments
HvFxrK
2
0.50
3.50

3.1.14 Pause Page

This page allows users to change the game volume, resume the game, or quit the level.



3.2 Hardware Interfaces

There are no hardware interfaces beyond the use of personal computers to access the game. If the device being used to access the game has modern computer capabilities and an internet connection through wired or wireless means, the game will function as intended.

3.3 Software Interfaces

Unity – Game Engine

Unity is a cross-platform game engine developed by Unity Technologies. The engine supports a variety of desktop, mobile, console and virtual reality platforms. For the game's purposes, the platform of focus will be desktop. Unity uses C# as the programming language, but this will be opaque to the end user.

Firebase – Cloud Firestore

Cloud Firestore is a NoSQL database that is hosted on the cloud by Firebase by Google that acts as the primary database for the game. The data stored within the database are records of each player's historical progress through the game, as well as questions, answers, and options for each question.

The communication between the database and the game is done through the backend server, using the Firebase library. The frontend may call the backend server through various API endpoints to fetch/update data from the database, with each endpoint meant for specific purposes.

3.4 Communications Interfaces

HTTPS Communication (Game to backend databases)

- HTTPS is HTTP encrypted with TLS. This protects against man-in-the-middle attacks and allows for secure communication between parties. This also prevents any possibility of eavesdropping and tampering with or forging the contents of the communication.

4. System Features

In this chapter, the system features will be presented. First, the functional requirements will be presented. This will be followed by a use-case diagram to show the various actors and use cases in the system, and then with detailed and specific use-case descriptions. This will be superseded by other relevant diagrams and analysis.

4.1 Main System Functions

4.1.1 Account Creation

4.1.1.1 Description and Priority

- Handles the account creation procedure so that users have an account to login in order to play the game.
- Highest priority as users must have an account to play/manage the game.

4.1.1.2 Stimulus and Response Sequences

User Action	**System Responses**
User clicks 'Register' button on user interface	Game directs user to account creation form page
User enters valid input in the account creation form	• Game performs successful validation of input fields • Game displays a message stating successful account registration
User enters invalid input in the account creation form	Game displays a message stating failed input fields validation

4.1.1.3 Functional Requirements

1 The game must allow users to create accounts

1.1 The account creation must either be a student account or teacher account

1.1.1 The user must choose either the student or teacher account option from the dropdown list in the account screen to indicate the type of account

1.2 User must provide a username during account creation

1.3 User must provide an email address during account creation

1.3.1 The game must validate the email address entered by the user

1.3.2 The email address must not already be in the game's account database

1.3.3 The email address must have an '@' in it

1.4 User must provide a password during account creation

1.4.1 The password provided must meet the minimum complexity requirements

1.4.1.1 The password must be at least 8 characters long and consist of at least one uppercase alphabet, one lowercase alphabet, one number, and one special symbol.

1.4.2 The password must be entered a second time for confirmation

1.4.3 The passwords entered during the two instances must match

1.5 To create a teacher account, the user must select the teacher account option and enter a valid school access code.

1.5.1 The access code provided must be validated by the game

4.1.2 Account Login

4.1.2.1 Description and Priority

- Allows users to access the game upon successful authentication of login credentials
- Highest priority as users will need to successfully login to access the game

4.1.2.2 Stimulus and Response Sequences

User Action	**System Responses**
User enters valid input in the login form and clicks 'Login' button on user interface	• Game performs successful validation of input fields • Game directs user to main menu page of the respective account type
User enters invalid account email	Game displays a message stating that the account does not exist
User enters valid account email and invalid password input in the login form	Game displays a message stating failed password validation

4.1.2.3 Functional Requirements

2 The game must allow users to log in to their own accounts

2.1 The game must present a login form for the user to enter their email and password

2.1.1 The game must validate the email entered by the user

2.1.2 The game must authenticate the login credentials using its authentication subsystem

2.1.3 In the event of failed authentication, the game must display an error message informing the user of the failed authentication attempt

4.1.3 Account Logout

4.1.3.1 Description and Priority

- Users can log out of their currently logged in accounts. This is useful for cases where the device being used is a shared device, and other users may wish to log into their own accounts to play the game
- Priority: Medium/Low

4.1.3.2 Stimulus and Response Sequences

User Action	**System Response**
User clicks on ‘Logout’ button on the game user interface at the main menu page	• Game goes back to login menu screen • Game ends current user’s session and removes any non-persistent data that might have been generated during the user’s session

4.1.3.3 Functional Requirements

3 The game must allow users to log out of their accounts by clicking the logout button

3.1 The game must end the user’s session and remove any other non-persistent data related to the terminated session

4.1.4 Forget/Reset Password

4.1.4.1 Description and Priority

- Users may forget the password they set for their account, and the game should let them reset it. To reset the passwords, users need to go to their email account, proceed to the link in the email and enter a new password at the link's destination page
- Priority: Medium

4.1.4.2 Stimulus and Response Sequences

User Action	**System Response**
User clicks on 'forget password' button at login menu	Game directs them to "forget password form" page
User enters valid input (valid email address which can be found in the existing user database) in the "forget password form" page	• Game performs successful validation of input fields and sends a new one-time password to email address • Game displays a message stating successful sending of one-time password to email to reset password
User enters invalid email address input in the "forget password form" page	Game displays a message stating failed email address validation
User enters valid new password in the "reset password form" page	• Page performs successful validation of input fields • Page displays a message stating successful reset password
User enters invalid new password in the "reset password form" page	Game displays a message stating failed input fields validation, asking user to enter a new password that meets the minimal complexity requirement

4.1.4.3 Functional Requirements

4 The game must allow users to reset their passwords if they forget their passwords

4.1 The game must present a “forget password form” page for the user to enter their email

4.1.1 The game must validate the email entered by the user

4.1.2 The game must ensure that the email exists in the database

4.1.3 If the email entered is valid and exists in the database, the system must send an email containing a password reset link to the user’s email.

4.1.4 The game must show a success or failure message depending on whether the email validation and checking was successful or not

4.2 The email must contain a link for the user where the user can reset their password

4.2.1 The destination page of the link must validate the password entered by the user

4.2.2 The new password entered by the user must meet the minimum complexity requirements

4.2.3.1 The password must be at least 8 characters long and consist of at least one uppercase alphabet, one lowercase alphabet, one number, and one special symbol.

4.1.5 Student – Main Menu

4.1.5.1 Description and Priority

4.1.5.2 Stimulus and Response Sequences

User Action	System Response
User selects ‘Runner Game’	Game directs user to ‘World Selection’ page
User selects ‘Leaderboard’	Game directs user to ‘Leaderboard’ page
User selects ‘Logout’	Game logs user out (See 4.1.3)
User selects ‘Quit Game’	Game exits

4.1.5.3 Functional Requirements

5 The main menu must allow students to choose which action they wish to perform

5.1 The game must direct the student to the selected page/screen/action

4.1.6 Single Player Mode

4.1.6.1 Description and Priority

- This feature allows users to play the single player mode of the game
- Very high priority as this is one of the main modes of gameplay

4.1.6.2 Stimulus and Response Sequences

User Action	System Response
User selects a world to play	Game directs user to the level selection screen
User selects a level to play	• Game directs user to the gameplay screen • Game loads first question for user to answer
User moves onto the correct running lane and collides with the correct answer option	• Game shows correct answer notification

	• Game adjusts combo value and score value, and adjusts the speed/difficulty of the level based on the combo value
User moves onto the incorrect lane running lane and collides with the wrong answer option	• Game resets the combo value to 0 and resets the speed to the lowest level • Game shows incorrect answer notification
User answers all the required answers	• Game shows complete level notification • Game brings user back to level selection and updates with new levels unlocked (if any)

4.1.6.3 Functional Requirements

General

6 The game must allow users to play the single player mode

6.1 The game must direct users to the single player mode screen upon clicking the single player mode button.

6.2 The game must have two tiers of hierarchy: world, and level.

Worlds

7 The game must have several worlds for the user to choose from to play

7.1 Each world must represent a unique concept category to learn, specifically addition, subtraction, multiplication, and division.

7.2 Students must be able to select which world they wish to play.

Levels

8 Each world must have several levels for the user to choose to play.

8.1 Each level must represent topics from that world's concept category, which can range from basic to advanced topics.

8.2 Users must be able to select which level they wish to play from a selection of unlocked levels

8.3 Subsequent levels must be unlocked after completing pre-requisite levels

8.4 Levels must be deemed to be completed once the user has reached a certain level of mastery and progress, which is determined by the internal progress subsystem

Gameplay within level

9 Within each level, users must answer questions to proceed with the next question.

9.1 For each question, users must be presented with three options to choose from, one of which is the correct option

9.2 For each question, upon the user confirming their choice of answer by running into the answer on the track, the game must display feedback based on whether the choice of answer was correct or wrong.

9.3 Based on the correctness of the answer, the game must adjust the score and the combo value

9.3.1 The score is incremented by the existing combo value (before update) if the option chosen is correct

9.3.2 The score does not increase if the option chosen is wrong

9.3.3 The combo value is incremented by 1 if the option chosen is correct

9.3.4 The combo value is reset to 0 if the last option chosen is wrong

10 Within each level, there must be a dynamic difficulty system

10.1 Users must not be able to choose the difficulty level, and instead the game dynamically adjust difficulty based on how the user is performing for the questions through real-time data analytics

10.2 The difficulty corresponds to the current combo value of the level. The higher the combo value, the higher the difficulty

10.3 The difficulty corresponds to the speed of the horizontal scroll of the game. The higher the difficulty, the faster the horizontal scroll, the shorter the time the player has to answer the question

4.1.7 Custom Level Mode – Playing

4.1.7.1 Description and Priority

- This feature allows users to play the custom level mode of the game
- Very high priority as this is one of the main modes of gameplay

4.1.7.2 Stimulus and Response Sequences

User Action	**System Response**
User clicks on ‘custom’ button	Game directs user to custom level mode page
User enters valid custom level seed	Game directs user to the level to play
User enters invalid custom level seed	Game displays a message stating failed seed validation

4.1.7.3 Functional Requirements

11 The game must allow users to play custom levels using a unique seed for that custom level

11.1 The game must allow users to input a seed at the custom level mode page to access the custom level

11.1.1 The custom level seed must be obtained from the user who made the custom level

11.1.2 The game must validate the custom level seed to ensure that the custom level exists

11.2 The game must load the specific custom level for the user to play upon successful validation of the custom level seed

4.1.8 Custom Level Mode – Creating Custom Level

4.1.8.1 Description and Priority

- This feature allows students and teachers to create custom levels
- Very high priority as these are two of the main modes of gameplay

4.1.8.2 Stimulus and Response Sequences

User Action	**System Response**
User clicks on ‘create custom level’ button on the custom level mode landing page	Game directs user to the ‘custom level setup’ interface
User enters the number of questions to be in the custom level	Game generates specified number of questions with default topic
User customizes each of the questions to their liking and clicks on ‘Create’ button	• Game saves custom level in the database, and generates a seed that acts as the key in the database for that custom level • Game displays seed to user for user to share with other users

4.1.8.3 Functional Requirements

12 The game must allow the user to make custom levels

12.1 The game must allow the user to set the number of questions in the custom level

12.2 The game must allow the user to customize each question in the custom level in terms of the mathematical operation and operand for the question

12.3 The game must provide a ‘Create’ button for the user to indicate that they have finished customizing the questions

12.4 Upon finishing customization of questions, the game must save the custom level in its database and generate a seed and show it to the user.

12.4.1 The seed must act as the key in the database for storing the custom level configuration and must be used by those playing the custom level to access the custom level.

4.1.9 Character Edit

4.1.9.1 Description and Priority

- This feature allows users to customize their characters in the game
- High priority as this feature allows users to express their individuality

4.1.9.2 Stimulus and Response Sequences

User Action	**System Response**
User updates character appearance by pressing arrow buttons	Game updates the appearance of the character in-game

4.1.9.3 Functional Requirements

13 The game must allow the user to change character appearance to suit the user's preferences

13.1 The game must update the user's in-game appearance according to the chosen appearance

4.1.10 Pause Menu

4.1.10.1 Description and Priority

- This feature allows users to edit the volume of the music in the game or resume playing
- Medium priority as the game is generally short in time, users can easily complete a level without pausing

4.1.10.2 Stimulus and Response Sequences

User Action	**System Response**
User clicks on pause button	Game pauses current level and shows the paused menu
User adjusts music slider	Game adjusts music level
User clicks on resume button	Game resumes current level

4.1.10.3 Functional Requirements

14 The game must have a pause functionality with a pause button within the level gameplay

14.1 The game must display the pause menu upon the user clicking on the pause button

14.2 The game must update itself with the new music level upon the user changing the music slider

14.3 The game must resume itself upon the user clicking the resume button

4.1.11 Leaderboard

4.1.11.1 Description and Priority

- This feature allows users to see the leaderboards of the game
- Very high priority as this is one of the main features of the game which allows for competitive gameplay and makes the game more exciting

4.1.11.2 Stimulus and Response Sequences

User Action	**System Response**
User clicks on leaderboard screen	Game displays the default leaderboard which is the global total score leaderboard
User filters the leaderboard according to different filters	Game displays the filtered leaderboard accordingly

4.1.11.3 Functional Requirements

15The game must allow the user to choose the type of Leaderboard that they want to view

15.1 The game must provide a dropdown list of worlds, levels, and custom levels for the user to choose to see specific leaderboards of.

15.2 The game must be flexible in terms of the worlds and levels chosen by the user for the leaderboard

15.2.1 The student may choose only a world with no levels

15.2.1.1 The game must display the leaderboard of the world with all the scores for all the levels in the world aggregated together

15.3 The default choice for the dropdown list must be ‘All’.

15.4 The game must update the leaderboard to show the leaderboard of the specific combination of world, and level the student has chosen using the dropdown lists

4.1.12 Teacher – Main Menu

4.1.12.1 Description and Priority

- This feature allows teachers to access the various options possible for them in the game
- Very high priority as this is one of the main menus of the game

4.1.12.2 Stimulus and Response Sequences

User Action	**System Response**
User selects ‘Runner Game’	Game directs user to ‘World Selection’ page
User selects ‘Leaderboard’	Game directs user to ‘Leaderboard’ page
User selects ‘Data Analytics’ mode	Game directs user to ‘Data Analytics’ mode/page
User selects ‘Logout’	Game logs user out
User selects ‘Quit Game’	Game exits

4.1.12.3 Functional Requirements

16 The main menu must allow teachers to choose which action they wish to perform

16.1 The game must direct the teacher to the selected page/screen/action

4.1.13 Teacher – Analytics

4.1.13.1 Description and Priority

- This feature allows teachers to check the progress of users for any world, level, or custom level.

