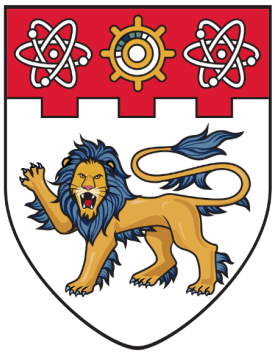

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SCSE23-0659

Metaverse for Virtual Education 1

Submitted by:

Lau Chen Yi Wynne (U2020016B)

Supervisor:

Prof Dusit Niyato

Examiner:

Prof Thambipillai Srikanthan

School of Computer Science and Engineering

Academic Year 2023/2024

Abstract

Several studies have indicated that the use of higher quality VR equipment provides a more immersive experience for its users, however, not all education institutes have the budget to invest in the more expensive and higher quality VR equipment for every one of their students. Hence, this project focuses on the development of a metaverse prototype for applications in virtual education to help educational institutes manage the use of the limited amount of VR equipment. The prototype is built using Unity, a Unity Technologies 3D/2D game development platform. Major features of the prototype include: providing users with functions such as the scheduling and reservation of the use of VR equipment, and collecting usage data efficiently for personalization and performance improvement. Managing the resource allocation of the limited quantity of VR equipment will greatly contribute to the adoption of Metaverse-based education.

Acknowledgements

First and foremost, I would like to express my deep and sincere gratitude to my supervisor, Prof Dusit Niyato, President's Chair Professor in Computer Science and Engineering, Nanyang Technological University (NTU), Singapore, for his patience and guidance throughout this project. He has provided me invaluable insights into the methodology to carry out the project and I am extremely grateful for what he has offered me.

I am also very grateful for the help from netizens Code Monkey, Blue Mask Games, ETAD Lab, Binary Lunar and the Unity Community. They have provided me with vital knowledge of Unity, which is essential for the progress of this project.

I would like to extend my gratitude to netizens GMGStudio, who has imparted me with knowledge on connecting Unity and MongoDB using MongoDB's Atlas Device SDK, and Microsoft Azure, who has equipped me with the knowledge of Azure PlayFab's REST API.

Finally, I would like to thank my family and friends for their kind consideration and support throughout this project.

Table of Contents

List of Tables

List of Figures

1 Introduction

The Metaverse introduces a new and revolutionary concept that allows users to engage in a virtual world, where one can seamlessly interact with other users and their digital surroundings in real-time regardless of where they physically are. Over the past few years, the Metaverse has had significant impacts on multiple industries, ranging from gaming to retail, and is expected to impact many more.

In the increasingly popular landscape of the Metaverse, the education industry emerges as a pivotal domain that stands to gain significantly from its advantages. Traditional education methods often require the physical presence of both lecturers and students at a specific time and location. However, commuting between locations consumes valuable time which can lead to reduced productivity for both the educators and learners. The Metaverse presents us with an opportunity to transform education by providing immersive and interactive environments for teaching and learning from the comforts of one's home [1].

Building upon these advantages, educational institutes worldwide are increasingly adopting the Metaverse as a transformative tool for enhancing the teaching and learning experience. According to the Metaverse: market data & analysis report (2023) by Statista, the value of the Metaverse Education market is projected to reach a value of US$2.5 billion in 2024 and is expected to have a compound annual growth rate of 46.14% from 2024 to 2030, resulting in a projected market volume of US$24.7 billion by 2030 [2].

Despite the increasing popularity of the adoption of Metaverse for education, there are still significant challenges to Metaverse-based education. Such challenges include the challenge of resource allocation due to the lack of proper infrastructure. The implementation of immersive Metaverse-based education often requires adequate technological infrastructure, which includes expensive hardware equipment like VR headsets and VR gloves. However, not all educational institutes can afford to provide every one of their students with VR equipment.

Metaverse-based education applications created thus far have provided users with an immersive virtual environment to learn and collaborate in. However, even though it is well known that most educational institutes have a limited budget to invest in VR equipment, there is a lack of attempts to help educational institutes manage the use of expensive hardware by developers of the Metaverse-based education applications to ensure the smooth adoption of Metaverse for education.

Therefore, this project attempts to develop a Metaverse prototype using Unity to provide users with functions such as the scheduling and reservation of VR equipment available in an educational institute. Furthermore, we attempt to collect data efficiently on the usage of virtual meeting rooms for personalization and performance improvement. Data such as the emails of the participants in a meeting will be collected and stored in a database.

2 Literature Review

Statistics have shown that there has been an increase in the adoption of Metaverse-based education by educational institutes all over the world and is projected to continue its upward trend, and Singapore is no exception. According to the Metaverse: market data & analysis report (2023) by Statista, the Metaverse Education market in Singapore is expected to reach a value of US$8.6 million in 2024 and is projected to have an annual growth rate of 49.00% from 2024 to 2030, resulting in a projected market volume of US$93.9 million by 2030 [2].

This upward trend is mainly due to the fact that Metaverse-based education has been proven to be more effective than traditional education methods. Existing studies show that benefits such as easy access to Metaverse-based education regardless of physical location and time, and a more interactive teaching and learning experience for users are some reasons why Metaverse-based education is an effective learning tool. Despite having limitations such as the high cost of VR equipment, it is believed that the benefits outweigh its limitations and is the future of education [3]. As a result of this upward trend in the Metaverse Education market, there has been an increasing number of applications made for Metaverse-based education.

These applications are often designed to target different educational disciplines. Some will bring students on virtual field trips, while others will allow students to conduct dangerous experiments safely. These applications mostly support the use of hardware devices such as tablets, laptops, VR headsets and many more. However, previous studies mentioned that in order to provide an immersive experience to its users, the Metaverse applications created for education require the use of higher quality VR equipment like VR headsets and VR gloves, which are often much

more expensive than devices like tablets and laptops [4], [5]. Even worse, educational institutes often need more money to invest in expensive VR equipment due to their limited budget.

The limited budget would result in an insufficient quantity of VR equipment that could be used and experienced by students of an educational institute. Also, using less immersive devices like tablets and laptops would significantly affect the student's experience of Metaverse-based education [6]. The lack of VR equipment in an educational institute greatly hinders the adoption of immersive Metaverse-based education, but little has been done by developers of the Metaverse application to help educational institutes manage the use of the limited quantity of VR equipment. Therefore, there is an urgent need to help educational institutes manage the use of the expensive hardware.

Hence, this project attempts to develop a Metaverse prototype to provide its users with functions such as the scheduling and reservation of the VR equipment that has been made available for use. In addition, in an attempt to personalize and improve the performance of the application, we will collect data efficiently on the usage of virtual meeting rooms.

3 Project Timeline and Resources

This section contains the project timeline and resources used for the development of the Metaverse prototype.

3.1 Project Timeline

Table 1: Project Timeline

Start date	End date	Duration (days)	Task details
14/08/23	20/08/23	7	1. Prepare Project Plan
21/8/23	27/8/23	7	1. Set up the working environment a. GitHub b. Unity 2. Learn Unity basics
28/8/23	3/9/23	7	1. Create LoginUI Scene 2. Create Login UI and its functions 3. Create Register UI and its functions 4. Create SetUserProfile UIs and its functions 5. Connect to PlayFab database through Azure PlayFab API calls
4/9/23	17/9/23	14	1. Create Main Scene 2. Create LogOut UI and its functions 3. Create UserProfile UIs and its functions 4. Create player controls 5. Create 1st-person and 3rd-person cameras in Main Scene
18/9/23	24/9/23	7	1. Create MainMenu UIs and its functions 2. Create ResourceReservation UIs and its functions 3. Create ManageSlots UI
25/9/23	29/10/23	35	1. Create ManageSlots UI functions 2. Create DateDetails UIs and its functions 3. Create TimeDetails UIs and its functions 4. Connect to MongoDB Atlas using Atlas Device SDK

30/10/23	19/11/23	21	1. Create MeetingSchedule UI and its functions 2. Create NewMeeting UI and its functions 3. Create ClassRoom Scene 4. Create local multiplayer
11/12/23	31/12/23	21	1. Create MeetingDetails UI and its functions 2. Create JoinMeeting UI and its functions 3. Create online multiplayer 4. Create NetworkManager UI and its functions
1/1/24	28/1/24	28	1. Ensure persistence of binding of GameObjects to scripts in Main Scene 2. Create 1st-person camera in ClassRoom Scene 3. Interim Report
29/1/24	25/2/24	28	1. Create audio chat in ClassRoom Scene 2. Create EnableQuiz UI and its functions 3. Create SampleQuiz UI and its functions 4. Create EnableClass UI and its functions 5. Create JoinCodeError UI and its functions
26/2/24	24/3/24	28	1. Testing of functionalities 2. Complete Final Report

3.2 Project Resources

3.2.1 Hardware

Table 2: List of hardware used and their specifications and descriptions

Hardware	Specifications	Description
Lenovo Yoga C740-14IML	**Processor:** Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz **RAM:** 16.0 GB (15.8 GB usable) **Operating System:** Microsoft Windows 11 Home	This device was used for the development and testing of the Metaverse prototype.
Acer Swift SF314-57G	**Processor:** Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz **RAM:** 16.0 GB (15.8 GB usable) **Operating System:** Microsoft Windows 11 Home	This device was used to test the online multiplayer functionality of the Metaverse prototype.

3.2.2 Software

Table 3: List of software used and their descriptions

Software	Description
GitHub Desktop	A free and open source application that allows for easier version control.
Unity Editor 2022.3.6f1	A Unity Technologies 3D/2D game development platform used to develop the Metaverse prototype.
Unity Gaming Services Multiplayer	A Unity Technologies service that provides support to multiplayer applications. This is used to develop the multiplayer meeting sessions and the voice chat of the Metaverse prototype.
Microsoft Visual Studio 2022	A Microsoft Integrated Development Environment (IDE) used to develop the Metaverse prototype.
Azure PlayFab	A Microsoft back-end platform that manages the login and registration of users. It is also used to store and manage personal data.
MongoDB Atlas	A multi-cloud database service that manages and stores non-personal data such as the scheduling and reservation of the use of VR equipment, meeting details and participants of meetings.

4 Design and Implementation

This section discusses the design and implementation of the Metaverse prototype. There are 3 scenes created in total, namely the LoginUI Scene, Main Scene and ClassRoom Scene. When the application starts, only the LoginUI Scene will be loaded.

There are three different account types, namely “Student”, “Professor/TA” and “Lab Admin”. The different account types will provide users with access to different functions of the application.

As shown in Figure 1 below, assuming that the account has already been registered, the “Student” account type will provide its users with functions such as:

1. logging into their “Student” account in Login UI,
2. quitting the application in LogOut UI,
3. updating user profile in UserProfileStudent UI,
4. retrieving details of a meeting in MeetingDetails UI,
5. starting a meeting created by themselves in MeetingDetails UI,
6. deleting a meeting created by themselves in MeetingDetails UI,
7. creating a new meeting in NewMeeting UI,
8. joining an ongoing meeting created by others in JoinMeeting UI.