4.1.13.2 Stimulus and Response Sequences

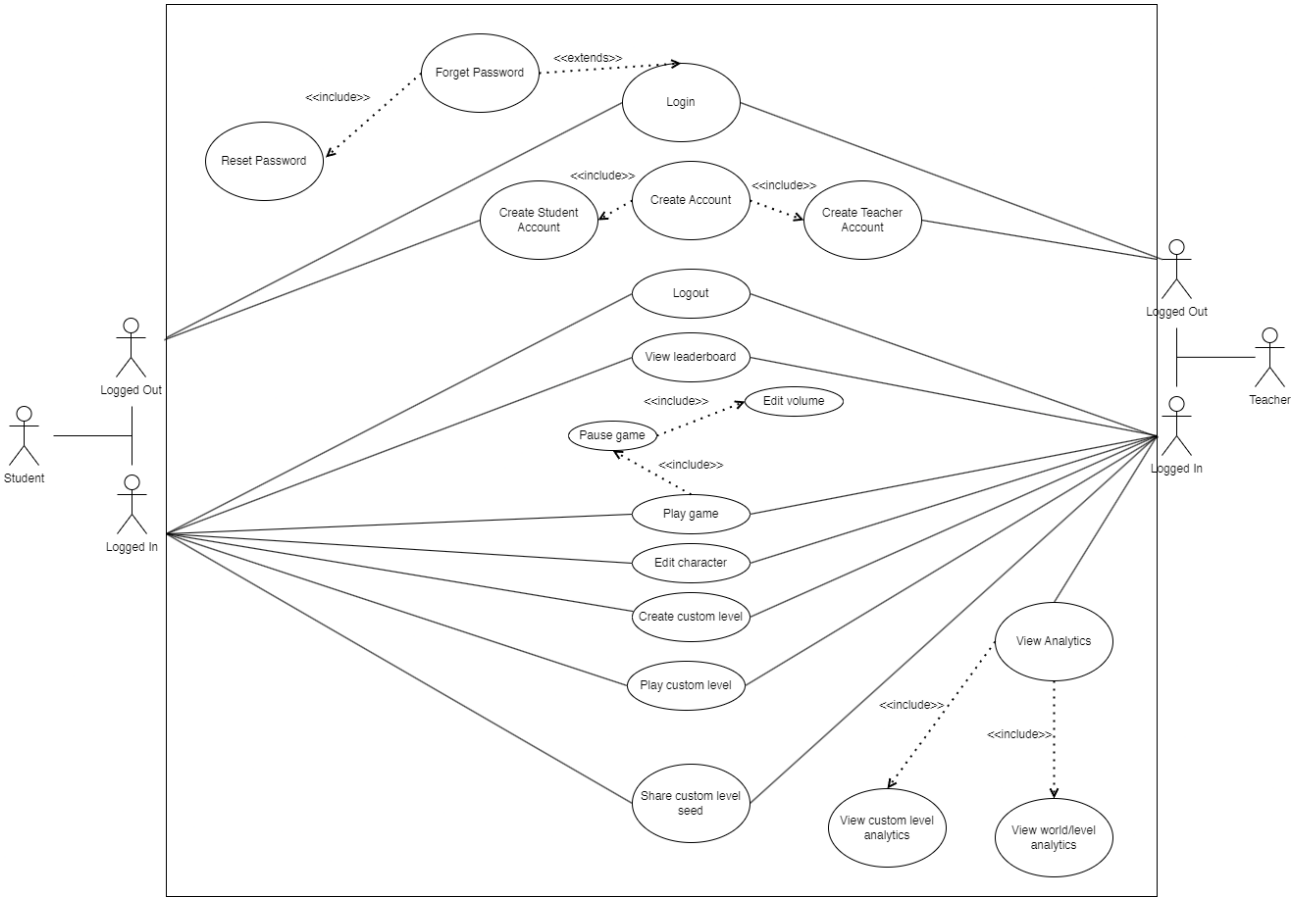
User Action	System Response
User selects ‘Assignments’ tab in the ‘Analytics’ page	• Display a list of custom levels, and their respective statistics.
User selects ‘Default Content’ tab in the ‘Analytics’ page	• Display a list of worlds and levels, and their respective statistics.

4.1.13.3 Functional Requirements

17 The game must allow teachers to check the data analytics of users for each world, level, and custom level.

17.1 The game must display the following statistics: number of students who have attempted the world/level/custom level, the average number of failures out of those who have attempted for the world/level/custom level, and the average number of questions answered correctly for the best attempt of a user for that particular world/level/custom level

4.2 Use-Case Diagram



4.3 Use-Case Descriptions

Use Case ID	1		
Use Case Name	Create Teacher Account		
Created By	Kyaw	**Last Updated By**	Kyaw
Date Created	4/9/2022	**Date Last Updated**	26/10/2022

Actor	Teacher – Not logged in (Initiating Actor)
Description	The teacher must be able to create a teacher account
Preconditions	1. The game must be running and be connected to the internet 2. The teacher must have a valid access code, and a valid email address
Postconditions	The game must create a teacher account and save its credentials in its database
Priority	Very high
Frequency of Use	Seldom
Flow of Events	1. Teacher clicks on the ‘Register’ button 2. The game directs the teacher to the account creation form page 3. Teacher selects ‘Teacher’ option as the account type 4. Teacher enters email in the email field 5. Teacher enters password in the password field 6. Teacher enters the access code in the access code field 7. Teacher clicks on ‘Register’ button 8. Game validates the input fields 9. Account is created by the game 10. Game displays a success message to the teacher 11. Game directs teacher to the login screen
Alternative Flows	AF-S8: Teacher enters invalid values in input field

	1. System displays appropriate error message depending on which input field had the validation error 2. System prompts teacher to reenter the input for the erroneous field
Exceptions	Nil
Includes	Nil
Special Requirements	Nil
Assumptions	The device the game is running on is connected to the internet
Notes and Issues	Nil

Use Case ID	2		
Use Case Name	Create Student Account		
Created By	Kyaw	**Last Updated By**	Kyaw
Date Created	4/9/2022	**Date Last Updated**	26/10/2022

Actor	Student – Not logged in (Initiating Actor)
Description	The student must be able to create a teacher account
Preconditions	• The game must be running and be connected to the internet • The student must have a valid email address
Postconditions	The game must create a teacher account and save its credentials in its database
Priority	Very high
Frequency of Use	Seldom
Flow of Events	1. Student clicks on the ‘Create Account’ button 2. The game directs the student to the account creation form page 3. Student selects ‘Student’ option in the account type 4. Student enters email in the email field 5. Student enters password in the password field 6. Student clicks on ‘Create Account’ button 7. Game validates the input fields 8. Account is created by the game 9. Game displays a success message to the student 10. Game directs student to the login screen
Alternative Flows	AF-S7: Student enters invalid values in input field 1. System displays appropriate error message depending on which input field had the validation error 2. System prompts student to reenter the input for the erroneous field

Exceptions	Nil
Includes	Nil
Special Requirements	Nil
Assumptions	The device the game is running on is connected to the internet
Notes and Issues	Nil

Use Case ID	3		
Use Case Name	Login		
Created By	Yin Ning	**Last Updated By**	Kyaw
Date Created	6/9/2022	**Date Last Updated**	26/10/2022

Actor	Either student or teacher – not logged in
Description	The user must be able to log into the game.
Preconditions	1. The game must be running and be connected to the internet 2. The user must have created a valid account.
Postconditions	Access must be granted, and the user shall access the content in the application.
Priority	Very high
Frequency of Use	Very high
Flow of Events	1. The game prompts the user to log in using email address and password 2. Account login details are authenticated. 3. Upon successful authenticated, the user is granted access to the application.
Alternative Flows	AF-S2: The user types in a wrong account login credentials. 1. The application must display an error message “Invalid Email/Password. Please try again.” 2. The application returns to Step 1.
Exceptions	Nil
Includes	Nil
Special Requirements	Nil
Assumptions	The device the game is running on is connected to the internet.
Notes and Issues	Nil

Use Case ID	4		
Use Case Name	View leaderboard		
Created By	Kyaw	**Last Updated By**	Kyaw
Date Created	7/9/2022	**Date Last Updated**	26/10/2022

Actor	Either student or teacher
Description	The user must be able to view and customize the leaderboard
Preconditions	1. The game must be running and be connected to the internet 2. The user must be logged in
Postconditions	The user is able to view their desired leaderboard
Priority	High
Frequency of Use	Medium
Flow of Events	1. System displays the leaderboard with the world, and level filters set to ‘All’ 2. User adjusts the filters according to their needs 3. System updates the leaderboard after querying the database based on the filters
Alternative Flows	Nil
Exceptions	Nil
Includes	Nil
Special Requirements	Nil
Assumptions	The device the game is running on is connected to the internet.
Notes and Issues	Nil

Use Case ID	5		
Use Case Name	Logout		
Created By	Kyaw	**Last Updated By**	Kyaw
Date Created	7/9/2022	**Date Last Updated**	26/10/2022

Actor	Either teacher or student – logged in
Description	The user must be able to logout
Preconditions	1. The game must be running and be connected to the internet 2. The user must be logged in
Postconditions	Game logs user out and allows another user to log in
Priority	Medium
Frequency of Use	Medium
Flow of Events	1. User clicks on logout button 2. Game logs user out and returns to the login screen
Alternative Flows	Nil
Exceptions	Nil
Includes	Nil
Special Requirements	Nil
Assumptions	The device the game is running on is connected to the internet.
Notes and Issues	Nil

Use Case ID	6		
Use Case Name	Play game level		
Created By	Kyaw	**Last Updated By**	Kyaw
Date Created	7/9/2022	**Date Last Updated**	26/10/2022

Actor	Either teacher or student – logged in
Description	The user must be able to play a game level
Preconditions	1. The game must be running and be connected to the internet 2. The user must be logged in
Postconditions	System updates necessary tables and fields in the database based on what world(s) and levels(s) the student has played
Priority	Very high
Frequency of Use	Very high
Flow of Events	1. User clicks 'Runner Game' button 2. Game directs user to world selection screen 3. User selects world to play 4. Game directs user to the level selection screen 5. User selects a level to play from amongst unlocked levels 6. Game directs user to the gameplay screen 7. Game loads question for user to answer and randomly generating possible answers to the question as options 8. User moves character to the correct answer option and collides with it 9. Game shows correct answer notification 10. Game adjusts score and combo value 11. Game adjusts speed/difficulty of the level and fetch next question 12. Steps 7 to 11 repeat until user answers a certain number of questions

	13. Game shows complete level notification 14. Game brings user back to level selection and updates with new levels unlocked (if any) 15. Game brings user back to level selection
Alternative Flows	AF1-S8: User moves onto the incorrect lane running lane and collides with the wrong answer option 1. Game shows incorrect answer notification 2. Game adjusts combo value to 0, resets speed to original speed 3. Continue with Step 7
Exceptions	Nil
Includes	Nil
Special Requirements	Nil
Assumptions	The device the game is running on is connected to the internet.
Notes and Issues	Nil

Use Case ID	7		
Use Case Name	Create custom level		
Created By	Kyaw	**Last Updated By**	Kyaw
Date Created	7/9/2022	**Date Last Updated**	26/10/2022

Actor	Student/Teacher (initiating actor)
Description	The user must be able to create a custom level
Preconditions	1. The game must be running and be connected to the internet 2. The user must be logged in
Postconditions	System saves the custom level in the database along with the seed
Priority	Very high
Frequency of Use	High
Flow of Events	1. User clicks on ‘create a custom level’ button in the custom level mode landing page 2. Game directs user to the ‘custom level setup’ interface 3. User enters number of questions to be in the custom level 4. Game generates specified number of questions in a vertical scrollable carousel for user to see and customize 5. User customizes questions according to user’s preferences in terms of operand and operators 6. User clicks ‘Create’ button 7. Game saves custom level in the database, and generates a unique seed that acts as the key in the database for that custom level 8. Game displays the seed to user 9. User is brought back to custom level landing page

Alternative Flows	Nil
Exceptions	Nil
Includes	Nil
Special Requirements	Nil
Assumptions	The device the game is running on is connected to the internet.
Notes and Issues	Nil

Use Case ID	8		
Use Case Name	Play custom level		
Created By	Kyaw	**Last Updated By**	Kyaw
Date Created	7/9/2022	**Date Last Updated**	26/10/2022

Actor	Either teacher or student – logged in
Description	The user must be able to play custom levels
Preconditions	1. The game must be running and be connected to the internet 2. The user must be logged in
Postconditions	System updates database to reflect the new progress of the custom level for the user
Priority	High
Frequency of Use	High
Flow of Events	1. User enters the custom level mode landing page of the game 2. User enters the seed in the 'Enter Level Seed…' field 3. The system validates the seed 4. If valid, the system loads the level and the user's game goes into the custom level 6. Upon completion of the custom level, the user is shown a successfully completed message and database updates the scores
Alternative Flows	AF1-S3: The student enters an invalid seed 1. System displays error message and prompts user to reenter a seed
Exceptions	Nil
Includes	Nil
Special Requirements	Nil

Assumptions	The device the game is running on is connected to the internet.
Notes and Issues	Nil

Use Case ID	9		
Use Case Name	Edit character		
Created By	Kyaw	**Last Updated By**	Kyaw
Date Created	7/9/2022	**Date Last Updated**	26/10/2022

Actor	Either teacher or student – logged in
Description	The user must be able to edit their character
Preconditions	1. The game must be running and be connected to the internet 2. The user must be logged in
Postconditions	Game updates the appearance of the character in-game
Priority	Low
Frequency of Use	Low
Flow of Events	1. At the main menu screen, user navigates using the left and right arrow buttons on the left to choose the preferred character appearance 2. Game updates character appearance in-level
Alternative Flows	Nil
Exceptions	Nil
Includes	Nil
Special Requirements	Nil
Assumptions	The device the game is running on is connected to the internet.
Notes and Issues	Nil

Use Case ID	10		
Use Case Name	Share custom level seed		
Created By	Kyaw	**Last Updated By**	Kyaw
Date Created	7/9/2022	**Date Last Updated**	26/10/2022