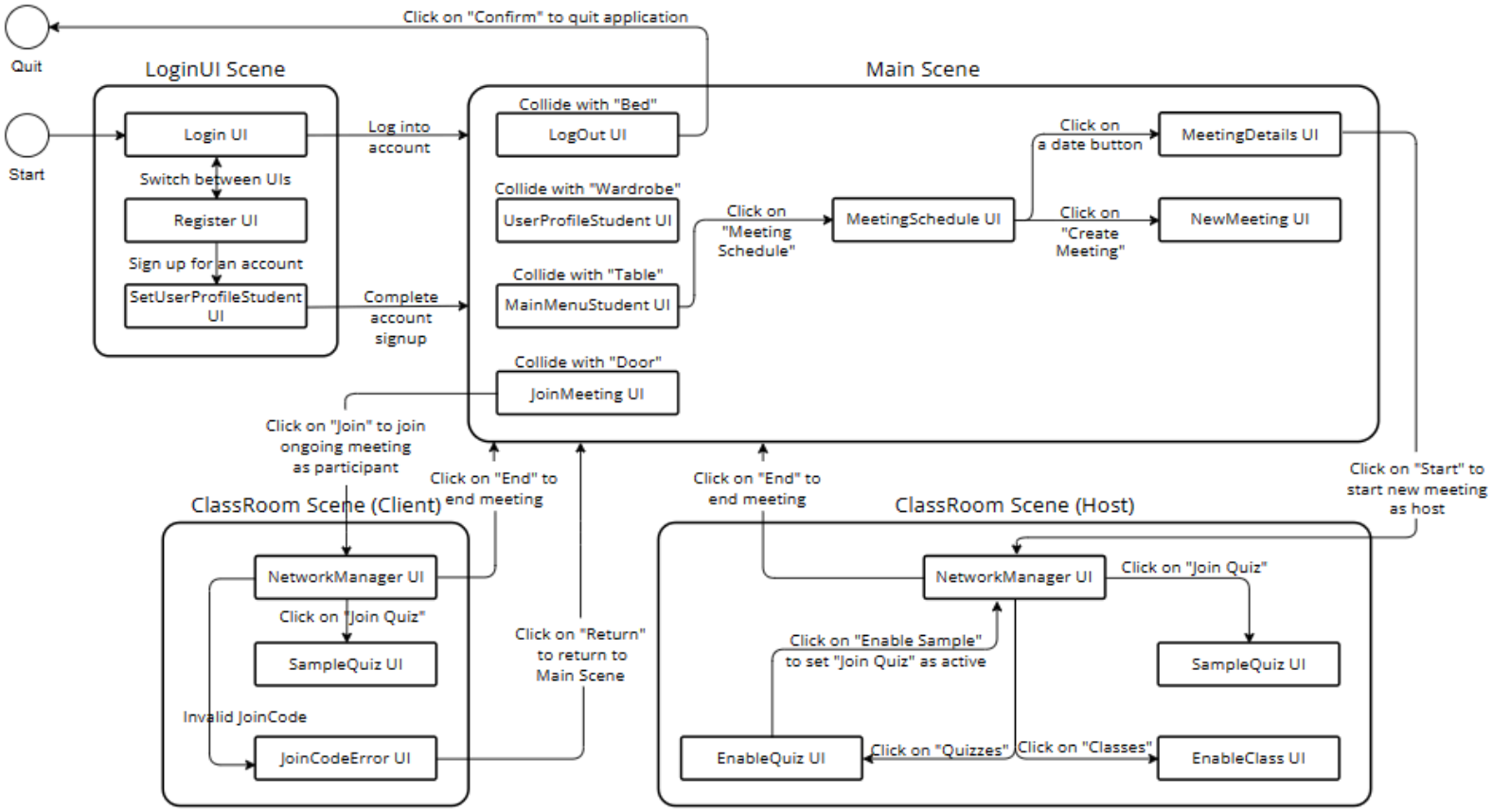

Figure 1: Overall architecture of the "Student" account

Similarly, as shown in Figure 2 below, assuming that the account has already been registered, the "Professor/TA" account type will provide its users with functions such as:

1. logging into their "Professor/TA" account in Login UI,
2. quitting the application in LogOut UI,
3. updating user profile in UserProfileOthers UI,
4. retrieving details of a meeting in MeetingDetails UI,
5. starting a meeting created by themselves in MeetingDetails UI,
6. deleting a meeting created by themselves in MeetingDetails UI,
7. creating a new meeting in NewMeeting UI,

8. reserving an available time slot in TimeDetailsProf UI,
9. cancelling a reserved time slot in TimeDetailsProf UI,
10. joining an ongoing meeting created by others in JoinMeeting UI.

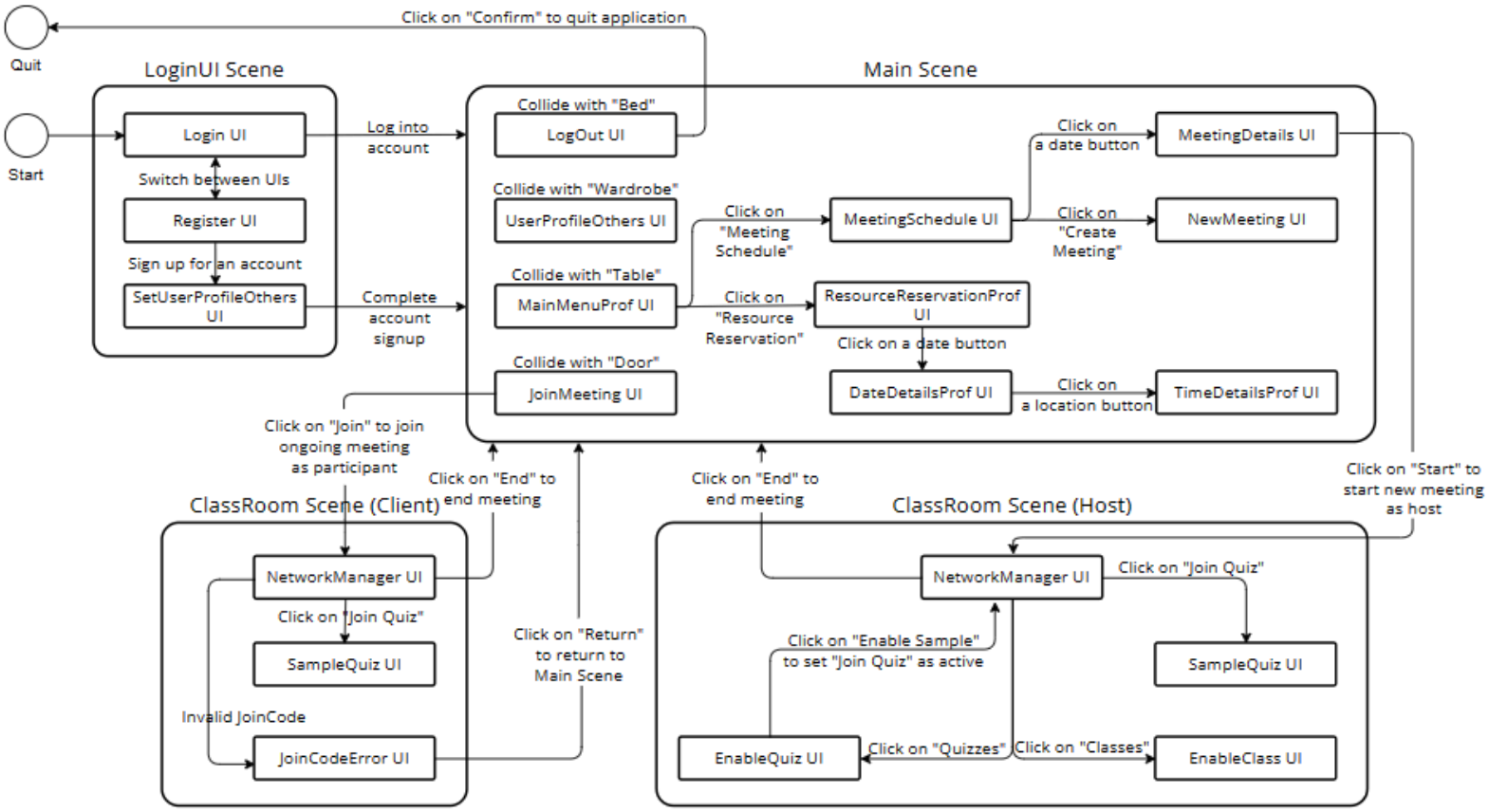

Figure 2: Overall architecture of the "Professor/TA" account

Similarly, as shown in Figure 3 below, assuming that the account has already been registered, the "Lab Admin" account type will provide its users with functions such as:

1. logging into their "Lab Admin" account in Login UI,
2. quitting the application in LogOut UI,
3. updating user profile in UserProfileOthers UI,
4. retrieving details of a meeting in MeetingDetails UI,

5. starting a meeting created by themselves in MeetingDetails UI,
6. deleting a meeting created by themselves in MeetingDetails UI,
7. creating a new meeting in NewMeeting UI,
8. adding available time slots in ManageSlots UI,
9. removing available time slots in ManageSlots UI,
10. retrieving details of a reserved time slot in TimeDetailsStaff UI,
11. joining an ongoing meeting created by others in JoinMeeting UI.

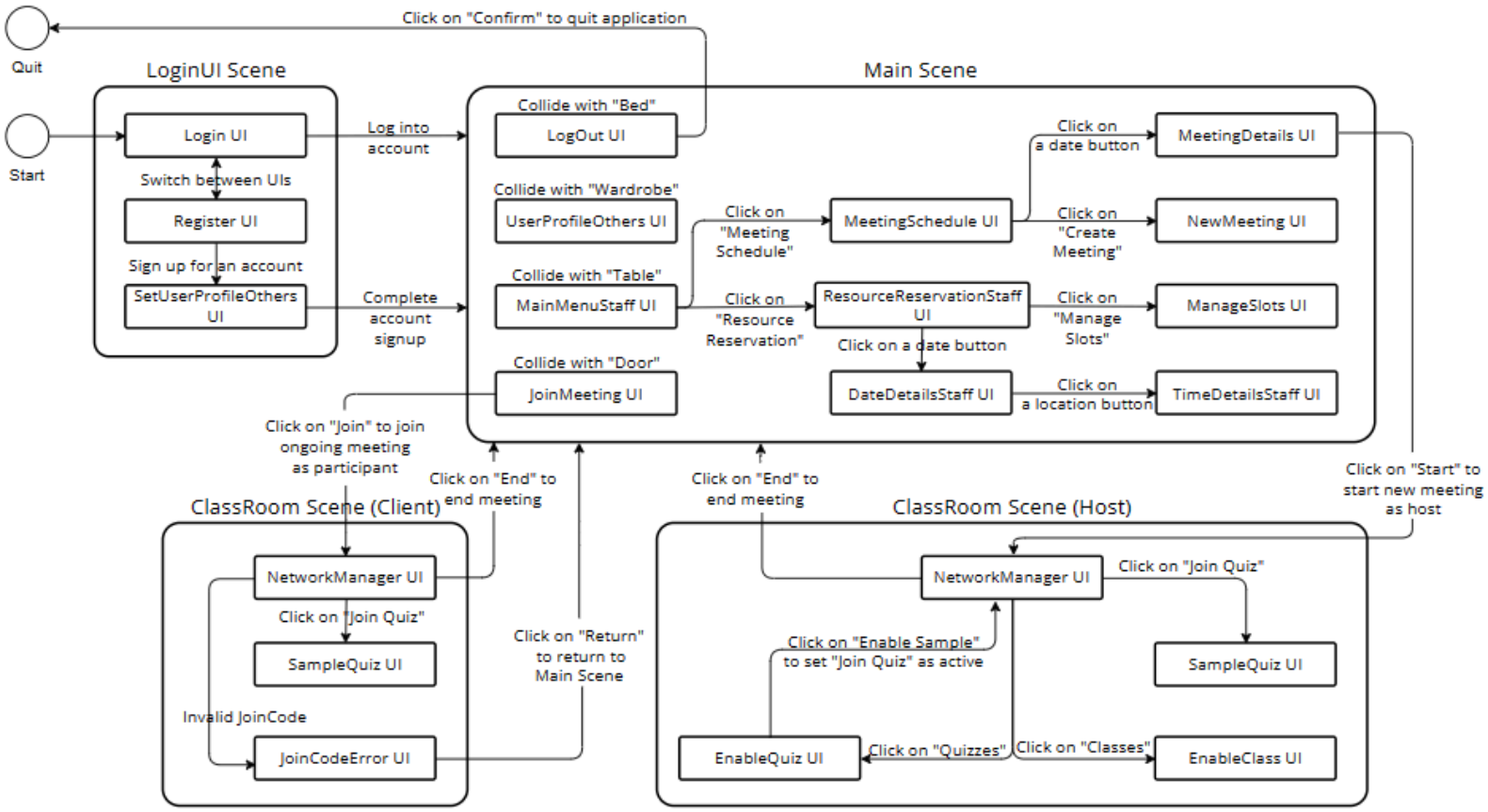

Figure 3: Overall architecture of the "Lab Admin" account

All the above accounts will have access to multiplayer functions in ClassRoom Scene according to their MeetingStatus, host or client (See Figures 1, 2, & 3).