Actor	Either teacher or student – logged in
Description	The user must be able to share custom level seed
Preconditions	1. The game must be running and be connected to the internet 2. The user must be logged in
Postconditions	The custom level seed is shared with the intended audience
Priority	High
Frequency of Use	Medium
Flow of Events	1. At the custom level mode landing page, user clicks on ‘share’ button 2. Popup appears with level seed field and optional message field 3. User enters a valid seed and an optional message, and clicks ‘Share’ 4. System successfully validates seed and shares the seed and message (if any) with the Telegram group that the bot is in
Alternative Flows	AF1-S3: The user enters an invalid seed 1. System displays error message and prompts user to reenter a valid seed
Exceptions	Nil
Includes	Nil
Special Requirements	Nil

Assumptions	The device the game is running on is connected to the internet.
Notes and Issues	Nil

Use Case ID	11		
Use Case Name	View custom level analytics		
Created By	Kyaw	**Last Updated By**	Kyaw
Date Created	7/9/2022	**Date Last Updated**	26/10/2022

Actor	Teacher (Initiating actor)
Description	The teacher must be able to view the analytics report of custom levels in the game
Preconditions	1. The game must be running and be connected to the internet 2. The teacher must be logged in
Postconditions	The game shows the teacher the analytics report of custom levels in the game
Priority	Medium
Frequency of Use	Medium
Flow of Events	1. User selects ‘Analytics’ at main menu 2. System directs user to analytics page 3. User selects ‘Assignments’ tab 4. System loads the report for all the custom levels 5. System displays detailed completion information for the custom levels, such as the average failures and average correct answers for best attempts
Alternative Flows	Nil
Exceptions	Nil
Includes	Nil
Special Requirements	Nil
Assumptions	The device the game is running on is connected to the internet.
Notes and Issues	Nil

Use Case ID	12		
Use Case Name	View world/level analytics		
Created By	Kyaw	**Last Updated By**	Kyaw
Date Created	7/9/2022	**Date Last Updated**	26/10/2022

Actor	Teacher (Initiating actor)
Description	The teacher must be able to view the analytics report of custom levels in the game
Preconditions	1. The game must be running and be connected to the internet 2. The teacher must be logged in
Postconditions	The game shows the teacher the analytics report of world/levels in the game
Priority	Medium
Frequency of Use	Medium
Flow of Events	1. User selects ‘Analytics’ at main menu 2. System directs user to analytics page 3. User selects ‘Default Content’ tab 4. System loads the report for all the worlds and levels 5. System displays detailed completion information for the worlds and levels, such as the average failures and average correct answers for best attempts
Alternative Flows	Nil
Exceptions	Nil
Includes	Nil
Special Requirements	Nil
Assumptions	The device the game is running on is connected to the internet.
Notes and Issues	Nil

Use Case ID	13		
Use Case Name	Forget password		
Created By	Kyaw	**Last Updated By**	Kyaw
Date Created	7/9/2022	**Date Last Updated**	26/10/2022

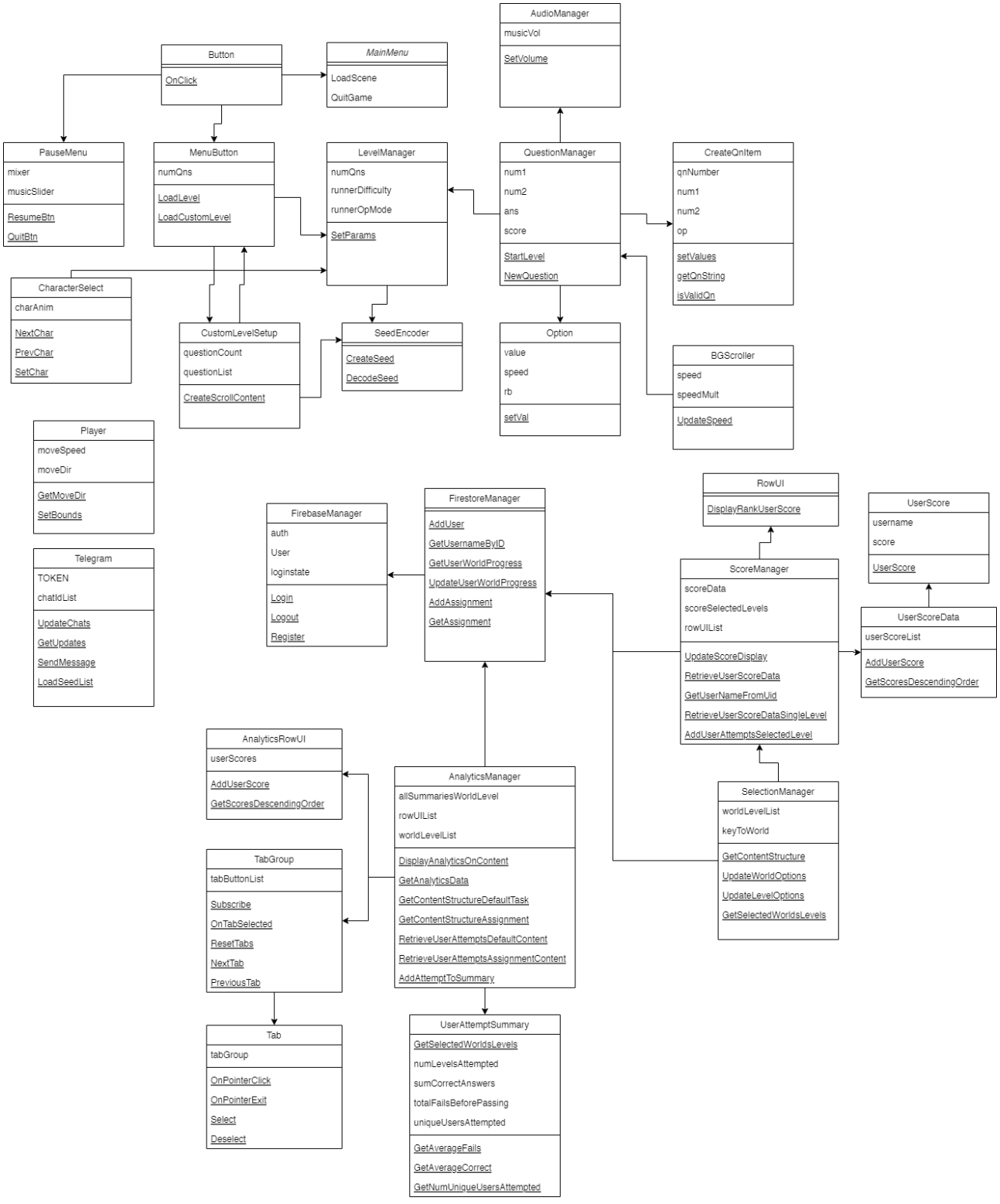
Actor	Either teacher or student – not logged in
Description	The user must be able to reset their password if they forget it
Preconditions	1. The game must be running and be connected to the internet 2. The user must not be logged in
Postconditions	The user successfully resets their password to a new one of their own choosing
Priority	High
Frequency of Use	Rare
Flow of Events	1. User clicks on ‘forget password’ button at login menu 2. System directs user to the “forget password form” page 3. System prompts user to enter user’s email address 4. User enters valid input (valid email address which can be found in the existing user database) in the “forget password form” page 5. System validates email successfully 6. System sends an email to the user’s email containing a link to reset password 7. User opens link in the email 8. User resets password following prompts at the link destination 9. System validates new password to meet requirements 10. System updates user’s password to the new password

	11. System displays successful password reset message to user
Alternative Flows	AF-S4: User enters invalid email address input in the “forget password form” page 1. System displays an error message 2. System prompts user to reenter the input for the erroneous field AF-S9: User enters invalid new password in the “forget password form” page 1. System displays an error message 2. System prompts user to reenter the input for the erroneous field
Exceptions	Nil
Includes	Nil
Special Requirements	Nil
Assumptions	The device the game is running on is connected to the internet.
Notes and Issues	Nil

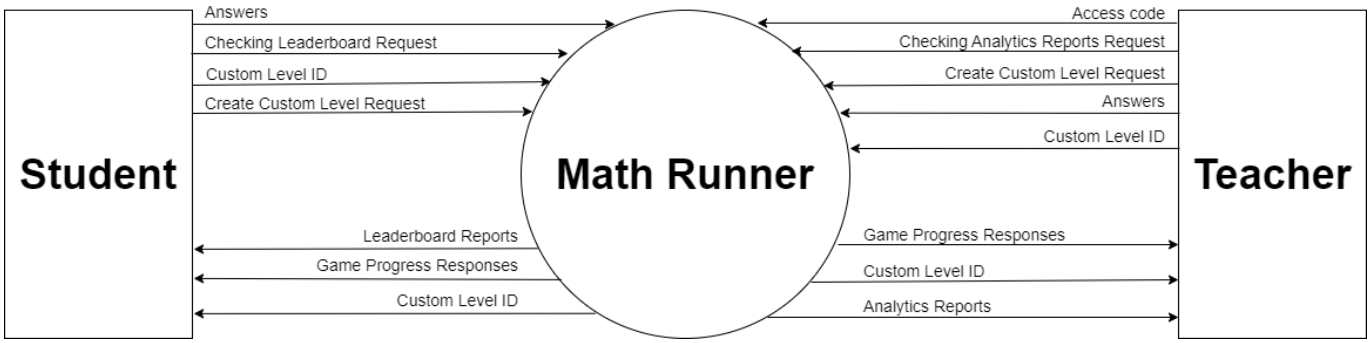
Use Case ID	14		
Use Case Name	Pause Game		
Created By	Kyaw	**Last Updated By**	Xinrui
Date Created	7/9/2022	**Date Last Updated**	27/10/2022

Actor	Either teacher or student – logged in
Description	The user must be able to pause their game
Preconditions	1. The game must be running and be connected to the internet 2. The user must be logged in
Postconditions	The level is temporarily paused
Priority	High
Frequency of Use	Rare
Flow of Events	1. While playing a level, user clicks on 'Pause button' 2. Game pauses the level temporarily and displays the paused menu overlay 3. After a period of time, user clicks 'Resume' button 4. System resumes the level
Alternative Flows	AF-S3: User adjusts the music slider 1. System adjusts the music volume according to the level set by the user
Exceptions	Nil
Includes	Nil
Special Requirements	Nil
Assumptions	The device the game is running on is connected to the internet.
Notes and Issues	Nil