4.1 LoginUI Scene

This subsection discusses the design and implementation of the LoginUI Scene. This scene consists of a 2D Canvas containing the Login, Register and SetUserProfile UIs. When the application starts, only the Login UI will be set as active.

4.1.1 Login UI

The Login UI allows users to log into their registered account. After providing the registered email and password, the user can either click on the "Log in" button (See Figure 4) or press on the "Enter" key to log into their account.

Figure 4: Screenshot of Login UI

As shown in Figure 5 below, when trying to log into an account, we will attempt to log in using a PlayFab API call. When the attempt to log in is successful, the Main Scene will be loaded. Otherwise, an error message will be shown in the Login UI to inform the user of the error.

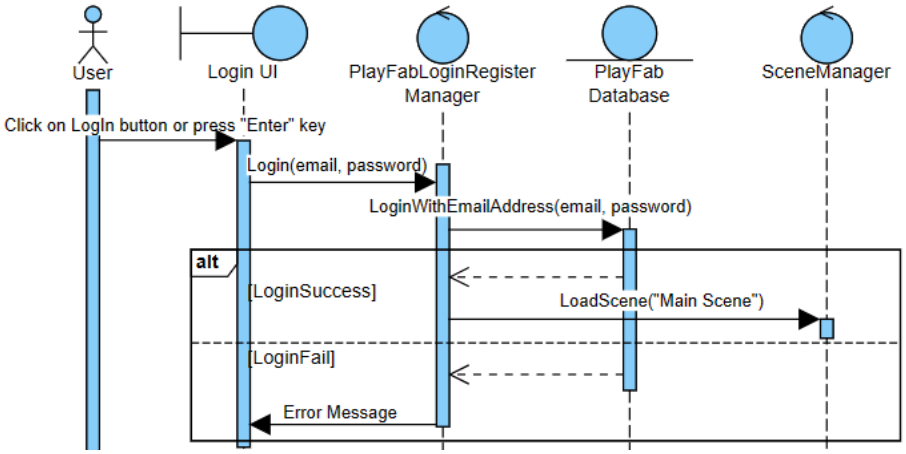

Figure 5: Sequence diagram for account login

The Login UI also consists of a "Register here" button (See Figure 4) that allows users to toggle to the Register UI to register for an account.

4.1.2 Register UI

The Register UI allows users to register for an account using their email and a valid password. Users will have to indicate their identity as "Student", "Professor/TA" or "Lab Admin" via the dropdown. After providing the required information, the user can either click on the "Register" button (See Figure 6) or press on the "Enter" key to register for an account.

Figure 6: Screenshot of Register UI

As shown in Figure 7 below, when trying to register for an account, we will attempt registration using a PlayFab API call. When the attempt to register is successful, followed by a successful user data update, the respective SetUserProfile UIs will be set as active. Otherwise, an error message will be shown in the Register UI to inform the user of the error.

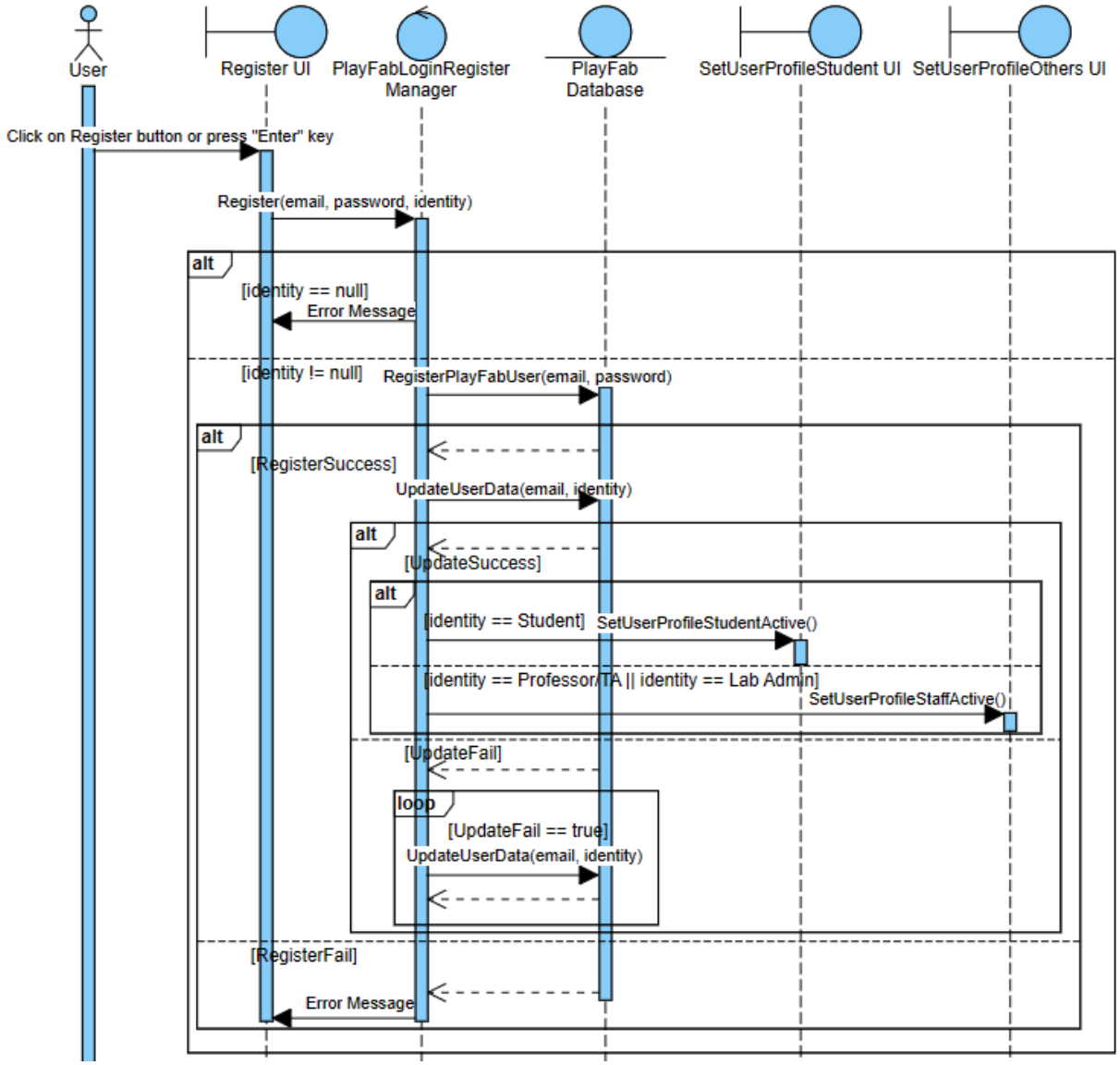

Figure 7: Sequence diagram for account registration followed by a user data update

The Register UI also consists of a “Log in here” button (See Figure 6) that allows users to toggle to the Login UI to log into an existing account.

4.1.3 SetUserProfile UIs

There are two types of SetUserProfile UIs, namely the SetUserProfileStudent UI and SetUserProfileOthers UI. When registration is successful from the Register UI, the respective SetUserProfile UIs are set as active. If the user is a “Student”, the SetUserProfileStudent UI will be set as active. Otherwise, if the user is a “Professor/TA” or “Lab Admin”, the SetUserProfileOthers UI will be set as active instead.

The SetUserProfileStudent UI requires the users to provide inputs such as the Preferred Display Name, School, Course and Year. It is optional to provide additional information about the users themselves. After providing the required information, the user will either click on the “Continue” button (See Figure 8) or press on the “Enter” key to continue.

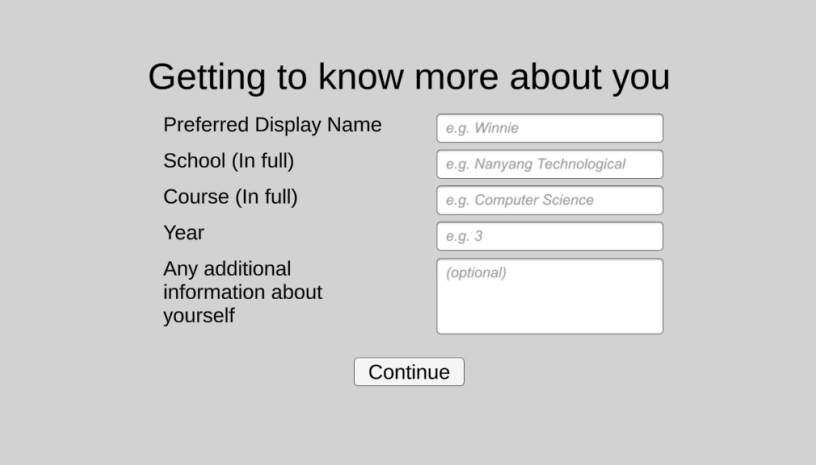

Figure 8: Screenshot of the SetUserProfileStudent UI

On the other hand, the SetUserProfileOthers UI only requires users to provide inputs such as the Preferred Display Name and School. Similarly, it is optional to provide additional information about the users themselves. After providing the required information, the user will either click on the “Continue” button (See Figure 9) or press on the “Enter” key to continue.

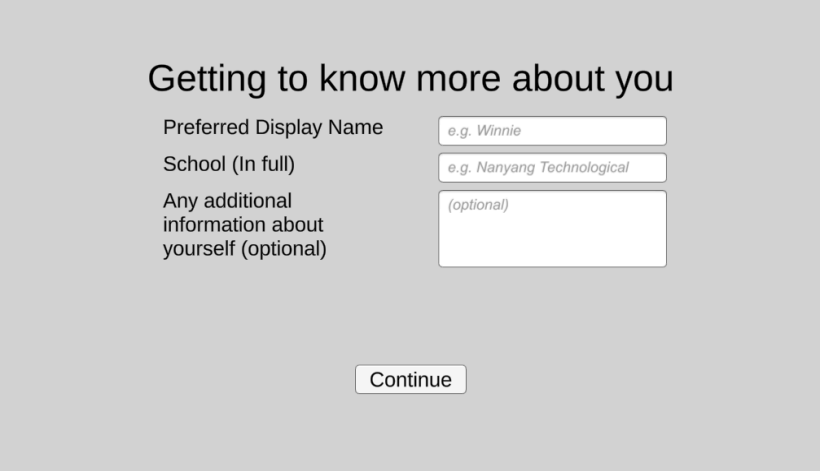

Figure 9: Screenshot of the SetUserProfileOthers UI

The two types of SetUserProfile UIs have a similar sequence in storing the user's profile details. As shown in Figure 10 below, when attempting to complete the registration process, if there is no missing required information, we will attempt to update the user's data using a PlayFab API call. For "Student", the Preferred Display Name, School, Course, Year and Description will be updated, while only the Preferred Display Name, School and Description will be updated for the "Professor/TA" or "Lab Admin". When a user's profile details are updated successfully, the Main Scene will be loaded. Otherwise, an error message will be shown in the respective SetUserProfile UIs to inform the user of the error.