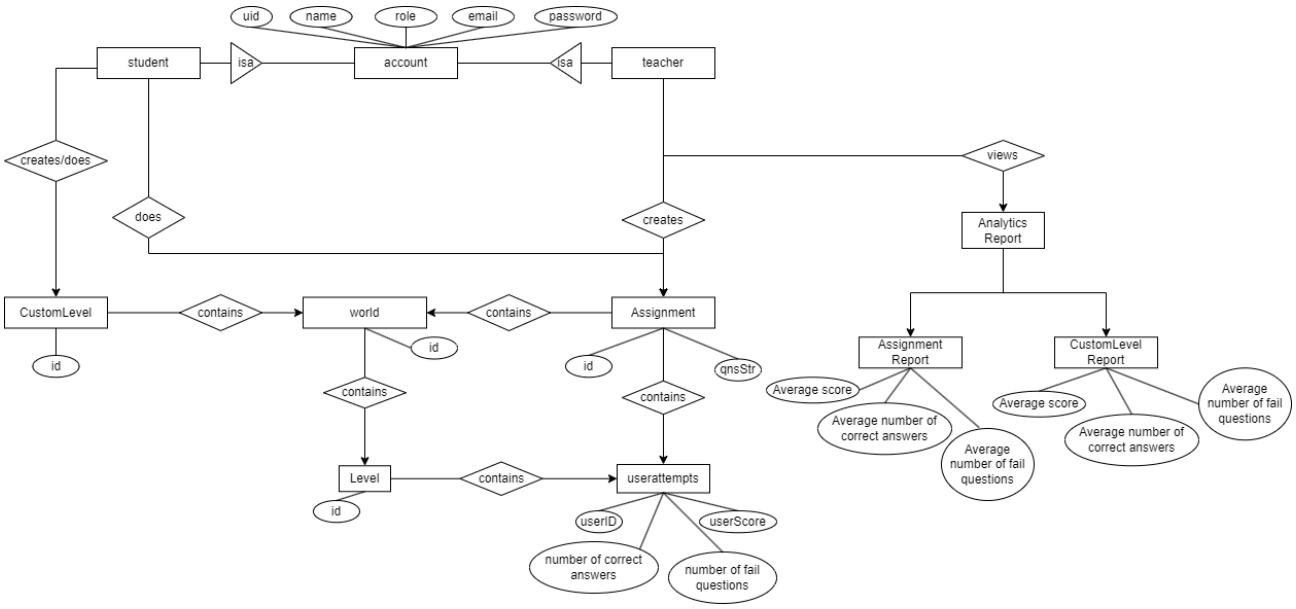
4.4 Class Diagrams



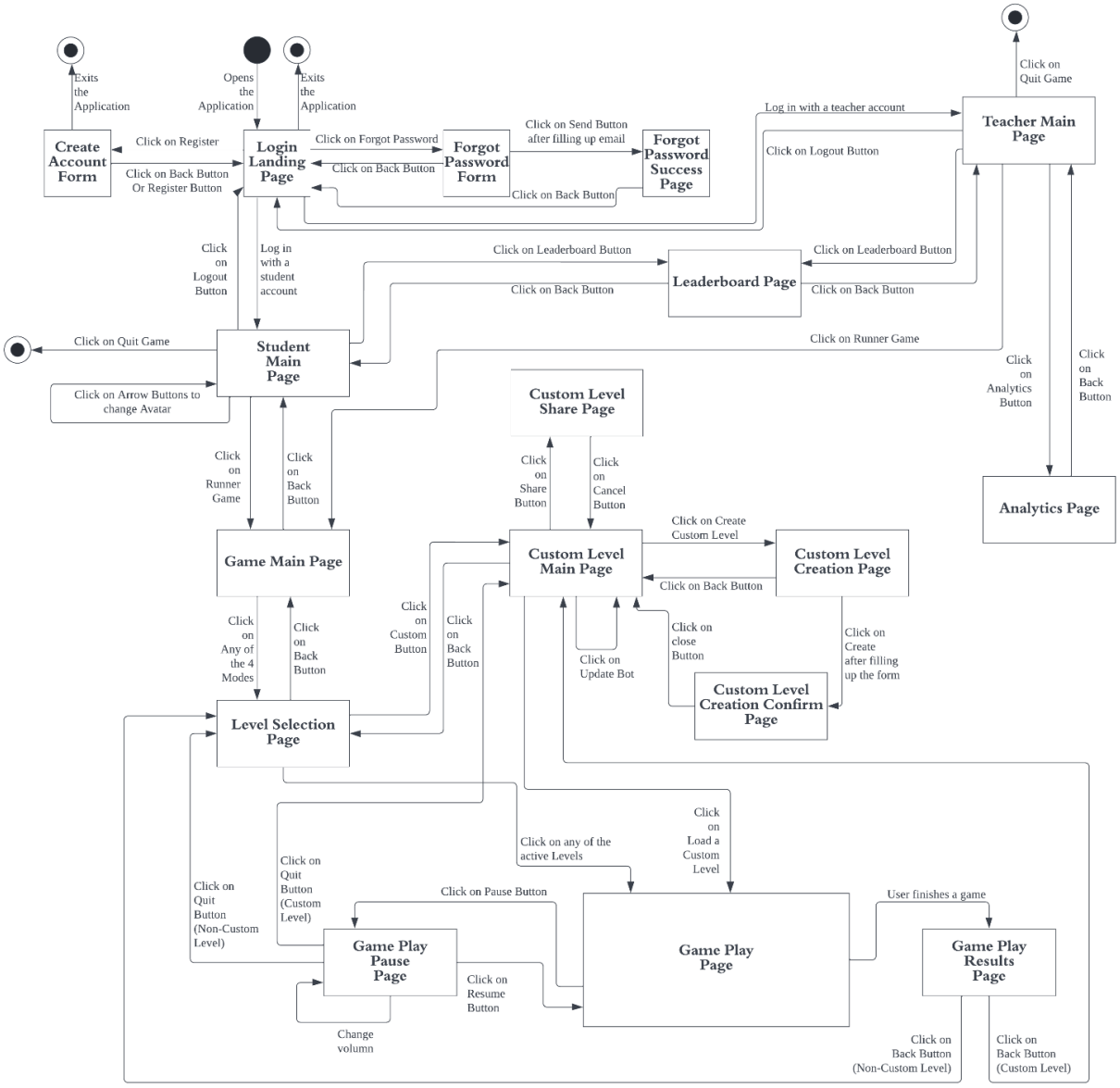
4.5 Context Diagram



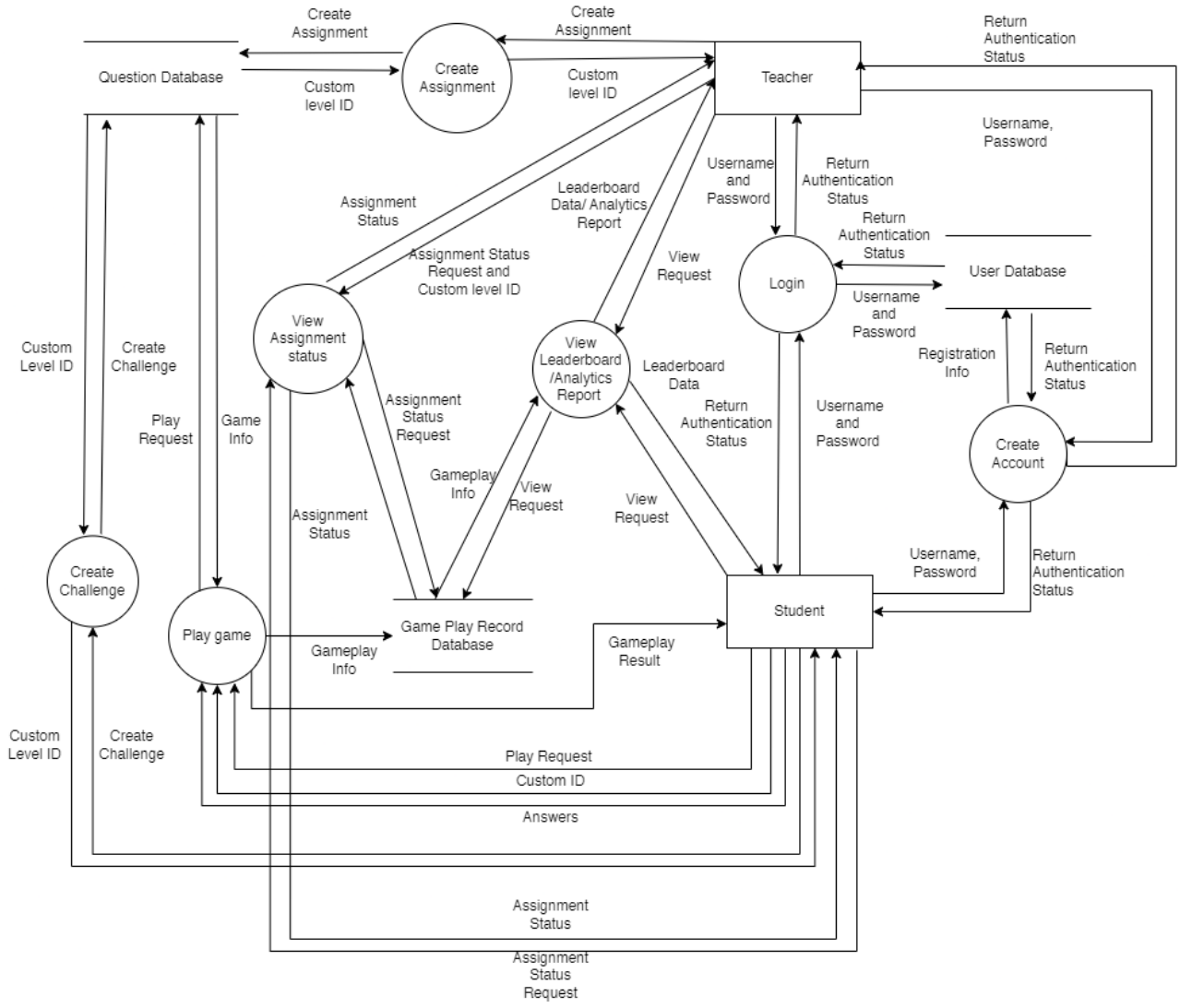
4.6 Entity Relationship Diagram



4.7 Dialogue Map



4.8 Data Flow Diagram



4.9 CRUDL Matrix

CRUDL: Create, Read, Update, Delete, or List

Use Case/Entity	Account	Custom Level	User Attempts	Worlds	Levels	Analytics
Create Teacher Account	C,R					
Create Student Account	C,R					
Teacher Login	R					
Student Login	R					
View leaderboard	R,L	R,L	R,L			
Create custom level	R	C				
Share custom level	R	R				
Play custom level	R	R	C,R,U			
Forget/reset password	R,U					
Play single player mode	R		C,R,U	R	R	
View world/level analytics	R		R,L			C,R,U,L
View custom level analytics	R		R,L			C,R,U,L

4.10 Decision Table(s)

1. Forget/Reset Password

Condition	1	2
User entered an existing email address	T	T
User entered a valid password to reset	-	T
Action		
Send reset password link to email	x	
Reset password		x

2. Login

Condition	1	2	3	4
User entered an existing email address	T	T	F	F
User entered a valid password	T	F	T	F
Action				
Allow user to log in	x			
Show error message		x	x	x

3. Unlocking Level

Condition	1	2	3
User has finished previous level	T	F	T
User has achieved passing score	F	F	T
Action			
Unlock next level			x

4. Creating Custom Levels

Condition	1	2	3
Number of questions is at least 1	F	T	T
All questions have both operands filled	-	F	T
Action			
Custom level can be created			x

5. Leaderboard

Condition	1	2	3
User has world filter	F	T	T
User has level filter	F	F	T
Action			
Shows summation of all scores	x		
Shows scores from selected world only		x	
Shows scores from selected world and level only			x

4.11 Communication Diagrams

Three communication diagrams are showcased below for the following scenarios:

1. Teacher registering a new account
2. User logging into account
3. User creating a custom level

These scenarios are chosen as they all involve several layers of communication between different components of the game, as well as the database.