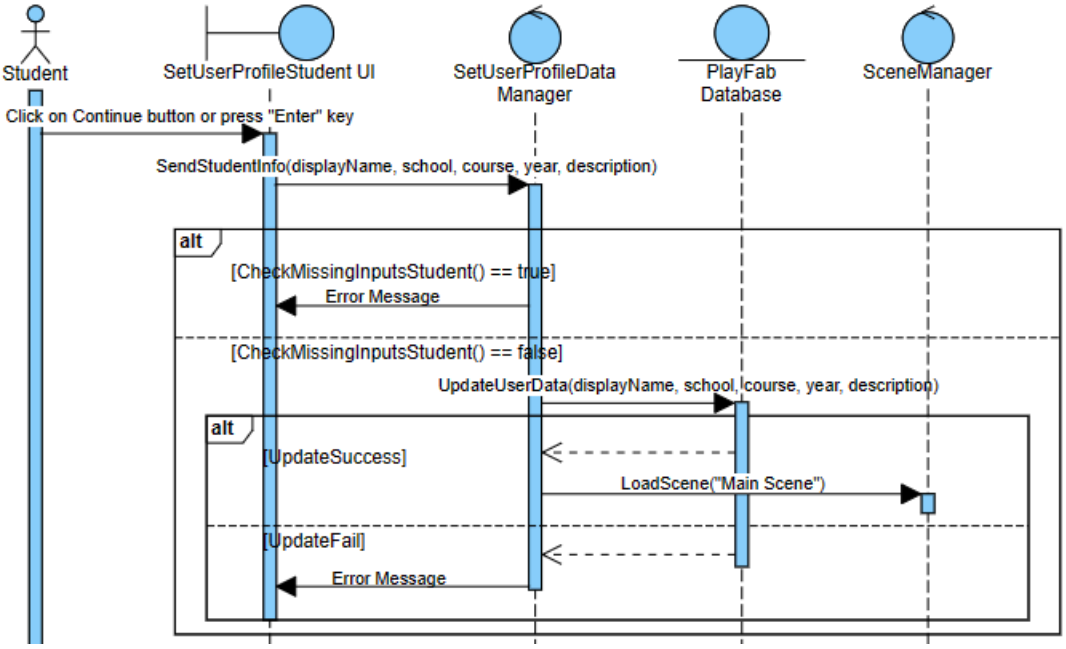

Figure 10: Sequence diagram for updating the student's profile details

4.2 Main Scene

This subsection discusses the design and implementation of the Main Scene. This scene consists of a 3D environment containing 3D objects representing the "Bed", "Wardrobe", "Table", "Door" and "Player" (See Figure 11). This scene will be loaded when login is successful from Login UI (See Section 4.1.1) or when the registration process is completed from the respective SetUserProfile UIs (See Section 4.1.3).

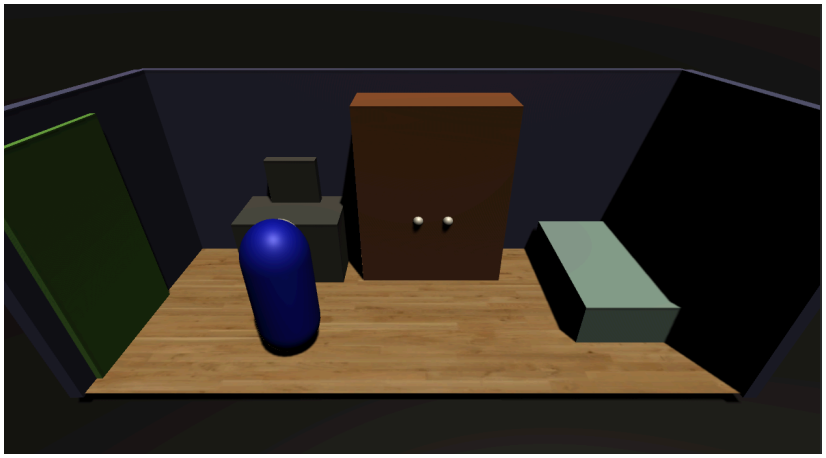

Figure 11: Screenshot of Main Scene in 3rd-person camera

The user will be able to control the “Player” using the “W”, “A”, “S” and “D” keys. When “Player” comes into contact with either the “Bed”, “Wardrobe”, “Table” or “Door”, their respective UIs will be set as active and “Player” will be disabled. “Player” will be enabled once the user closes the respective UIs.

The user will also be able to toggle between a 3rd-person and 1st-person camera by pressing on the “C” key (See Figure 12). In the 1st-person perspective, the user will be able to rotate the camera left and right by pressing on the “Q” and “E” keys respectively.

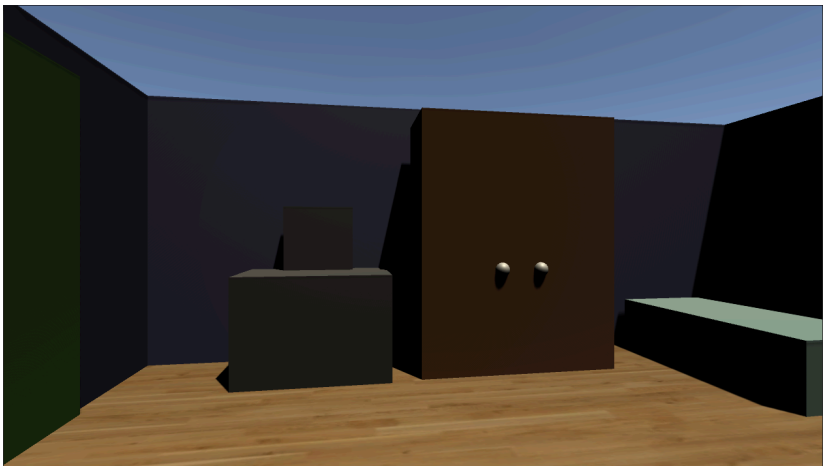

Figure 12: Screenshot of Main Scene in 1st-person camera

To ensure the persistence of the bindings of GameObjects to the scripts in Main Scene, DontDestroyOnLoad(gameObject) was utilised to ensure that the GameObjects do not get destroyed between scenes. We are also checking if the instance of the GameObject is the first instance, else the newly created GameObjects will be destroyed to ensure that only the first instance of the GameObject exists at all times (See Figure 13).

```
/*
 * Purpose: Make sure the gameObject is persistent across scenes
 * Input: NA
 * Output: If instance is null, let instance = this and DontDestroyOnLoad
 *         else destroy the gameObject
 */
Unity Message | 0 references
void Awake()
{
    if (instance == null)
    {
        instance = this;
        DontDestroyOnLoad(gameObject);
    }
    else
    {
        Destroy(gameObject);
    }
}
```

Figure 13: Screenshot of function ensuring persistence of bindings of GameObjects to scripts in Main Scene

4.2.1 LogOut UI

When “Player” comes into contact with the 3D object “Bed”, the LogOut UI will be set as active and “Player” will be disabled. The LogOut UI consists of a “Confirm” button and a “Cancel” button (See Figure 14). Clicking on the “Confirm” button will allow the user to quit the application while clicking on the “Cancel” button will set the LogOut UI as inactive and enable the “Player”.

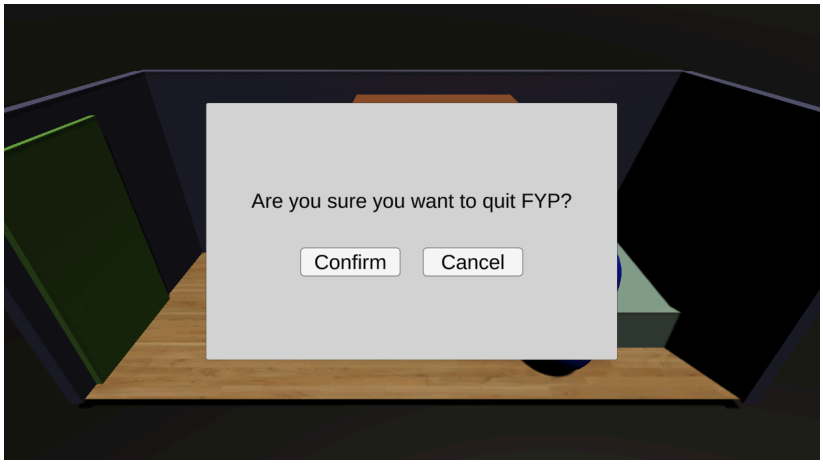

Figure 14: Screenshot of LogOut UI

4.2.2 UserProfile UIs

There are two types of UserProfile UIs, namely the UserProfileStudent UI and UserProfileOthers UI. When "Player" comes into contact with the 3D object "Wardrobe", the respective UserProfile UIs will be set as active and "Player" will be disabled. Similar to the SetUserProfile UIs, if the user is a "Student", the UserProfileStudent UI will be set as active (See Figure 15), and if the user is a "Professor/TA" or "Lab Admin", the UserProfileOthers UI will be set as active instead (See Figure 16).

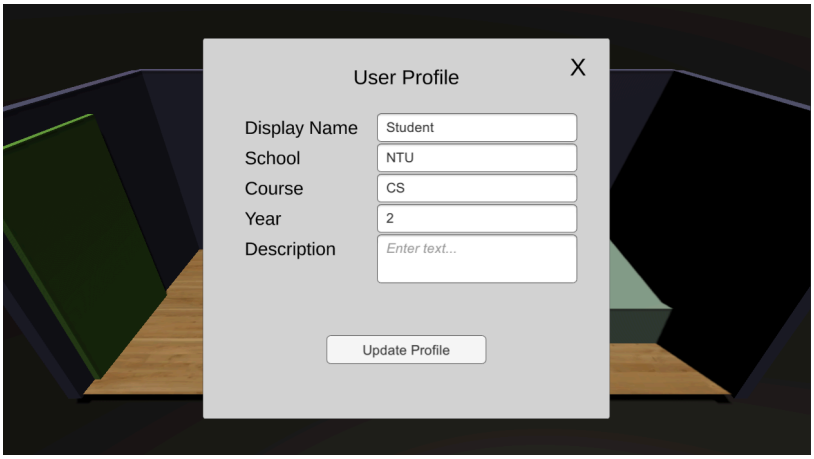

Figure 15: Screenshot of UserProfileStudent UI

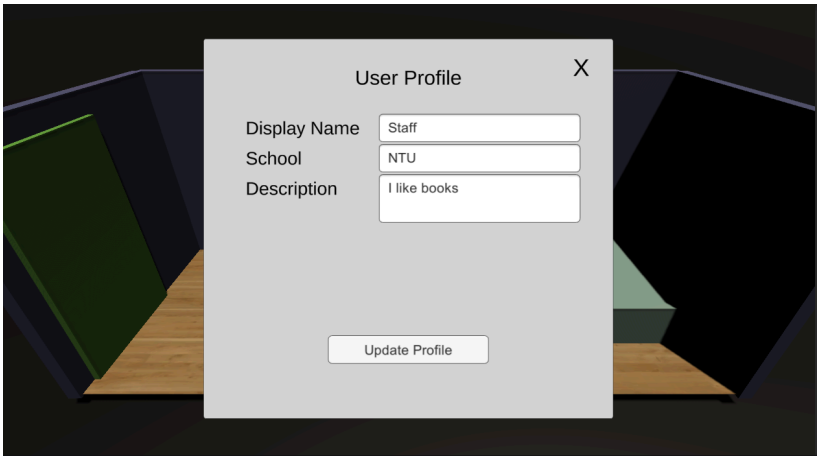

Figure 16: Screenshot of UserProfileOthers UI

As shown in Figure 17 below, when trying to retrieve a user's profile details, we will attempt to retrieve the information using a PlayFab API call. When the retrieval of the profile details is successful, they will be displayed in the UserProfile UIs accordingly.