Diagram 1: Teacher Register Account

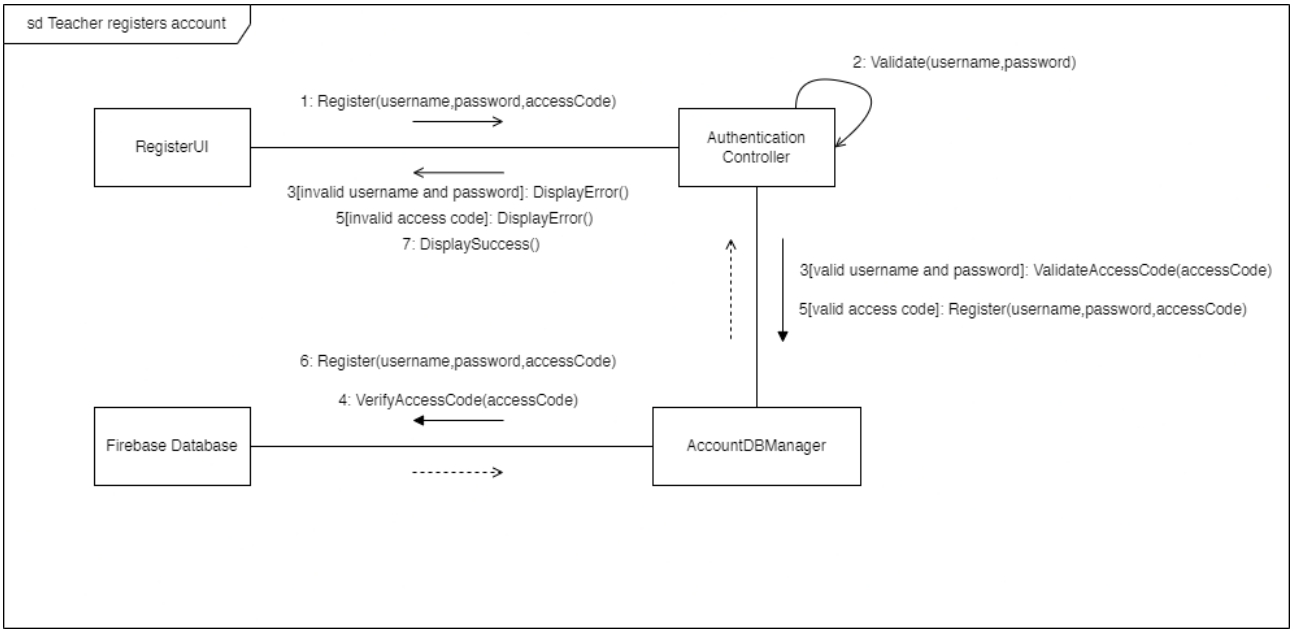


Diagram 2: User Logins to Account

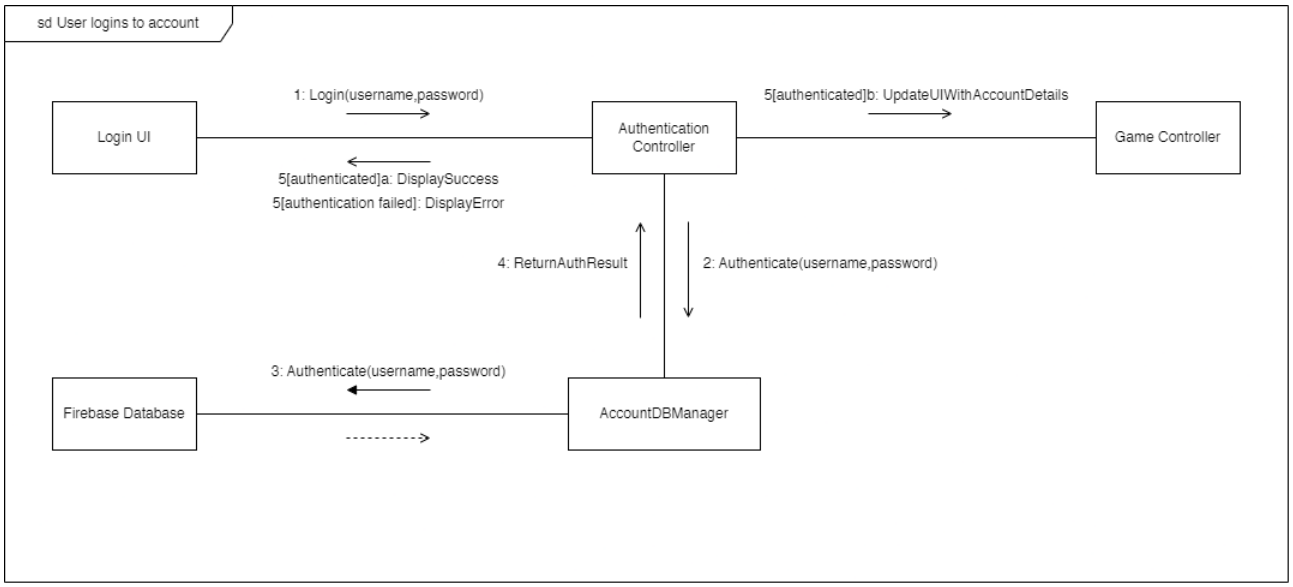
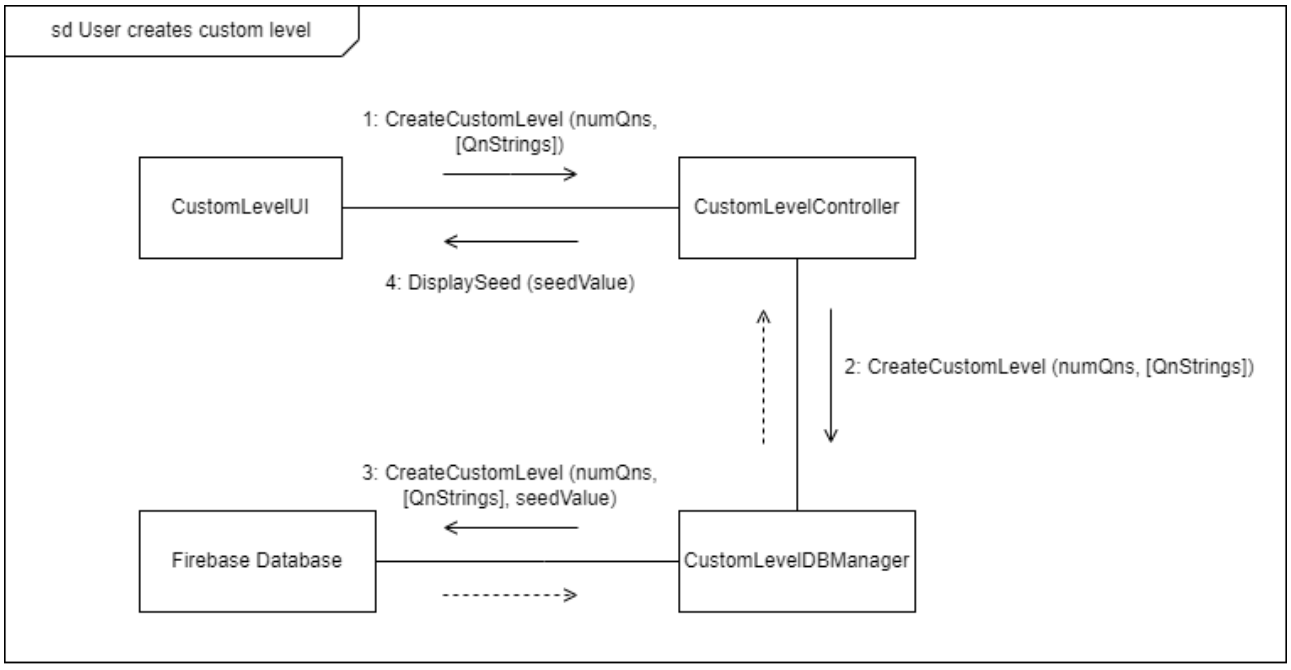
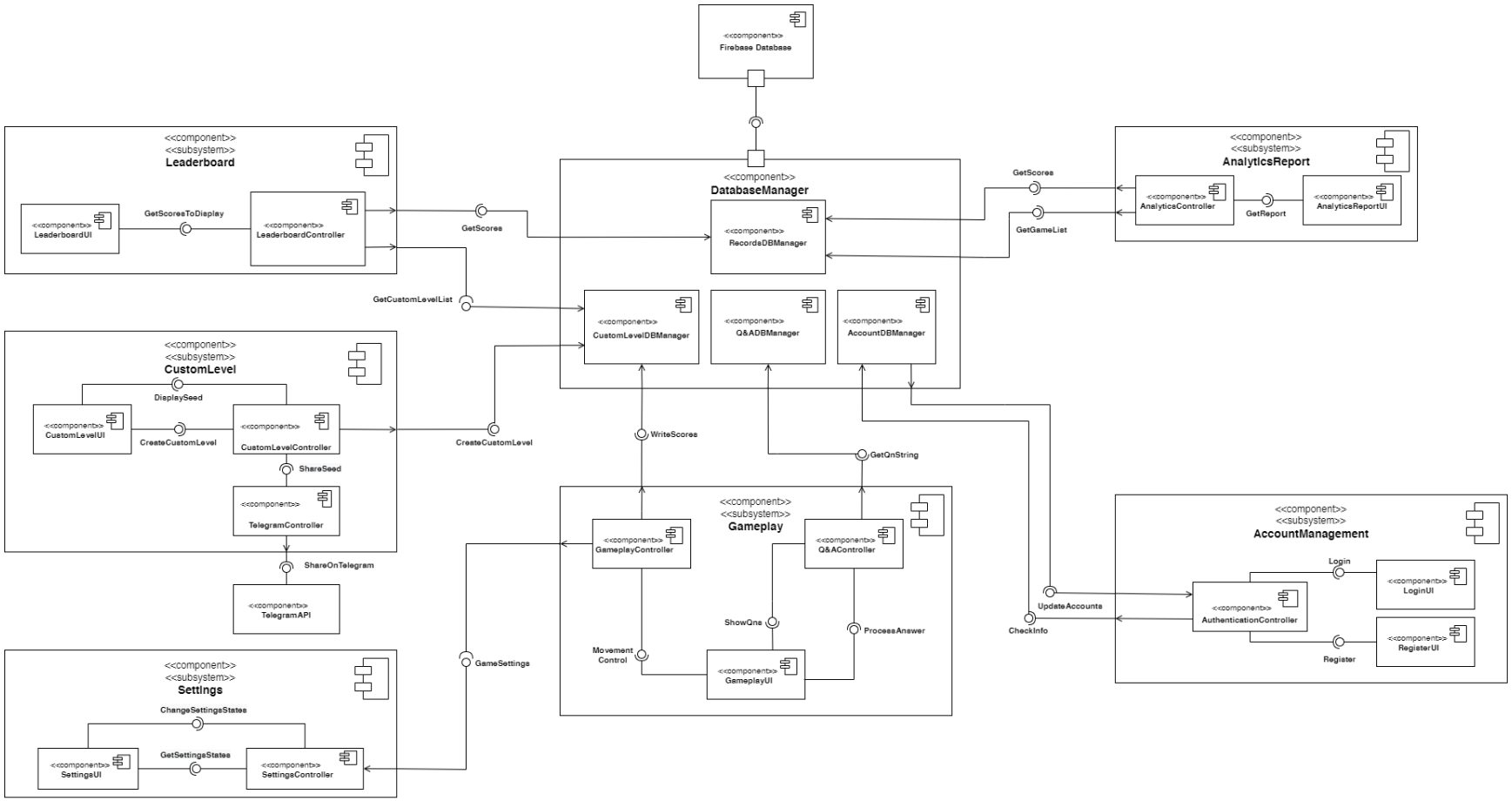


Diagram 3: User Creates a Custom Level



4.12 Component Diagram



5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 Startup Related

5.1.1.1 Game must take less than 3 seconds to be fully loaded and at the login screen

5.1.1.2 Game must take less than 3 seconds to be fully loaded upon restarting the game from an unexpected crash

5.1.1.3 Login authentication must be done in less than 3 seconds

5.1.2 In-Game Related

5.1.2.1 In between worlds and levels, the game must take less than 3 seconds to load the next one

5.1.2.2 User input to the game must provide a response in less than 3 seconds.

5.1.2.3 Data analysis report must be generated in less than 5 seconds upon request by a teacher

5.1.2.4 The game must support multiple concurrent users using the application

Rationale for quick load times: Reduce loading times on user's side to improve user satisfaction when using the application. Users typically consider waiting times less than 3 seconds to be 'fast', waiting times less than 1 second to be 'instantaneous', and any waiting time beyond 3 seconds to be 'slow' or 'unresponsive'. Therefore, the system must be considered 'fast' by users, as instantaneous is not possible due to the need for data processing and calling of external APIs. Even in the case of an unforeseen crash, being able to restart quickly would improve the user's impression of the application.

5.1.3 Database Related

5.1.3.1 Game database must be able to store the account-related data such as the historical progress of students of at least 1000 users.

5.1.3.2 Game database must store at least 1000 questions, options, and answer triplets.

5.1.3.3 Game database must support at least 1000 concurrent queries and provide the correct response within 30 seconds.

5.2 Flexibility Requirements

5.2.1 The system architecture must allow for flexibility in terms of new features such as additional worlds and levels.

5.2.1.1 This must be done without there being extensive modifications to existing code and architecture. In other words, the system architecture should have a low degree of coupling and a high degree of cohesion.

5.3 Data Persistence Requirements

5.3.1 User data such as progress in the game, account credentials must be stored in a secured database

5.3.2 In the event of any application crashes, the game must be able to restore the latest saved data upon restarting.

5.4 Security Requirements

5.4.1 Users must have an account in order to access the game

5.4.2 A single user must have access only to their account

5.4.3 Student accounts must only have access to their own data only, except for leaderboard scores for which they can view others' scores

5.4.4 Teachers must have access to their own data and students' data

5.4.5 Passwords must fulfill a minimum complexity requirement. Specifically, passwords must be at least 8 characters long and consist of at least one uppercase alphabet, lowercase alphabet, number, and symbol.

5.4.6 Passwords must be stored securely in an encrypted manner in the authentication service's database.

5.5 Software Quality Attributes

5.5.1 Usability

5.5.1.1 The game must be in English.

5.5.1.2 The user interface of the game must be simple and intuitive, with proper design language guidelines followed.

5.5.2 Reliability

5.5.2.1 The game's services must be up and accessible by all users for at least 99% of the time, i.e., 99% uptime.

5.5.3 Portability

5.5.3.1 The game's database(s) must be stored in the cloud so that users can access the game and their account using other devices without needing to be on the same device all the time

5.5.4 Maintainability

5.5.4.1 The game's system architecture must be designed such that subsystems that are present are decoupled as much as possible so that changes in one subsystem's internals does not adversely affect other subsystems.

5.5.4.1.1 The codebase must make use of appropriate design patterns to ensure good system design.

5.5.4.2 The codebase must have extensive comments and documentation to ensure that new developers who take up the task of maintaining the codebase will not have difficulty in understanding what is happening.

5.5.5 Testability

5.5.5.1 The game's individual components must be testable, and issues found must be easily narrowed down to individual subsystems or components

5.6 Business Rules

5.6.1 Students must be from a school that has obtained the game for use

5.6.2 Teachers must be from a school and have an access code to create an account

6. Candidate Architecture

Architectural design for a software system is to analyze and identify the various subsystems that make up the system and how control and communication of and between subsystems would function. The output of the process is a high-level overview and description of the system architecture and will describe how the major components of the software are related to one another and which components interact with one another. Having such a high-level overview aids in communication amongst various stakeholders, both technical and non-technical, as the architecture diagram is not cluttered up in minute details but rather an abstracted overview.

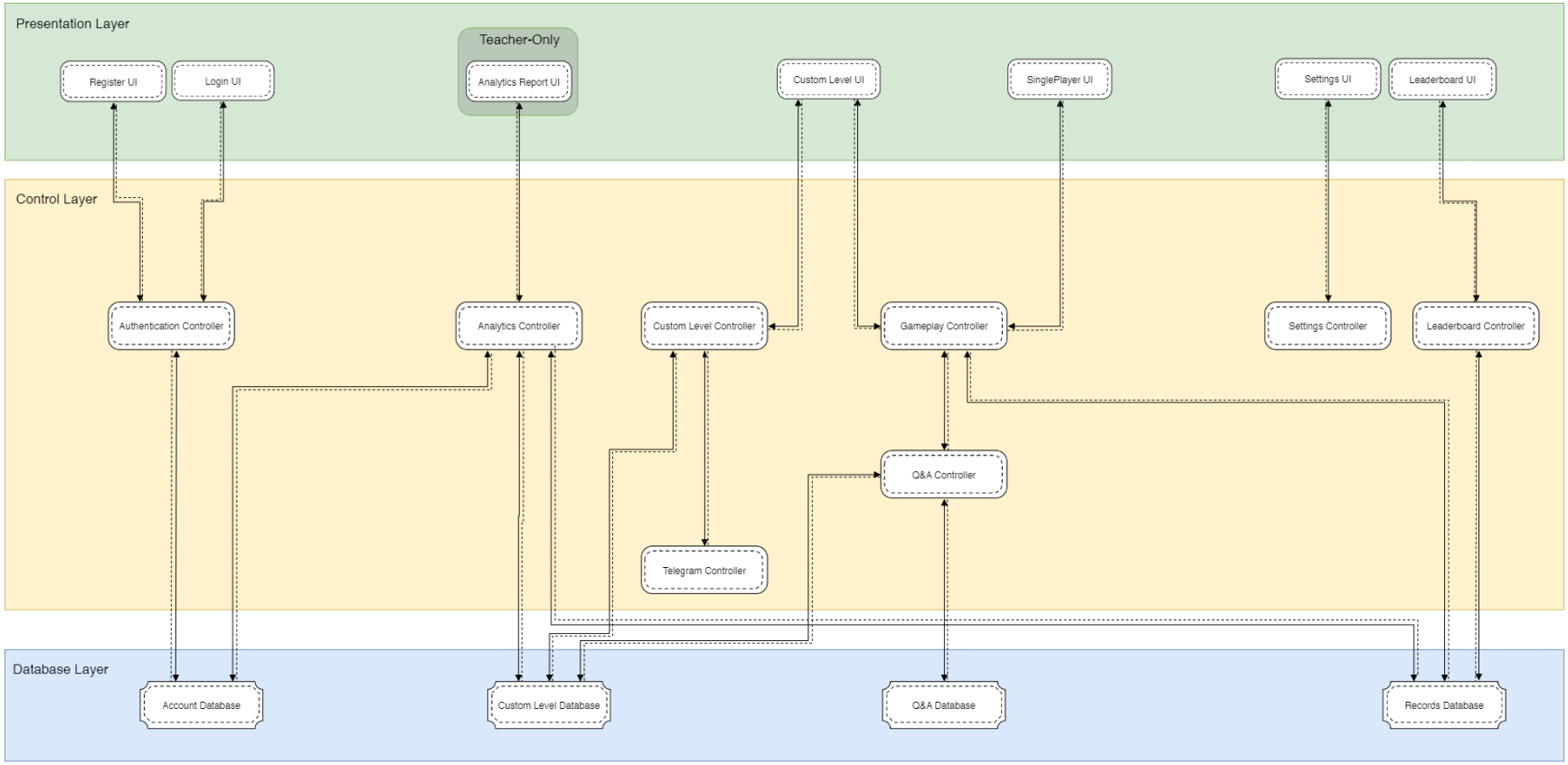
In choosing a particular architecture of the system, we kept in mind the relevant non-functional requirements and software quality attributes of the system that have been laid out in the SRS such as:

- Performance
- Flexibility & Scalability
- Security
- Maintainability
- Testability

We also compared and contrasted the various architectural styles possible, their respective pros and cons, and came to a decision after careful deliberation.

6.1 Candidate Architecture – Layered Subtype of Call & Return Architectural Style

Unity-specific components such as rendering engines and game engines are excluded from the diagram.



6.1.1 Rationale For Chosen Architecture

We chose the layered subtype of the call-and-return architectural style, which we will refer to ask the layered architecture. The layered architecture segregates the software system into separate layers where each layer looks after its set of components, functions, processes, and controls.

What is Layered Architecture?

The layered architectural style, like its name suggests, comprises of several layers of components. Each layer would be responsible for a high-level role/responsibility such as user interaction, or database interaction. Within each layer, there would be individual components that perform more specific functions. For example, within the database layer, there might be components that interact with a specific database, such as an accounts management component interacting with an accounts database.

The layers themselves are considered ‘closed’, which refers to the fact that an API request or function call from a component within a layer would only be directed towards another component in the same layer, or a component in the layer that is immediately below it, i.e. it cannot skip layers to reach non-adjacent layers.

Due to there being multiple layers and multiple components per layer, this architectural style allows for separation of concerns across different components and different layers. No component would be responsible for multiple functionalities or have a scope that extends across multiple layers. This also ensures that any changes that might be made in one layer will not affect other layers unless the API functions or subroutines change in their inputs and outputs.

Layers

Our proposed layered architecture will have three layers: presentation layer, control layer, and database layer.

The presentation layer has all the user-facing functions such as the user interfaces for logging in or registering an account, creating custom levels, playing the game, and viewing analytics reports.

The control layer has all the components that the user-facing components in the presentation layer rely upon to operate. The components here help bridge the gap between what is shown to/input by the user, and what is obtained from/sent to the database and are responsible for the critical parts of the application logical flow.

The database layer is immediately below the control layer and contains components that are each responsible for interactions with a particular table within the database. Interaction refers to various database operations such as read, write, update, delete and list. It comprises four components:

- Accounts Database – Contains methods to interact with the table containing all user information related to their profile and credentials
- Records Database – Contains methods to interact with the table containing the progress and scores of students for the various worlds/levels of the game
- Custom Level Database – Contains methods to interact with the table containing data related to custom levels
- Q&A Database – Contains methods to interact with the table containing the questions and answers for the various worlds/levels of the game

6.1.2 Benefits/Advantages

The benefits/advantages of using the layered architecture as follows:

1. **Less complexity**: Due to the modular design nature of components and layers, the complexity of individual components are much lesser than other architectural styles
2. **Extensibility**: As components are modular, new components can be easily added to enhance existing functionalities, as long as API requirements are met.

3. **Reuse of lower layer components**: Multiple upper layer components can easily use multiple lower layer components due to the loose coupling between components
4. **High testability**: With 'separation of concerns' being an inherent characteristic of the architecture where each layer only needs to concern itself with the data and controls under its scope, the resulting modular/individual components are much easier to test individually, which greatly eases and simplifies the testing process. Furthermore, abstracting the database APIs allows for improved testability, as we are able to verify the correctness of database calls easily.
5. **Maintainability**: The layered architecture makes for efficient maintenance of the program as maintenance and improvement can be done for separate components without affecting other components or layers, as long as the APIs remain the same.
6. **Low Coupling**: With the above features, layered architecture encourages low coupling as components are only linked to components from the same or adjacent layers as well as high cohesion as each component and layer has a specific role to perform.
7. **Flexibility**: It also provides great flexibility in terms of adding, removing and updating classes while also providing clarity in terms of dividing the code into smaller logical sections.

6.1.3 Trade-offs/Disadvantages

1. **Difficult to find the right levels of abstraction**: There might be over or under abstraction of components, whereby some subroutines might be oversimplified or be overcomplicated, in which case undesired effects such as low performance (explained below)
2. **Lower performance**: Performance may suffer due to there being more subroutine/function calls between multiple components due to over-abstraction. For example, if a database update operation function is abstracted into its own subroutine call and it is only able to accept a single update, then that might be worse in terms of performance than having a less abstracted component that is able to handle hundreds of operations at once but is more complicated. On the other hand, performance might also suffer if a component is not abstracted enough. For example, for database operation

function is built to process hundreds of operations at once, it might have additional safeguards in place to ensure correctness, which might not be necessary to be in place if the need was only for a single update operation, incurring additional overhead. A simple database update operation done by the end user will also have to go through all of the layers, which adds inefficiencies.

3. **Low scalability**: Due to tightly coupled and monolithic implementations of this pattern, the overall granularity is too broad, making it expensive to scale.

6.1.4 Comparison with other Architectural Styles

Independent-component architectures

This architectural style has components that communicate by passing messages to one another, but this is not enough for the requirements of our software system. We would require some shared data and variables that can be used by all functions and components. Additionally, the components are all independent, which means any logic to be executed to be taken for any message received will have to be inherently programmed into every component, which makes message-passing protocols required by components in this architectural style difficult to check and implement. It would require a lot of integrity protocols to determine whether or not messages have been successfully sent and received between components if the program was organized in this manner.

For the *client & server substyle*, there is only one ‘server’ serving multiple ‘clients’ which does not fit our requirements as there would be multiple databases, each of which act as a single ‘server’. This would incur a negative cost on flexibility of the entire system by having to work with just a single server.

For the *event-based implicit invocation substyle*, the publish-subscribe paradigm is key to it. The paradigm does not fulfil our requirements as there are components that do not work with such a publish-subscribe paradigm, like the login and register components. For the login and register components, if we have to fit into a publish-subscribe paradigm, it will mean that the authentication API has to subscribe to the account creation component and wait for

the publication of new accounts to act, and vice versa, which overcomplicates and shoehorns the components into an unnatural structure.

Data flow architectures

The inherent and most obvious weakness of data flow architectures is that there is only flow of data and no control flows within this architecture. This goes against our requirements completely, as we will require certain components in our system to have control over other components in certain situations and contexts.

The *pipe-and-filter substyle* is purpose-built for concurrency, which is not critical to our system and is not a key non-functional requirement as the system does not process a lot of data. Having such a data pipe would result in less interactivity as this is focused more on performing operations on the data along the pipeline, rather than having subroutine calls based on user interactions.

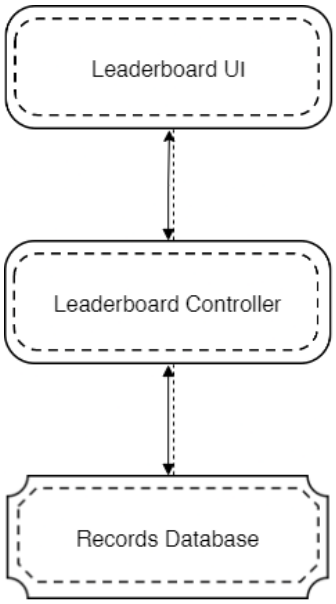
For the *batch sequential processing substyle*, a single process has to run from start to finish before another process can start, i.e., there is no concurrency at all, which we do not want as well as this will negatively impact our non-functional requirements such as performance.

Call and return architectures

The *main-program-and-subroutine with shared data substyle* is not suitable for us as there will be multiple components that are central to the system, rather than a single component that acts as the orchestrator for the rest.

7. Subsystem Architecture Design

7.1 Subsystem Interface Design – Leaderboard



Leaderboard Interface

This interface supports the viewing of the leaderboard by users, extracting information from Records Database.

Important Interface Functions:

- GetCustomLevelsList (string UserID): List (IEnumerator)
    - This function gets the list of custom levels
- GetScores (string LevelID): IEnumerator
    - This function gets from the Records Database all the scores of users from the custom level database. The leaderboard will be displaying these scores after ranking them.

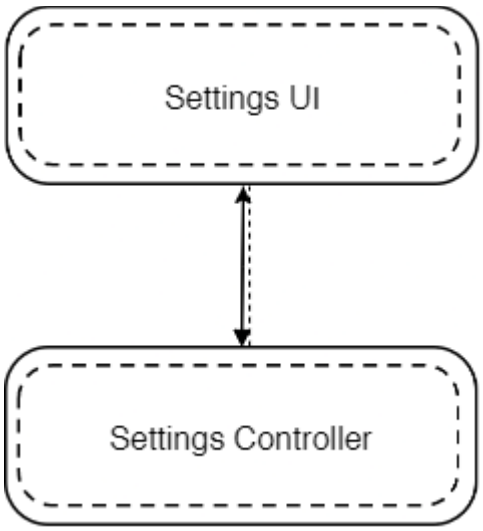
For this subsystem, we chose the classical Batch Sequential data processing model with communications done through temporary intermediate files that can be removed by successive subsystems. The Leaderboard Controller first gets all the existing scores and attempts from the Records Database. Then the functions GetCustomLevelsList and

GetScores will be sieving the data, extracting only relevant records and scores. The information is then passed to Leaderboard UI and further filtered by the UI and displayed in descending order.

There is bi-directional control flow from UI to controller and from the controller to the database as the controller needs to make requests and also filter and format the data from the database, and the database can cause the execution of the controller upon returning the data requested.

There is bi-directional data flow across the components in the form of leaderboard filters being passed from UI to controller to database, and the records being passed from database, controller, and UI.

7.2 Subsystem Interface Design – Settings



Settings Interface

This interface supports the changing of settings options by users.

Important Interface Functions:

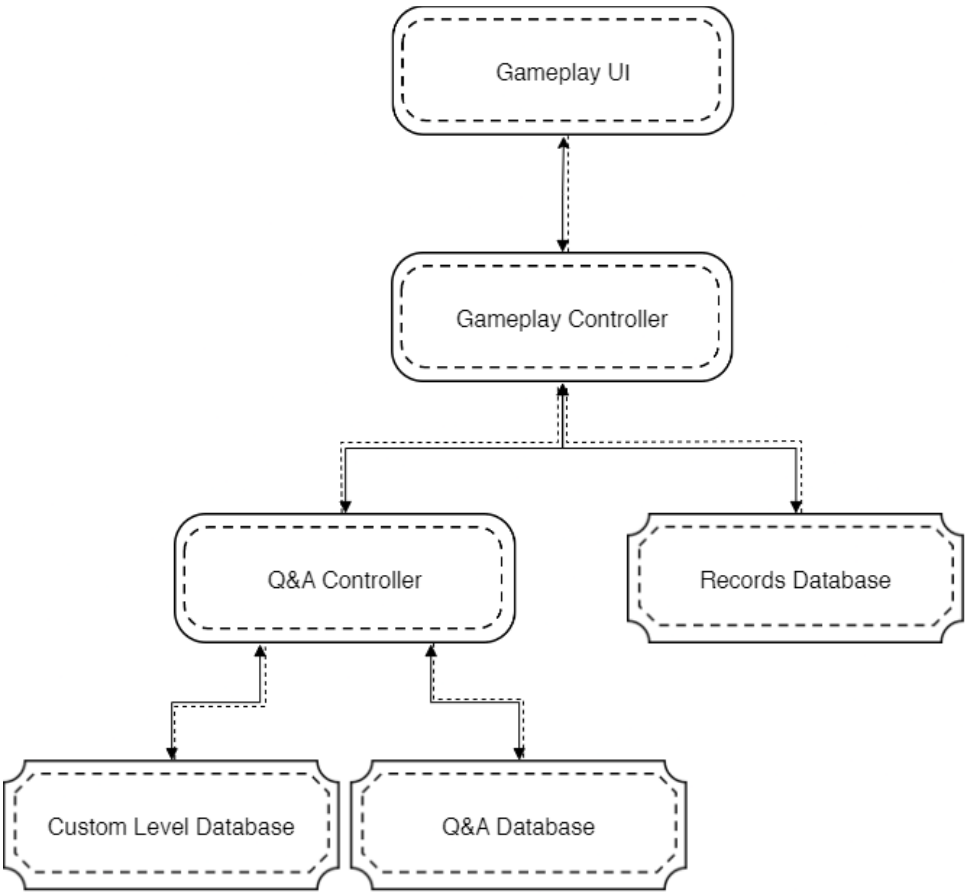
- GetSettingsStates (string UserID): String
    - This function gets the list current states of settings from settings controller in the form of a string
- ChangeSettingsStates (string UserID): bool
    - This function changes the states of setting in settings controller and return a Boolean value, indicating if the change was made successfully

For the Settings subsystem, we chose a Call-and-Return architecture between the Settings Controller and Settings UI. The current state of settings will be collected using the GetSettingsStates function. When the user made changes and click “Save Changes” on the Settings UI. The changes are passed to Settings Controller. The function check if the changes are valid before makings changes and return a True for successful changes made. The Settings UI will then display that the changes have been made.

There is bidirectional data and control flow in this subsystem as we consider the settings to be stateful data, so the data is modified and passed between the components. Control is also passed along both components as displaying the settings and modifying the settings would need control passed between both components.

For this version of the game, the music volume is a changeable setting within the game level.

7.3 Subsystem Interface Design – Gameplay



Gameplay Interface

This interface controls the game under both single-player mode and custom-level mode. Extracting questions from Q&A Database before a game starts and writes into Records Database after a game ends.

Important Interface Functions:

- GetQnString (string LevelID, string QuizID): String
    - This function gets the list of Questions and Answers for the game from Q&A Database
- WriteScores (string LevelID, string UserID): Bool

- This function writes into the Records Database the score of the game just ended and return a Boolean value to indicate if the write action had been done successfully.

For the Gameplay subsystem, we chose a Call-and-Return architecture centered around Gameplay Controller.

Before the game starts, the Gameplay controller will be fetching sets of questions from Q&A Database using the function GetQnString,

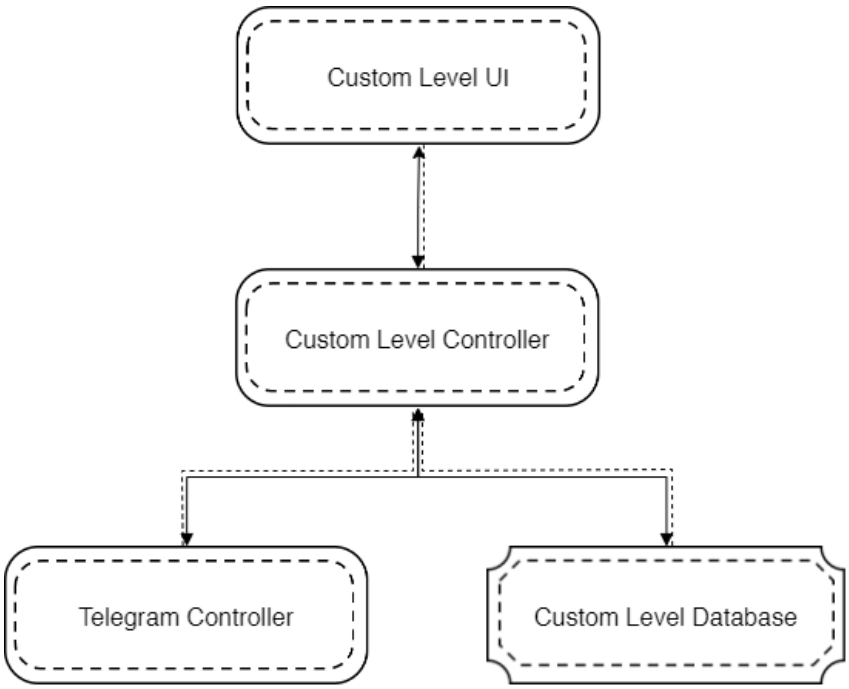
During gameplay, functions such as motions and feedback of correct/incorrect answers will be controlled by Gameplay Controller and displayed by Gameplay UI. These methods are abstracted here as this happens within the Gameplay Controller and not necessarily public methods.