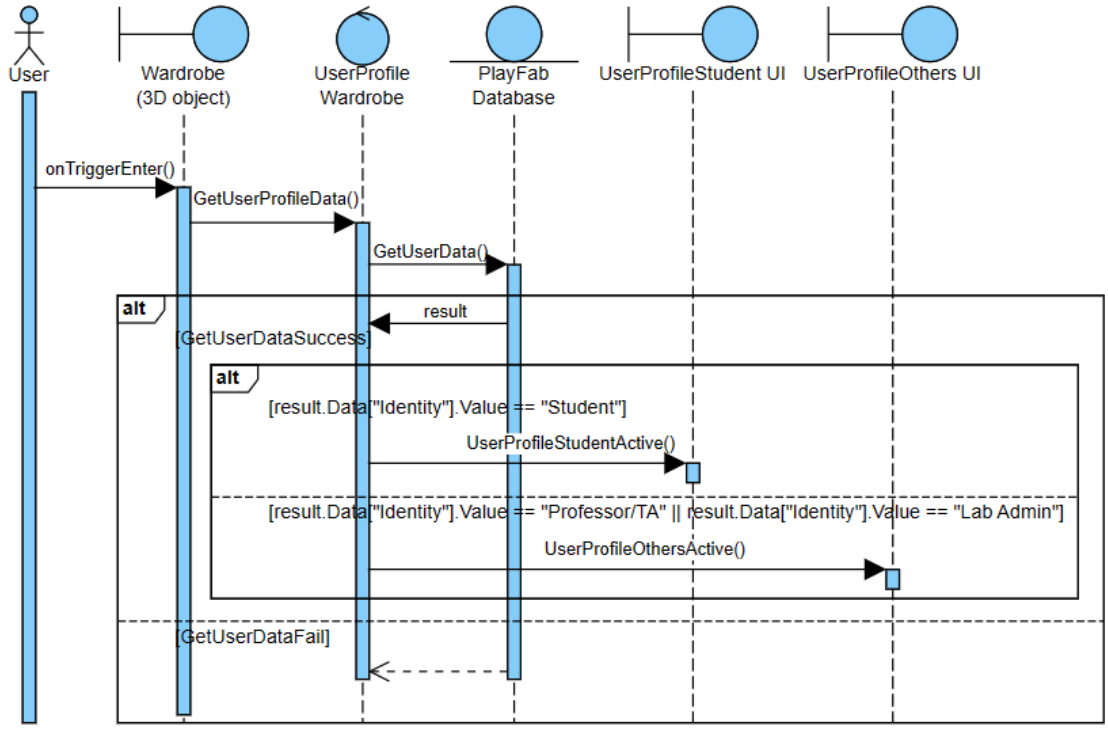

Figure 17: Sequence diagram for displaying the respective UserProfile UIs

One important thing to note here is that “Player” will be disabled only after the information retrieval attempt is successful. This is to ensure that the user will still be able to control “Player” in the case that the attempt to get the user’s profile details has failed.

The UserProfile UIs consist of an “Update Profile” button (See Figures 15 & Figure 16) that allows users to update their profile details. As shown in Figure 18 below, when a user tries to update their profile details, we will attempt to update the user’s data using a PlayFab API call. When the user’s profile details are updated successfully, there will be a success message shown in the respective UserProfile UIs to inform the user of the successful update. Otherwise, an error message will be shown to inform the user of the error.

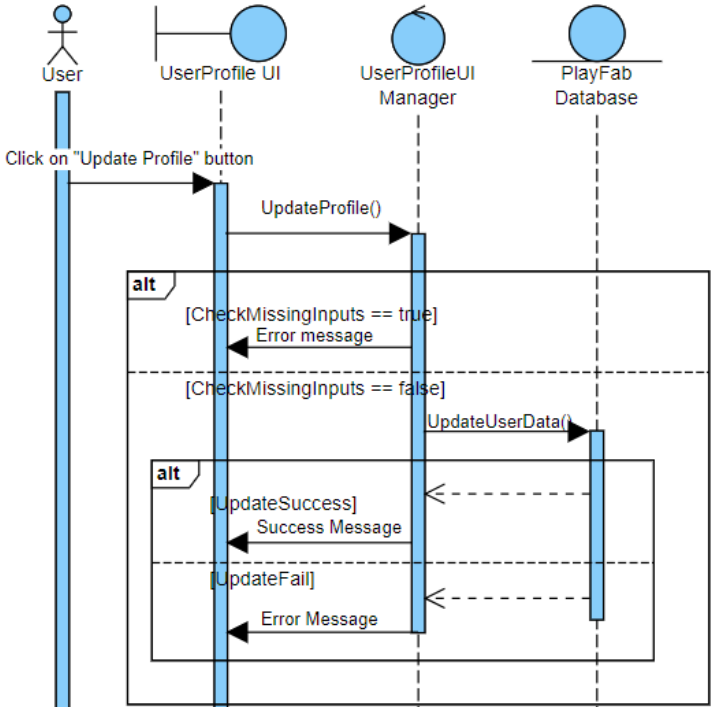

Figure 18: Sequence diagram for updating the user profile details

Ultimately, we can choose to close the UserProfile UIs by clicking on the “X” button in the top right-hand corner (See Figures 15 & Figure 16). Clicking on the “X” button will set the respective UserProfile UIs as inactive and enable the “Player”.

4.2.3 MainMenu UIs

There are three types of MainMenu UIs, namely the MainMenuStudent UI, MainMenuProf UI and MainMenuStaff UI. When “Player” comes into contact with the 3D object “Table”, the respective MainMenu UIs will be set as active and “Player” will be disabled. If the user is a “Student”, the MainMenuStudent UI will be set as active (See Figure 19), and if the user is a “Professor/TA” or “Lab Admin”, the MainMenuProf UI or MainMenuStaff UI will be set as active respectively (See Figure 20).

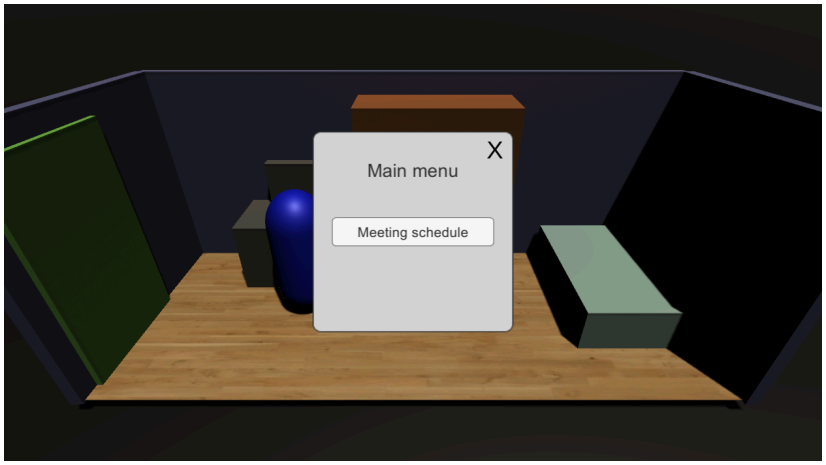

Figure 19: Screenshot of MainMenuStudent UI

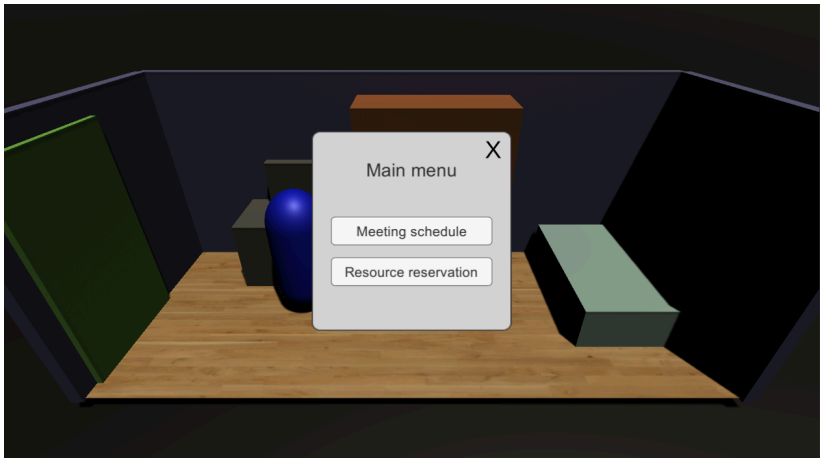

Figure 20: Screenshot of MainMenuProf or MainMenuStaff UI

All three types of MainMenu UIs consist of a “Meeting Schedule” button that will set the respective MainMenu UIs as inactive and the MeetingSchedule UI as active when clicked. On the other hand, only the MainMenuProf UI and MainMenuStaff UI contain a “Resource Reservation” button, which will set the respective MainMenu UIs as inactive and the respective ResourceReservation UIs as active when clicked.

Ultimately, we can choose to close the MainMenu UIs by clicking on the “X” button in the top right-hand corner (See Figures 19 & Figure 20). Clicking on the “X” button will set the respective MainMenu UIs as inactive and enable the “Player”.

4.2.4 MeetingSchedule UI

By clicking on the “Meeting Schedule” button in the MainMenu UIs (See Figures 19 & 20), the MeetingSchedule UI will be set as active. The MeetingSchedule UI is a calendar that consists of clickable date buttons. Users can also toggle between the different months by clicking on the left and right arrows located at the top of the UI (See Figure 21).

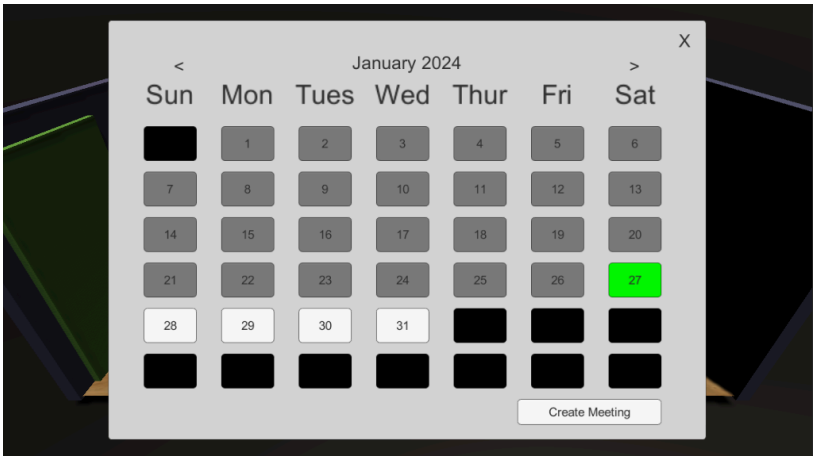

Figure 21: Screenshot of the MeetingSchedule UI

A listener has been added to each of the clickable dates in the MeetingSchedule UI. Clicking on a date will set the MeetingSchedule UI as inactive and the MeetingDetails UI as active. As shown in Figure 22 below, when a user clicks on a date button in the MeetingSchedule UI, we will first set the MeetingDetails UI as active. Then, we will retrieve all the meetings scheduled for the user on that day from MongoDB Atlas. Once the list of scheduled meetings is retrieved, the dynamic button creator will create the meeting buttons in the MeetingDetails UI.

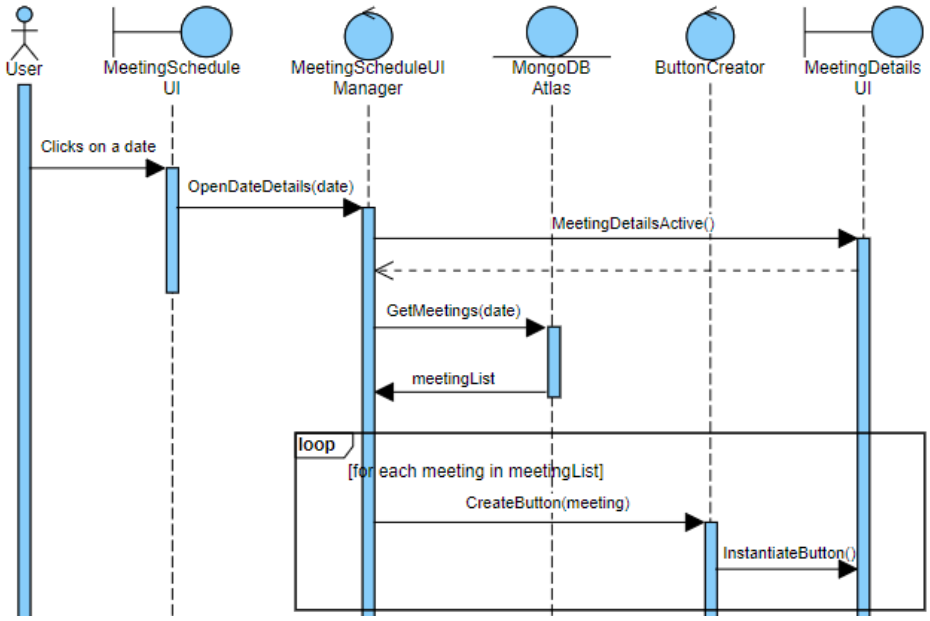

Figure 22: Sequence diagram for opening MeetingDetails UI

The MeetingSchedule UI also consists of a “Create Meeting” button in the bottom right-hand corner (See Figure 21). Clicking on the “Create Meeting” button will set the MeetingSchedule UI as inactive and the NewMeeting UI as active.