After a game has ended, the Gameplay Controller will write the score into Records Database using function WriteScores for record keeping purposes, for the data to be used in other components such as the progress and leaderboard subsystems.

Q&A controller also interacts with the custom level database for cases where the gameplay level is a custom level for an assignment or a challenge.

We chose to portray the subsystem components as having bidirectional data and control flow, as we design these components to be stateful and executable.

7.4 Subsystem Interface Design – Custom Level



Custom Level Interface

This interface controls the process of users generating custom levels and a unique seed code. Once a level had been created, the questions will be stored in Custom Level Database. The gameplay aspect of custom level is not covered here as it is handled by the gameplay subsystem.

Important Interface Functions

- CreateCustomLevel (int numQn): Boolean
    - This function gets the number of questions that the user wishes to create in a new custom level via Custom Level UI and store it in Custom Level Database
- CreateScrollContent (string Question): Boolean
    - This function gets the list of Questions from users via Custom Level UI and store them in Custom Level Database
- CreateSeed (string QuizID): String

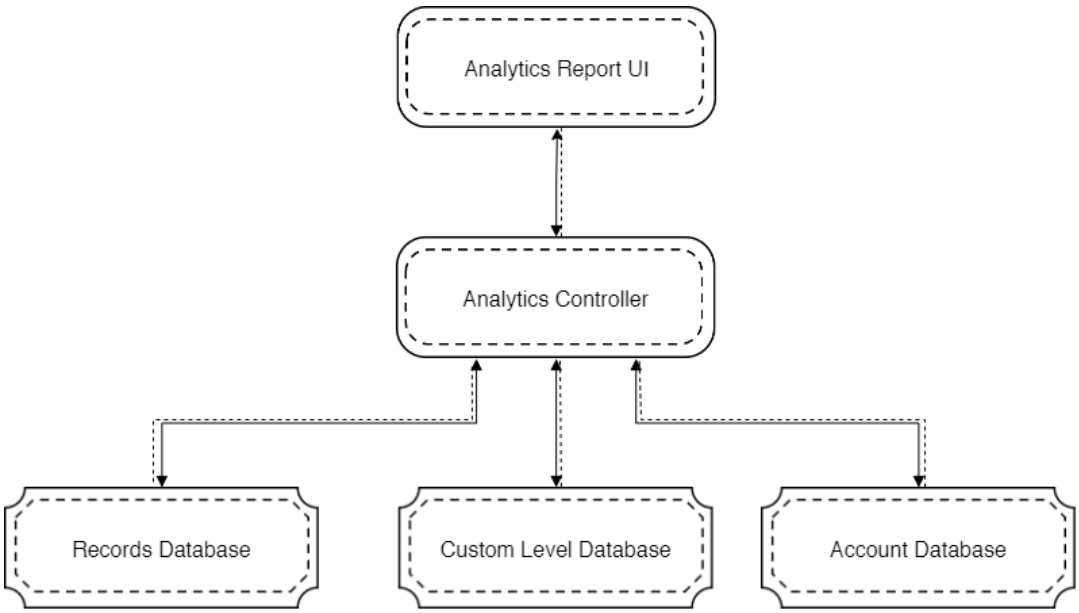
  - This function map the QuizID stored in Custom Level Database to a new unique Seed value and display it on Custom Level UI for user to share with other users

- ShareOnTelegram (string Seed): Bool
  - This function shares the Seed value and calls Telegram API to send the seed value through a Telegram bot. It then returns a Boolean value to indicate that the sharing action was done.

For the custom level interface, we chose a Call-and-Return architecture centred around Custom Level Controller. User will be first typing in the number of questions they wish to include in the new custom level before clicking “Create Level” button on Custom Level UI. Then the function CreateCustomLevel function and CreateSeed function will be invoked. After that, the Custom Level UI will be prompted for user to key in the questions in the level. After the users had keyed in the questions for the level and pressed “Submit”. CreateScrollContent function will be invoked, and the questions will be stored in the Custom Level database and Custom Level UI will be displaying the created seed value for users to share with other users. After user clicked on the “Share” button, the function ShareOnTelegram will be invoked to call the Telegram API (not part of this software scope) and copied over the seed value. If sharing is successful, the function ShareOnTelegram will return a True value and the Custom Level UI will display that the sharing action was done successfully.

We chose to portray the subsystem components as having bidirectional data and control flow, as we design these components to be stateful and executable.

7.5 Subsystem Interface Design – Analytics Report



Analytics Report Interface

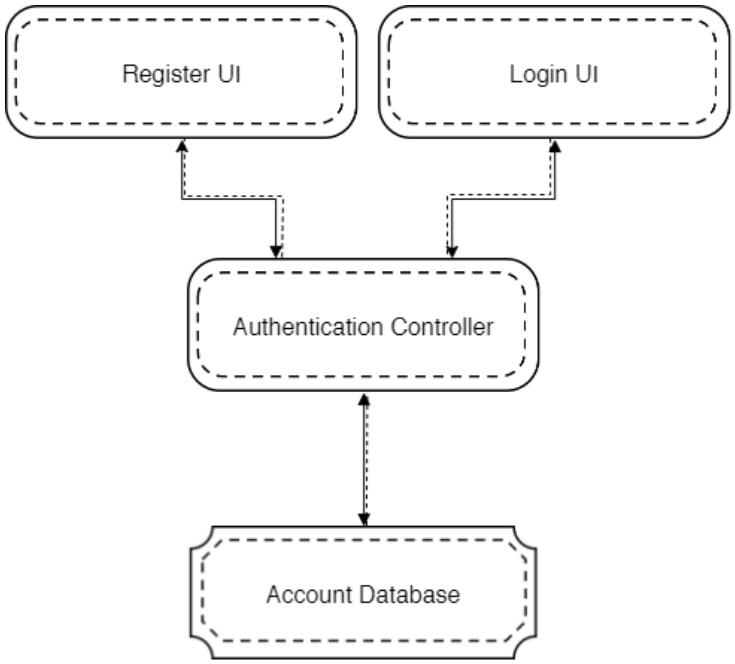
This interface supports the generation of the Analytics Report for teachers to view the mastery of users for worlds, levels, and custom levels, extracting information from Records Database.

Important Interface Functions:

- GetGameList (): List (IEnumerator)
    - This function gets the list of worlds, levels, and custom levels that users have played.
- GetScores (): IEnumerator
    - This function gets the scores of the games played by users from the Records Database.
- GetReport (): IEnumerator
    - This function generates the report, which includes various statistics about the user's performance and mastery of content.

For the Analytics Report subsystem, we chose a Batch Sequential Processing architecture. The Analytics Controller gets the desired game records from the Record Database using the GetGameList and GetScore function. The controller then generates and passes the reports to the Progress Report UI to be displayed.

7.6 Subsystem Interface Design – Account Management



Account Management Interface

This interface controls the process of users logging in and registering for account. User account registration and login are handled by separate user interface components, but are handled by the same controller, and the account credentials are stored in a single account database.

Important Interface Functions

- Register (string Email, string Password, string accessCode, int accountType): string
    - This function performs registration for the user account, given the Email, Password, accountType, and accessCode variables. Internally, there will be validation ongoing for the various inputs, and the returned string will have a success or error message that will be parsed by the UI component.
- Login (string Email, string Password): string
    - This function performs logging in of the user given the inputs. Like the register function, it will perform validation and return a success or error message that will be parsed by the UI component.

8. Testing

Testing is an essential part of our development flow to check whether our system being developed meets the expected requirements and helps to identify any problems or errors that may arise along the way.

8.1 Blackbox testing

For our Blackbox testing, we implement functional testing to check that the system fulfils the functional requirement stated in the SRS document.

Functionality Tested	Expected Results	Actual Results	Verdict
Firebase Authentication	Credential supplied correctly will be direct to Main Menu screen	Main Menu scene appear	PASS
	Invalid credentials will receive a prompt on the screen according to the type of invalid information	Invalid email input will prompt “Invalid Email” Wrong password input will prompt “Wrong Password” Email input that has not been registered will prompt “Account does not exist”	PASS
	Account can be registered and logged in after when proper credentials are input	Account is registered and able to login	PASS
	Account cannot be registered when invalid credentials are input	When passwords do not match, prompt will show “Passwords do not match”	

		When invalid email type is input, prompt will show "Register Failed!" When already registered email is input again for registration, prompt will show "Email is in use"	PASS
Mode and Level Selection	From Main Menu, able to select different modes and levels according to level progress	From Main Menu, able to select different modes Levels after level 1 for each mode are locked initially, each consecutive level is unlocked after completing the previous level	PASS
Playing the Game	Level will launch and run properly	Level will launch and run properly	PASS
	Can move the player character with arrow keys	Can move the player character with arrow keys	
	Question types are based on mode selected	Question types are based on mode selected	
	Question difficulty is based on level selected	Question difficulty is based on level selected	
	Score and combo will be added accordingly to my input	Score and combo will be added accordingly to my input	
	Level ends after questions are finished	Level ends after questions are finished	
	Final score is displayed at the end	Final score is displayed at the end	

Accessing the Leaderboard	Leaderboard loads correctly and displays names and scores Leaderboard updates according to filter set by user	Leaderboard loads correctly and displays names and scores Leaderboard updates according to filter set by user	PASS
Accessing Progress Reports	Progress report loads correctly and displays correct progress of desired student(s)	Progress report loads correctly and displays correct progress of desired student(s)	PASS
Making Custom Levels	Teachers and students can make custom levels	Teachers and students can make custom levels	PASS
Playing Custom Levels	Students can play custom levels made by other students or teachers	Students can play custom levels made by other students or teachers	PASS

8.2 Whitebox testing

For our Whitebox testing we perform a manual unit testing where each subsystem component system is tested individually during development.

8.2.1 White Box Testing in Unity Subsystems

The test cases are as follows:

Flow to be tested	Expected Result	Actual Result
Game Launch	Game successfully launches and shows Login screen	Game successfully launches and shows Login screen.
Player Registration	Player able to enter registration page and create an account. Explanation text displayed when player inputs invalid information.	Player able to enter registration page and create an account Explanation text displayed when player inputs invalid information.
Player Login	Player able to input account information. Explanation text displayed when player inputs wrong account information. Player successfully logs in when correct account information is entered.	Player able to input account information. Explanation text displayed when player inputs wrong account information. Player successfully logs in when correct account information is entered.

	On successful login, changes to Main Menu screen.	On successful login, changes to Main Menu screen.
Mode and Level Selection	From the Main Menu, player can select Runner Game, and the respective Modes and Levels. Levels should be available depending on the player's progress. Consecutive levels are unlocked after the player completes each previous level successfully. When the level is selected, Runner Game is started.	From the Main Menu, player can select Runner Game, and the respective Modes and Levels. Levels are available depending on the player's progress. Consecutive levels are unlocked after the player completes each previous level successfully. When the level is selected, Runner Game is started.
Runner Game	Game starts automatically when launched. Player can move the player character on arrow key input and is properly bounded within the game area.	Game starts automatically when launched. Player can move the player character on arrow key input and is properly bounded within the game area.

	Questions and answers are loaded and displayed properly. Player can move the player character to select answer choices. Score and combo are given according to correct or wrong answer selection. Game will end properly when questions are finished, and player score is displayed accordingly. Player can pause the game, change audio volume and quit the level. Player is brought back to level select when quitting the level.	Questions and answers are loaded and displayed properly. Player can move the player character to select answer choices. Score and combo are given according to correct or wrong answer selection. Game will end properly when questions are finished, and player score is displayed accordingly. Player can pause the game, change audio volume and quit the level. Player is brought back to level select when quitting the level.
Custom Level Creation	Player can select number of questions and set each question individually. Explanation text is displayed when any fields	Player can select number of questions and set each question individually. Explanation text is displayed when any fields

	are input wrongly or left empty. Player is given a custom level seed after setting every question correctly and clicking on Create Player can go back to mode select any time	are input wrongly or left empty. Player is given a custom level seed after setting every question correctly and clicking on Create Player can go back to mode select any time
Custom Level Game	Player can input a level seed that has been created previously Prompt is displayed when invalid seed is input Correct questions and answers are loaded according to when it was created	Player can input a level seed that has been created previously Prompt is displayed when invalid seed is input Correct questions and answers are loaded according to when it was created

*Players refer to the student or anyone playing the game.

8.2.2 White Box Testing in Firebase Subsystem Component

Flow to be tested	Expected Result	Actual Result
Registration	Player account is created in Firebase when registered in the game Player account is not created in Firebase when invalid information is input in the game	Player account is created in Firebase when registered in the game Player account is not created in Firebase when invalid information is input in the game
Login	Player account is loaded properly from Firebase on valid login information No account information is loaded when login information is invalid	Player account is loaded properly from Firebase on valid login information No account information is loaded when login information is invalid
Player Info	Correct player information such as name, class, and level progress are fetched properly on successful login	Correct player information such as name, class, and level progress are fetched properly on successful login
Custom Level	Custom Level seed and questions are stored in Firebase properly when created in game	Custom Level seed and questions are stored in Firebase properly when created in game

	Custom Level questions are fetched correctly according to the seed input by the user	Custom Level questions are fetched correctly according to the seed input by the user

8.2.3 Unit Testing

Test Stubs and Drivers

Stubs and drivers in software testing refer to dummy portions of code that are used in the place of either a sub-module or major module within the program respectively. An example of such usage in our project is shown below:

```
[UnityTest]
0 references
public IEnumerator GetUserInfo()
{
    yield return SceneManager.LoadSceneAsync("Login");
    FirebaseManager fm = FirebaseManager.Instance;
    yield return new WaitUntil(() => fm.instantiated);
    FirestoreManager fsm = FirestoreManager.Instance;
    yield return fm.Login(
        "yoho@mail.com",
        "123456");

    yield return fsm.GetUserWorldProgress(res =>
    {
        Assert.IsNotNull(res);
    });
}
```

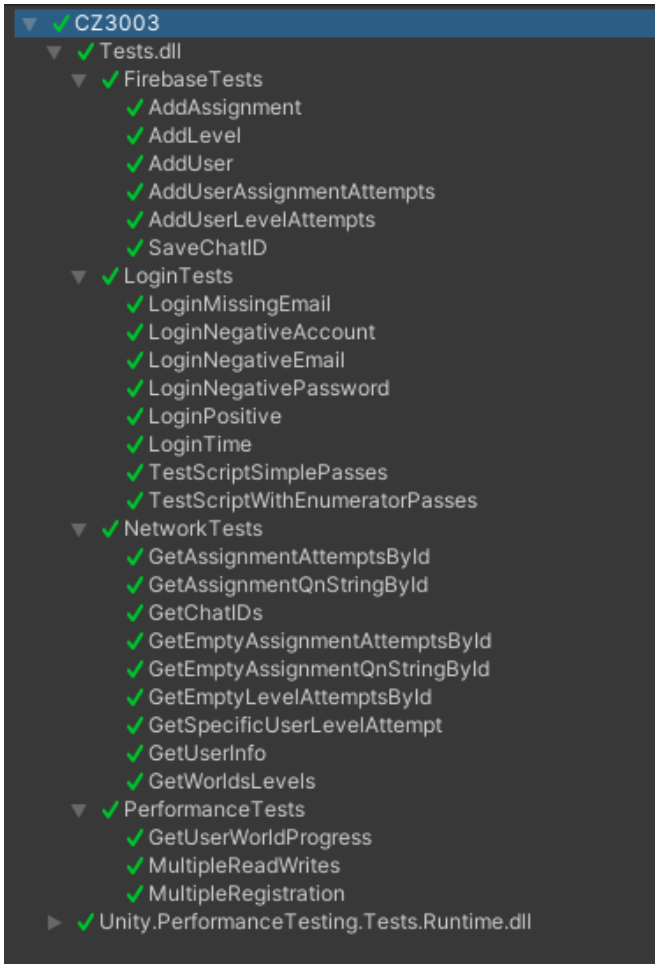
The code block outlined in red acts as a driver that takes the place of the scene and level management to be able to perform a user login before we can perform a retrieval of user information for the test case.