Ultimately, we can choose to close the MeetingSchedule UI by clicking on the “X” button in the top right-hand corner (See Figure 21). Clicking on the “X” button will set the MeetingSchedule UI as inactive and then the respective MainMenu UIs as active according to the user’s identity.

4.2.5 MeetingDetails UI

By clicking on a date button in the MeetingSchedule UI (See Figure 21), the MeetingDetails UI will be set as active. The MeetingDetails UI consists of a list of clickable meeting buttons created by the dynamic button creator (See Figure 22). Users can use the scrollbar to scroll through the whole list of scheduled meetings. If a meeting was created by the users themselves, the meeting button will be blue, whereas if a meeting was created by other users, the meeting button will be white. The MeetingDetails UI also consists of a “Start Meeting” and a “Delete Meeting” button, which are initially disabled (See Figure 23).

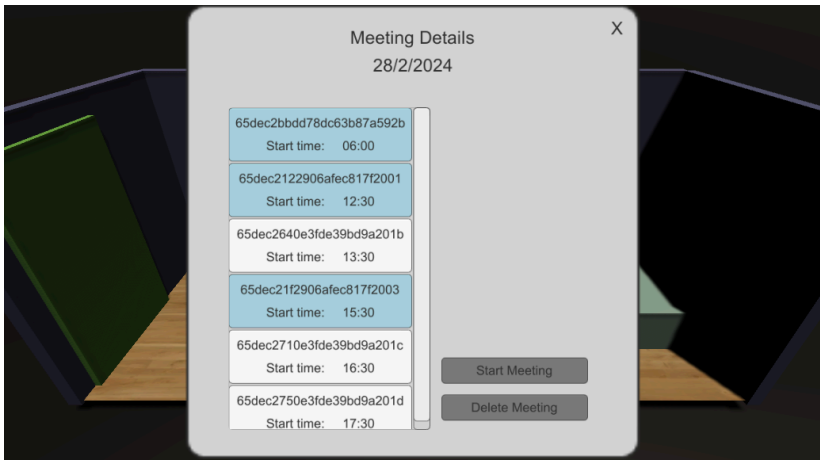

Figure 23: Screenshot of MeetingDetails UI

When the user clicks on a white meeting button, the details of the meeting will be shown on the right-hand side of the MeetingDetails UI. Details such as the MeetingID, HostEmail, Duration, Description, JoinCode and Participants will be shown. Since the meeting was created by another user, the “Start Meeting” and “Delete Meeting” buttons will be disabled (See Figure 24).

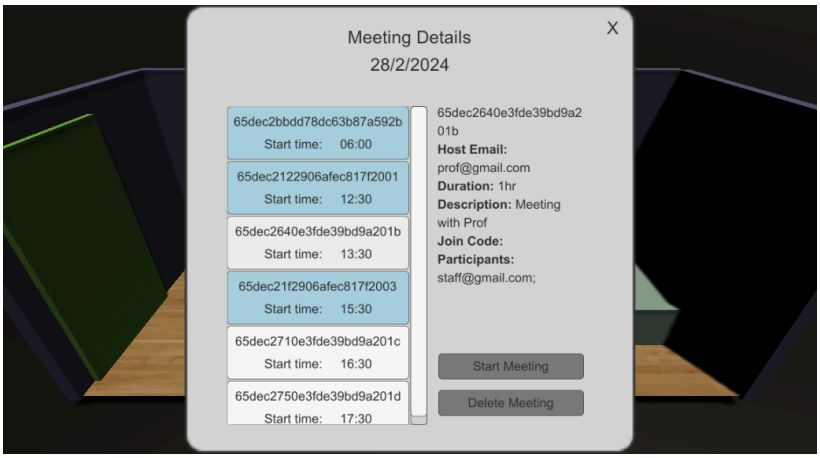

Figure 24: Screenshot of MeetingDetails UI when a white meeting button is clicked

Similarly, when the user clicks on a blue meeting button, the details of the meeting will be shown on the right-hand side of the MeetingDetails UI. However, since the meeting was created by the user, the “Start Meeting” and “Delete Meeting” buttons will be enabled (See Figure 25).

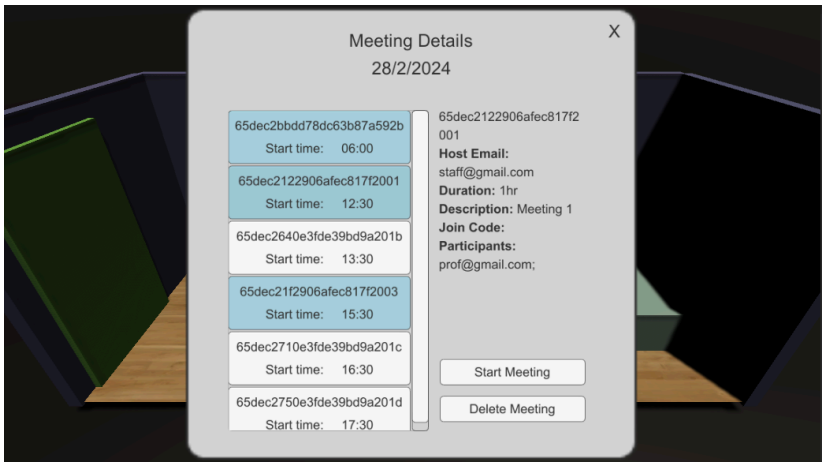

Figure 25: Screenshot of MeetingDetails UI when a blue meeting button is clicked

As shown in Figure 26 below, when a user clicks on the “Start Meeting” button, we will attempt to update the MeetingID and MeetingStatus using a PlayFab API call. When the information is updated successfully, the ClassRoom Scene will be loaded.

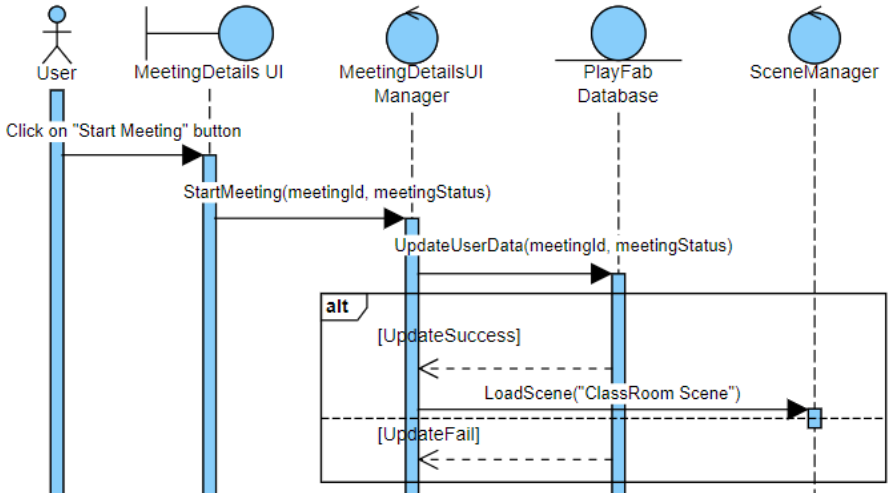

Figure 26: Sequence diagram for starting a scheduled meeting

As shown in Figure 27 below, when the user clicks on the “Delete Meeting” button, we will remove the scheduled meeting selected from MongoDB Atlas. Once the scheduled meeting is removed successfully, the MeetingDetails UI will refresh to show the updated list of scheduled meetings.

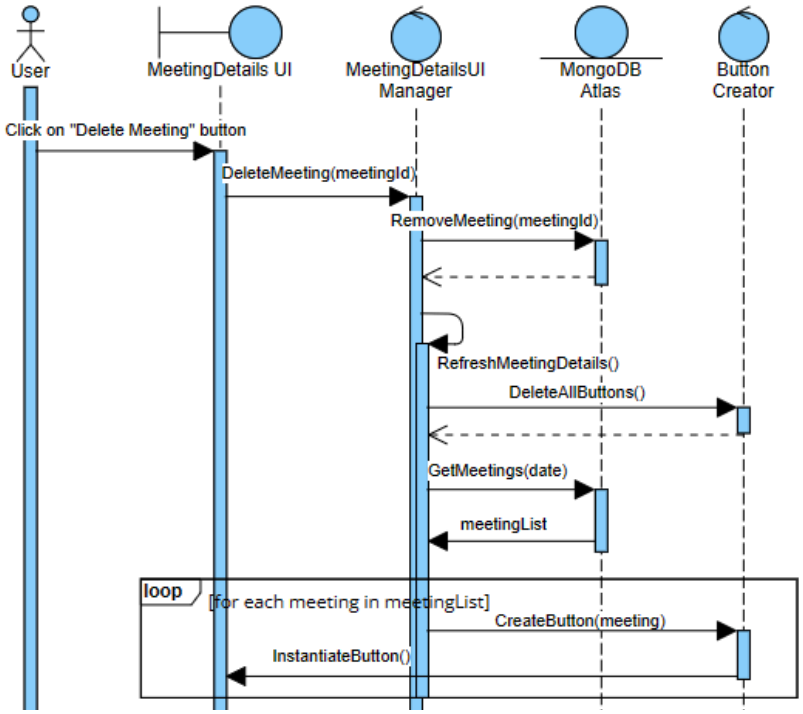

Figure 27: Sequence diagram for deleting a scheduled meeting

Ultimately, we can choose to close the MeetingDetails UI by clicking on the "X" button in the top right-hand corner (See Figure 23). Clicking on the "X" button will set the MeetingDetails UI as inactive and the MeetingSchedule UI as active.

4.2.6 NewMeeting UI

By clicking on the "Create Meeting" button in the MeetingSchedule UI (See Figure 21), the NewMeeting UI will be set as active. The NewMeeting UI consists of a "Create Meeting" button (See Figure 28) that allows users to create a new meeting by providing required information such as the Date, Start time, Duration and Description. Users can choose to create a new meeting without participants by leaving Participant as empty.

Figure 28: Screenshot of NewMeeting UI

As shown in Figure 29 below, when the user clicks on the "Create Meeting" button, we will attempt to add the meeting to MongoDB Atlas. Once the scheduled meeting is added successfully, there will be a success message shown in the NewMeeting UI to inform the user of the successful update. Otherwise, an error message will be shown.

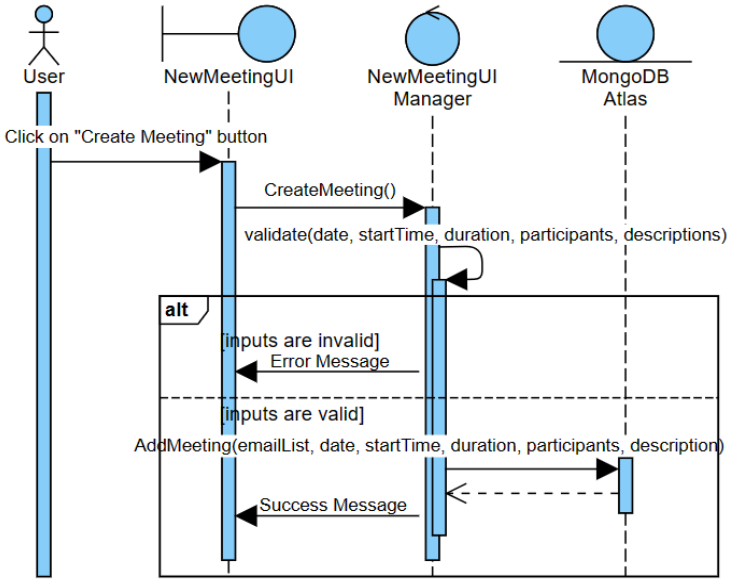

Figure 29: Sequence diagram for adding a new meeting

The schema of data stored in the Meetings collection in MongoDB Atlas is seen in Figure 30 below.


```
_id: ObjectId('65d5dc6870321c39fce85c3e')
_partition: "FYP"
date: "21/2/2024"
description: "12"
duration: "12"
host_email: "staff@gmail.com"
join_code: ""
▾ participant_emails: Array (1)
  ▾ 0: Object
      participant_email: "prof@gmail.com"
start_time_hr: 0
start_time_min: 12
```


Figure 30: Meeting details stored in MongoDB Atlas (Meetings)

To add participants to the meeting, the user will have to input the participants' emails and click on the "Add" button (See Figure 28). As shown in Figure 31 below, when the user clicks on the "Add" button, we attempt to search for the participant using a PlayFab API call. If the participant is a registered user, their email will be added to the list of participants if it has not already been added. Otherwise, an error message will be shown to inform the user of the error.