For our project, we made use of the Unity testing framework available in the editor, Unit Test Framework to perform our unit tests, functional tests, and performance tests. The unit tests from 8.2.2 and 8.2.3 can be seen in the figure below.

We split the categories into:

1. **FirebaseTests**, which test the ability of the program to communicate with our Firestore Database to update information to the database correctly.
2. **LoginTests**, which checks functions related to Firebase Authentication. Registering and logging in of user accounts with different inputs and ensuring the correct response is attained.
3. **NetworkTests**, with a focus on the retrieval of information from our Firestore Database both accurately and efficiently.

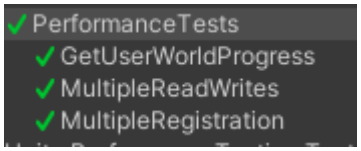
As Unity projects tend to work with game objects and scripts in different scenes, we made use of drivers to take the role of the various manager classes to test different functions.



The diagram above shows that all test cases passed, as seen by the green check marks.

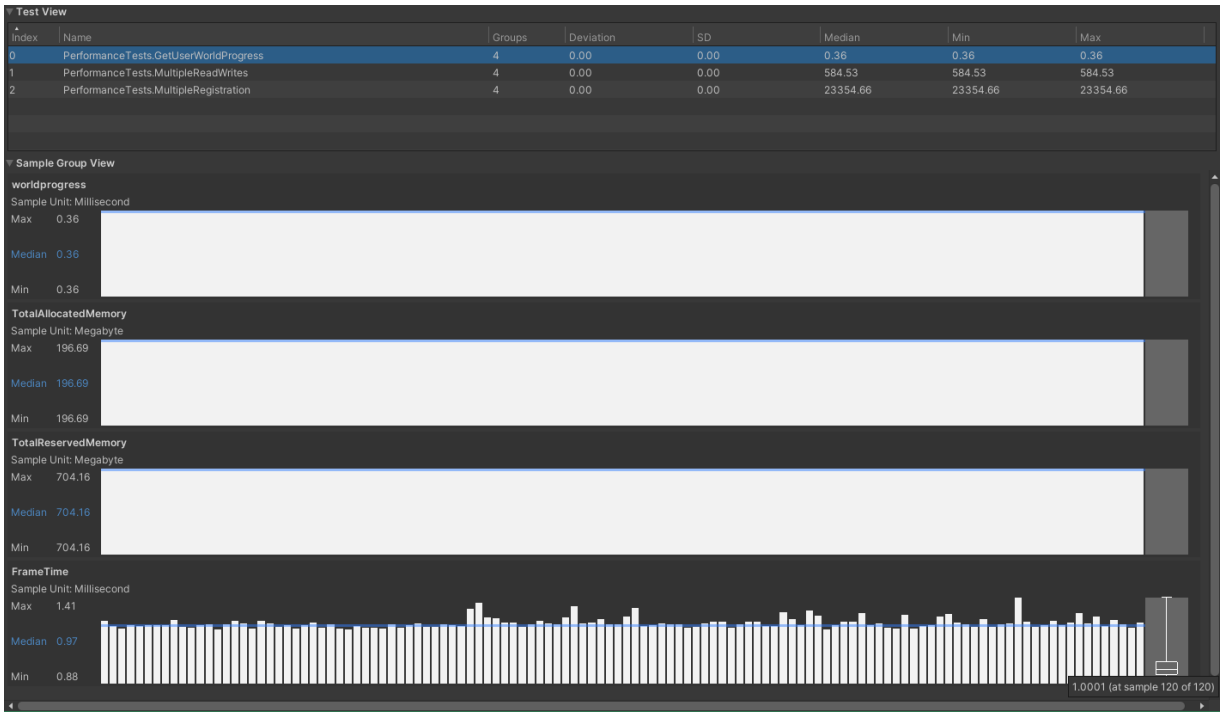
8.3 Performance/Load Testing

For performance testing of our software, we made use of Unity's performance testing package, "Unity.PerformanceTesting" to visualize and test for stability in the program under different loads. Using this, we can measure the speed, average frame times and memory usage of any blocks of code in our test scripts. The performance tests passing can be seen in the figure below.



8.3.1 In-Game Performance

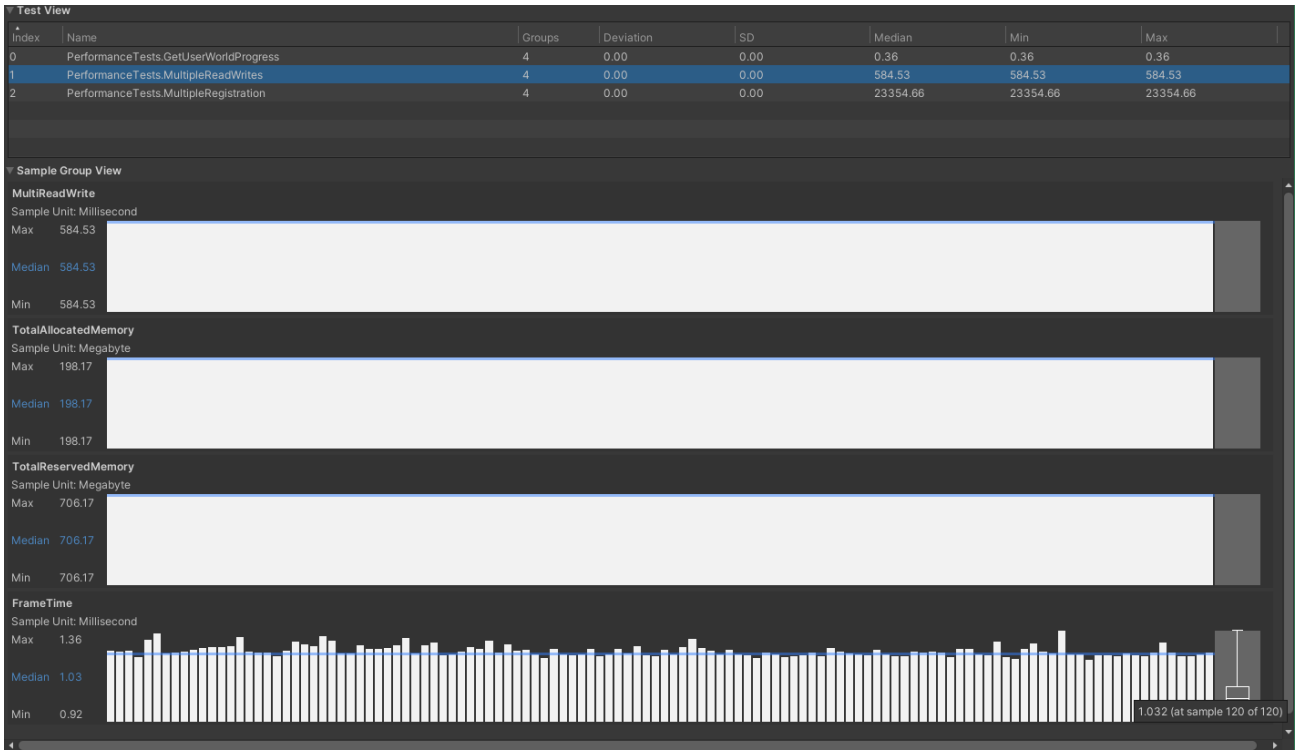
We define performance tests for two critical functions that needs to be working within expectations to ensure a smooth user experience. We have a function that reads a user's level progression through the game that has to be run and completed upon logging in in order to reflect the user's progress accurately. Running the performance test for GetUserWorldProgress, we can get the performance report below:



We confirm that the time taken for execution is 0.44 milliseconds and that the memory usage is as expected. We can also verify that the frame time across 120 frames is stable and did not suffer any performance loss.

8.3.2 Database Operation Performance

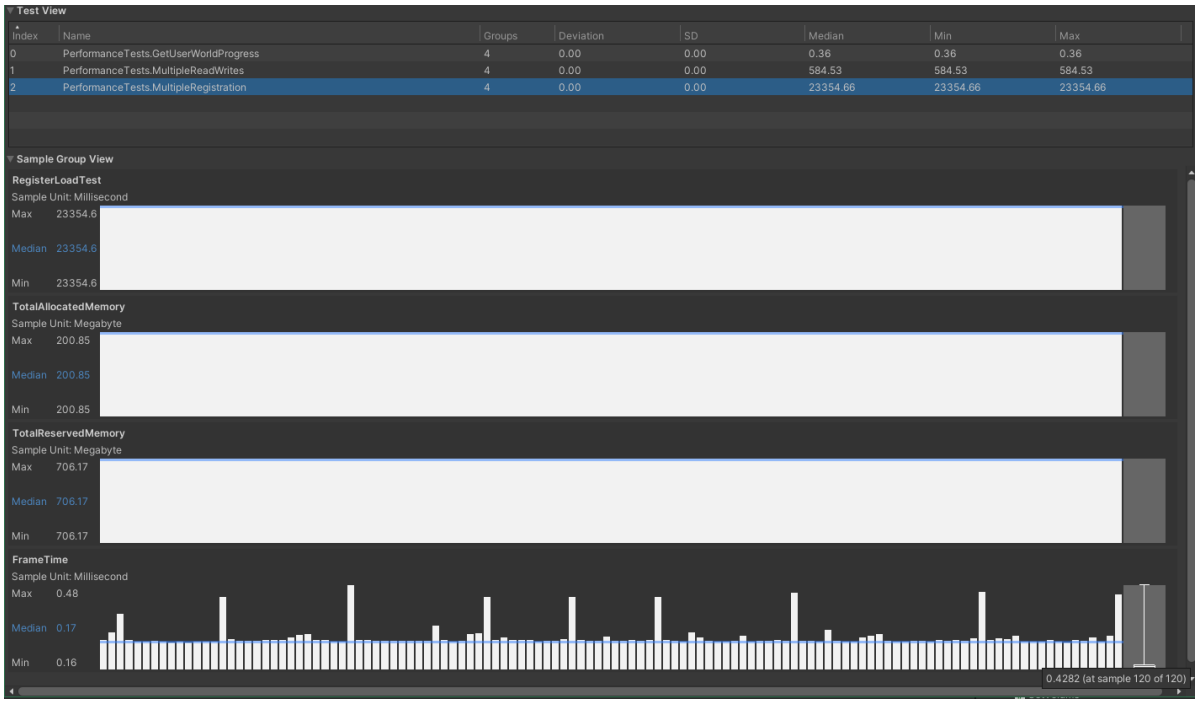
Similarly, there are cases where many students could be playing our game at the same time and there could be multiple read-and-write operations being sent to the database from various students. We test an extreme case of this where all the read and write operations come from a single student using the performance test for MultipleReadWrites.



Even with the assumption that all the read and write operations come from a single student's machine, the measured time is still under 1 second with consistent frame times across 120 frames. We can confirm that the database is able to handle high loads and that the client side is also able to handle even unusually high loads for any use case of a single student.

8.3.3 Account Authentication Performance

We performed a similar test for the Firebase authentication system simulating another extreme scenario where many students are registering an account, then logging in, logging out and finally deleting their accounts all at once. The summary of results are as follows:



This time we notice a performance impact where the total time taken is about 23 seconds and frame times are much lower across 120 frames. Again, this is assuming that all read and write operations come from a single machine. We use these 2 extreme tests to attain some information on where the upper load limits of our system lie during such operations.

As seen above, the performance suffers a drop only under very extreme conditions outside of the normal operating procedures, conditions and load limits and we have determined that our program can meet the performance expectations.

The tests conducted above show that the performance requirements that were laid out in the non-functional requirements section have all been fulfilled by the system.

Appendix A: Data Dictionary

Term	Definition
Student	A student is a user who is supposed to play the game for learning purposes
Teacher	A teacher is a user who acts as the administrator of the students
Game	The game refers to the software system
Leaderboard	A representation of data in a format which students can view scores for all combinations of worlds, levels and custom levels
Single player	Single player mode is a mode where the student plays the game by themself, this is the default mode of the game when user click on “Gameplay” button
Custom Level	Custom levels are special games that teachers and students can create by setting their own questions and number of questions
Profile	Each user has a profile, which includes basic information of the user such as the username, email address, account type and password
Login	Users need to login with a valid email and password in order to access the game’s functionalities.
World	There are four different worlds in the game, each representing different kinds of mathematical operations: Addition, Subtraction, Multiplication and Division.
Level	Within each game world, there will be several levels; students must complete the levels in sequence. Teacher users will have all levels available for them to test.
Combo	A set of actions performed in sequence, usually with strict timing limitations, that yield a significant benefit or advantage. In our game, the set of actions is referring to consecutively answering the questions correctly.
Seeds	Seeds are referring to custom level IDs generated at the time of Custom Level creation. Users must provide this seed before they play the custom level.