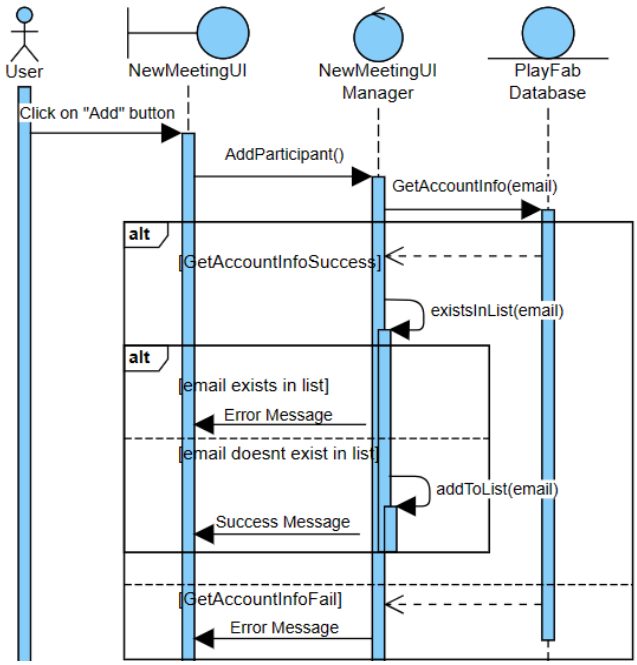

Figure 31: Sequence diagram for adding a new participant through email

Ultimately, we can choose to close the NewMeeting UI by clicking on the “X” button in the top right-hand corner (See Figure 28). Clicking on the “X” button will set the NewMeeting UI as inactive and set the MeetingSchedule UI as active.

4.2.7 ResourceReservation UIs

There are two types of ResourceReservation UIs, namely the ResourceReservationProf UI and ResourceReservationStaff UI. By clicking on the “Resource Reservation” button in the respective MainMenu UIs (See Figure 20), the respective ResourceReservation UIs will be set as active. The ResourceReservationProf UI and the ResourceReservationStaff UI are calendars similar to that of the MeetingSchedule UI, where users can click into dates and toggle between the different months (See Figure 32 & 33).

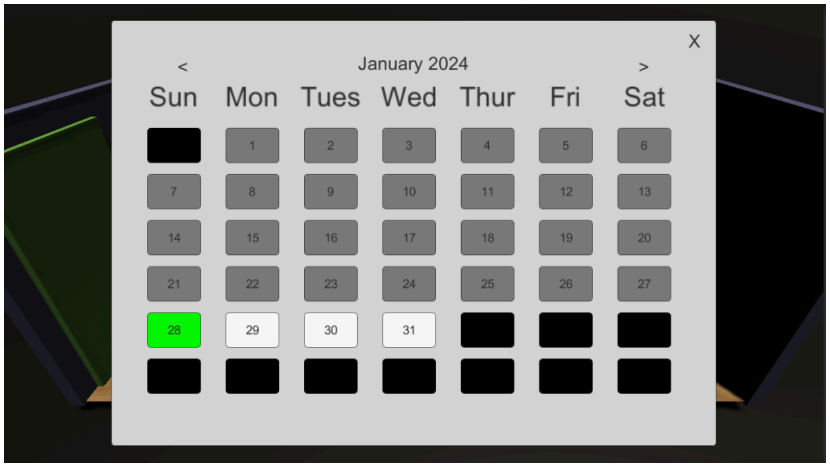

Figure 32: Screenshot of ResourceReservationProf UI

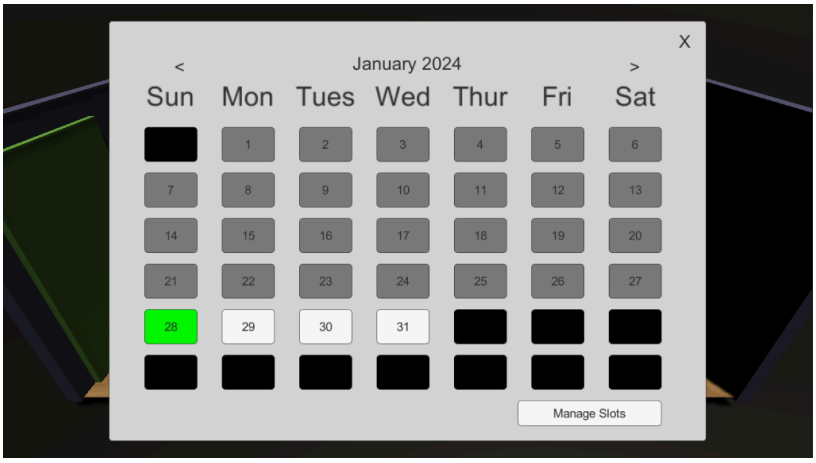

Figure 33: Screenshot of ResourceReservationStaff UI

Similar to the MeetingSchedule UI, a listener has been added to each of the clickable dates in both the ResourceReservationProf UI and ResourceReservationStaff UI. Clicking on a date will set the respective ResourceReservation UIs as inactive and the respective DateDetails UIs as active. As seen in Figure 34 below, when a user clicks on a date button in the respective ResourceReservation UIs, we will first set the respective DateDetails UIs as active. Then, we will retrieve all the locations with available time slots from MongoDB Atlas. Once the list of locations is retrieved, the dynamic button creator will create the location buttons in the respective DateDetail UIs.

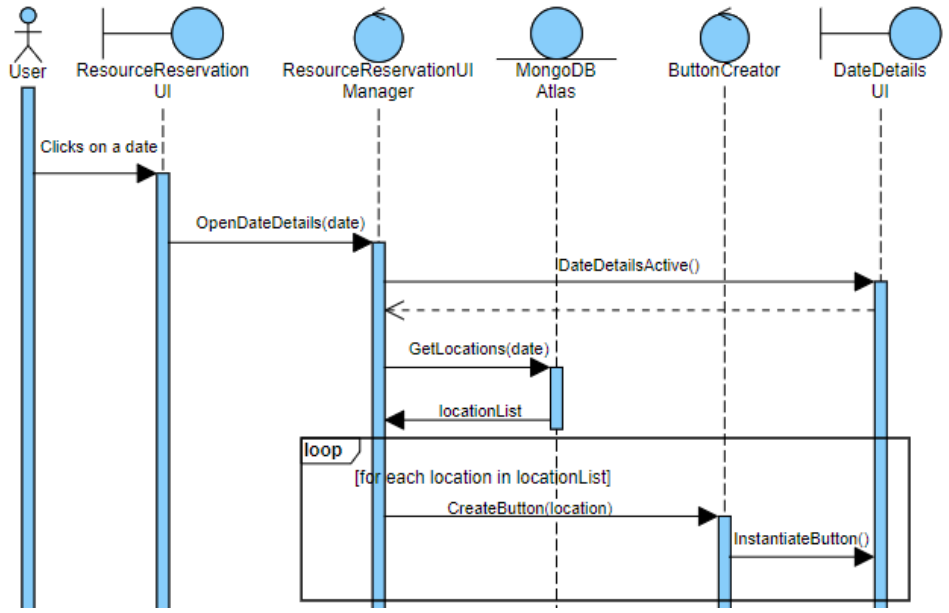

Figure 34: Sequence diagram for opening DateDetails UI

The ResourceReservationStaff UI also consists of a "Manage Slots" button in the bottom right-hand corner (See Figure 33). Clicking on the "Manage Slots" button will set the ResourceReservationStaff UI as inactive and the ManageSlots UI as active.

Ultimately, we can choose to close the ResourceReservation UIs by clicking on the "X" buttons in the top right-hand corner (See Figures 32 & 33). Clicking on the "X" buttons will set the respective ResourceReservation UIs as inactive and then the respective MainMenu UI as active according to the user's identity.

4.2.8 ManageSlots UI

By clicking on the "Manage Slots" button in the ResourceReservationStaff UI (See Figure 33), the ManageSlots UI will be set as active. The ManageSlots UI consists of a "Remove slots" button and an "Add slots" button (See Figure 35) that allows users to remove and add available

time slots by providing required information such as the Location, StartDate, EndDate, StartTime and EndTime.

Figure 35: Screenshot of the ManageSlots UI

As shown in Figure 36 below, when the user clicks on the “Remove slots” button, we will attempt to remove the specified available time slots from MongoDB Atlas. Once the slots have been removed successfully, there will be a success message shown in the ManageSlots UI to inform the user of the successful update. Otherwise, an error message will be shown. It is important to note that only available time slots that have not already been reserved will be removed from MongoDB Atlas. Clicking on the “Add slots” button will result in a similar sequence as when the “Remove slots” button is clicked, where AddAvailableSlots() and AddAvailable() are called instead of RemoveAvailableSlots() and RemoveAvailable(). Duplicate time slots for the same location will not be created when trying to add new slots.

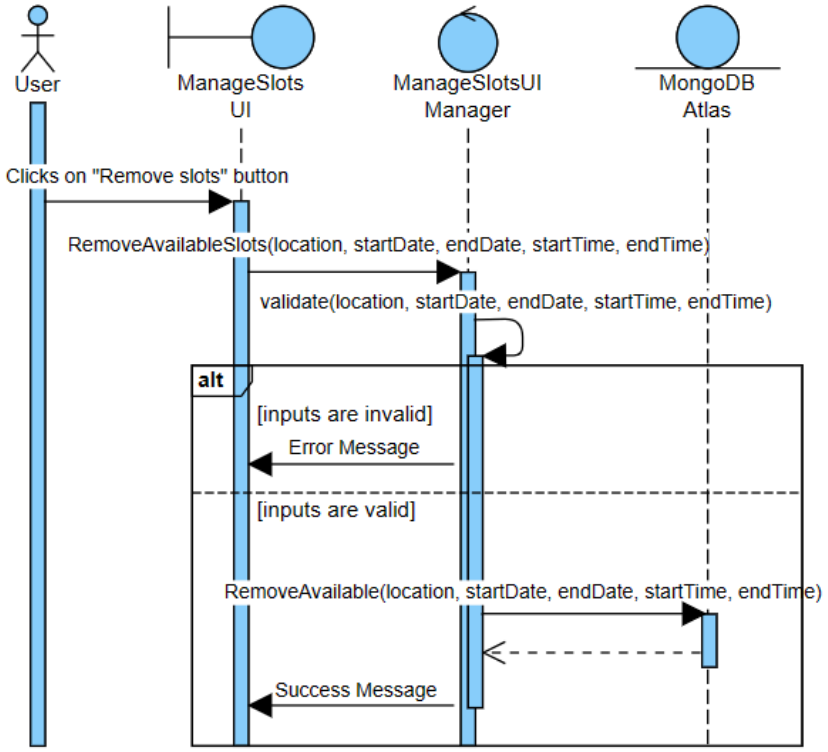

Figure 36: Sequence diagram for removing an existing time slot

The schema of data stored in the Available collection in MongoDB Atlas is seen in Figure 37 below.

```
_id: ObjectId('658180f23147cc4b3c8ed9ab')
_partition: "FYP"
date: 12
hour: 1
location: "lab1"
min: 0
month: 12
year: 23
```

Figure 37: Reservation slot stored in MongoDB Atlas (Available)

Ultimately, we can choose to close the ManageSlots UI by clicking on the “X” button in the top right-hand corner (See Figure 35). Clicking on the “X” buttons will set the ManageSlots UI as inactive and the ResourceReservationStaff UI as active.

4.2.9 DateDetails UIs

There are two types of DateDetails UIs, namely the DateDetailsProf UI and the DateDetailsStaff UI. By clicking on a date button in the respective ResourceReservation UIs (See Figures 32 & 33), the respective DateDetails UIs will be set as active. The DateDetails UIs consist of a list of clickable location buttons created by the dynamic button creator (See Figure 34). If there is at least one time slot made available for the chosen date, a location button will be created (See Figure 38).

Figure 38: Screenshot of the DateDetailsProf and DateDetailsStaff UI

A listener has been added to each of the location buttons in the DateDetails UIs. Clicking on a location will set the respective DateDetails UIs as inactive and the respective TimeDetails UI as active. As seen in Figure 39 below, when a user clicks on a location button in the DateDetails UI, we will first set the respective TimeDetails as active. Then, we will retrieve all time slots made available with the selected location and date from MongoDB Atlas. Once the list of timings is retrieved, the dynamic button creator will create the reservation buttons in TimeDetails UI.

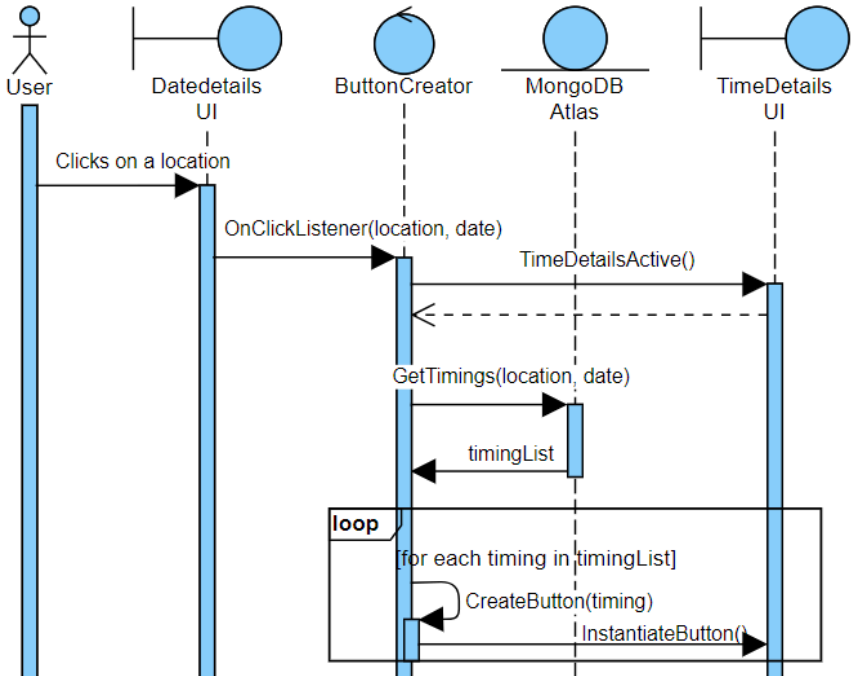

Figure 39: Sequence diagram for opening TimeDetails UI

Ultimately, we can choose to close the DateDetails UI by clicking on the "X" button in the top right-hand corner (See Figure 38). Clicking on the "X" button will set the respective DateDetails UIs as inactive and the respective ResourceReservation UIs as active.

4.2.10 TimeDetails UIs

There are two types of TimeDetails UI, namely the TimeDetailsProf UI and the TimeDetailsStaff UI. By clicking on a location button in the respective DateDetails UIs (See Figure 38), the respective TimeDetails UI will be set as active. The TimeDetailsProf UI contains a list of clickable time slot buttons created by the dynamic button creator (See Figure 39). Users can use the scrollbar to scroll through the whole list of time slots. If a time slot is still available for reservation, the time slot button will be in white. However, if a time slot has already been reserved by the users themselves, the time slot button will be in blue, and if a time slot was

reserved by other users, the time slot button will be in grey. The TimeDetailsProf UI also consists of an “Add Reservation” and a “Remove Reservation” button (See Figure 40).

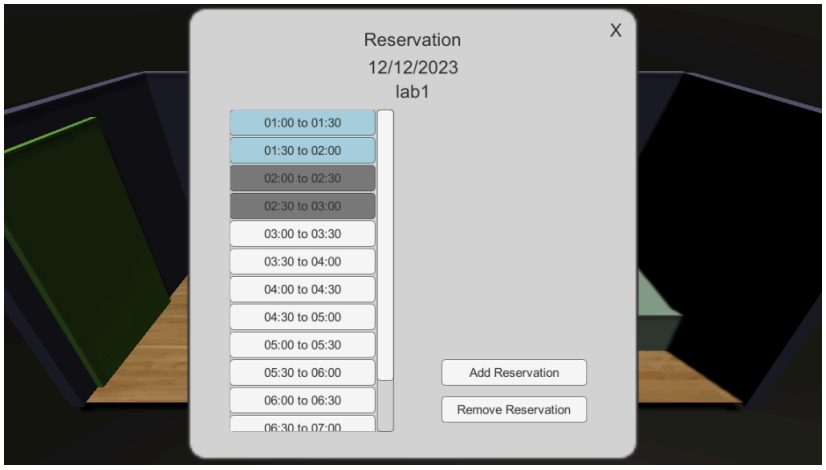

Figure 40: Screenshot of TimeDetailsProf UI

When a user tries to reserve more time slots by clicking on a white time slot button, the slot selected will turn from white to green (See Figure 41) and will be added to a list of time slots for adding reservations.

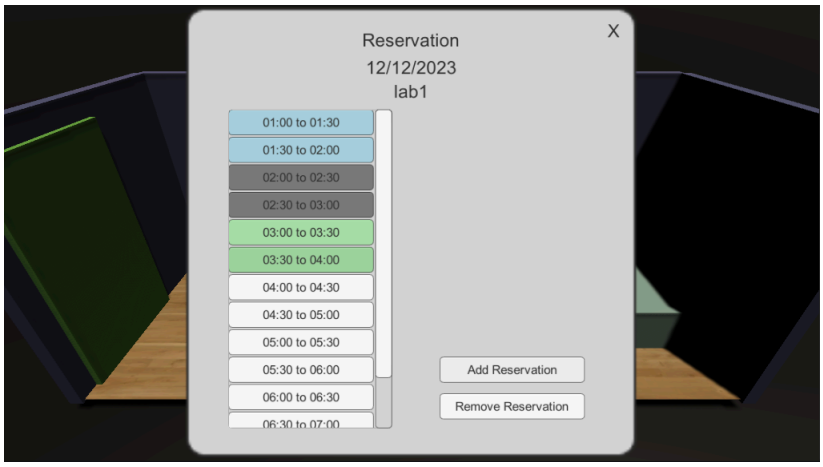

Figure 41: Screenshot of TimeDetailsProf UI when trying to reserve more slots

As shown in Figure 42 below, when the user clicks on the “Add Reservation” button, we first retrieve all the time slots to be reserved from the list of time slots for adding reservations and

then we will add the selected time slots to MongoDB Atlas. Once the time slots are added successfully, the list of time slots for adding reservations and for removing reservations will be cleared and TimeDetails UI will refresh to show the updated list of time slots.

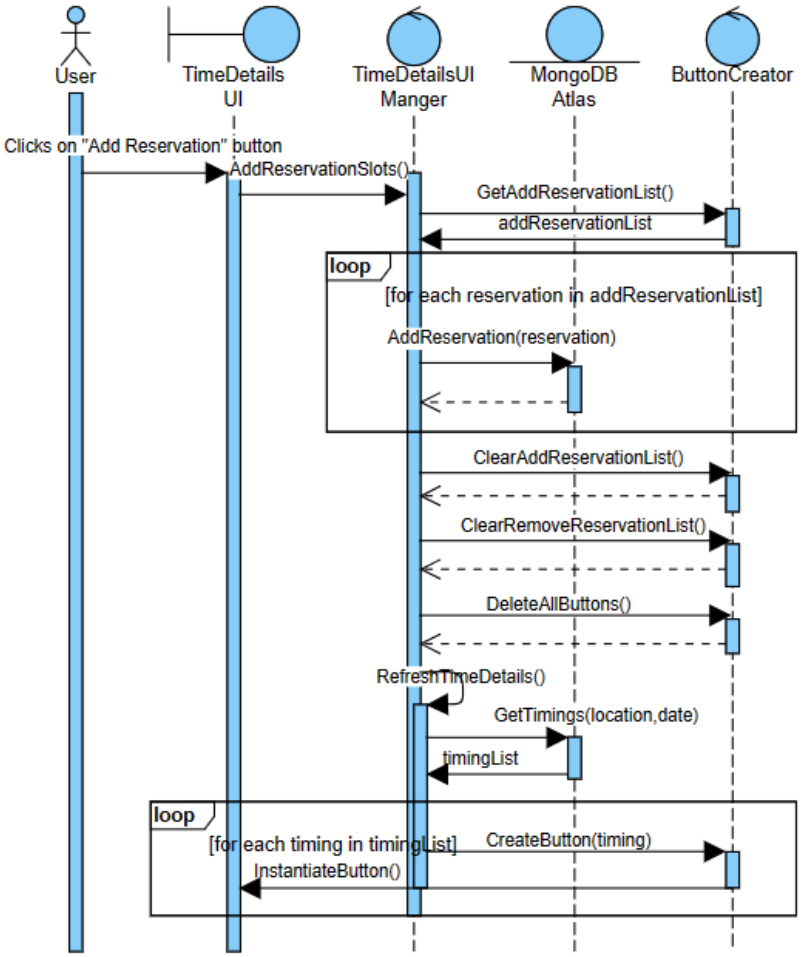

Figure 42: Sequence diagram for reserving available time slots

The schema of the data stored in the Reserved collection in MongoDB Atlas is seen in Figure 43 below.

```
_id: ObjectId('658181124449343b154589f5')
_partition: "FYP"
date: 12
hour: 1
location: "lab1"
min: 30
month: 12
name: "prof@gmail.com"
year: 23
```


Figure 43: Reserved reservation slot stored in MongoDB Atlas (Reserved)

On the other hand, when a user tries to remove a reserved time slot by clicking on a blue time slot button, the slot selected will turn from blue to red (See Figure 44) and will be added to a list of time slots for removing reservations.

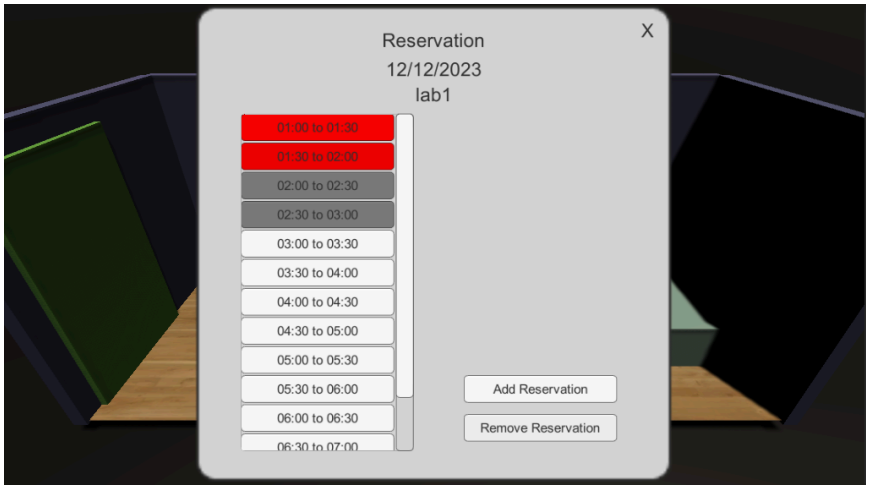

Figure 44: Screenshot of TimeDetailsProf UI when trying to remove reserved slots

As shown in Figure 45 below, when the user clicks on the "Remove Reservation" button, similar to when the user clicks on the "Add Reservation" button, we first retrieve all the time slots to be unreserved from the list of time slots for removing reservations and then we will remove the selected reserved time slots from MongoDB Atlas. Once the reserved time slots are removed

successfully, the list of time slots for adding reservations and for removing reservations will be cleared and TimeDetails UI will refresh to show the updated list of time slots.

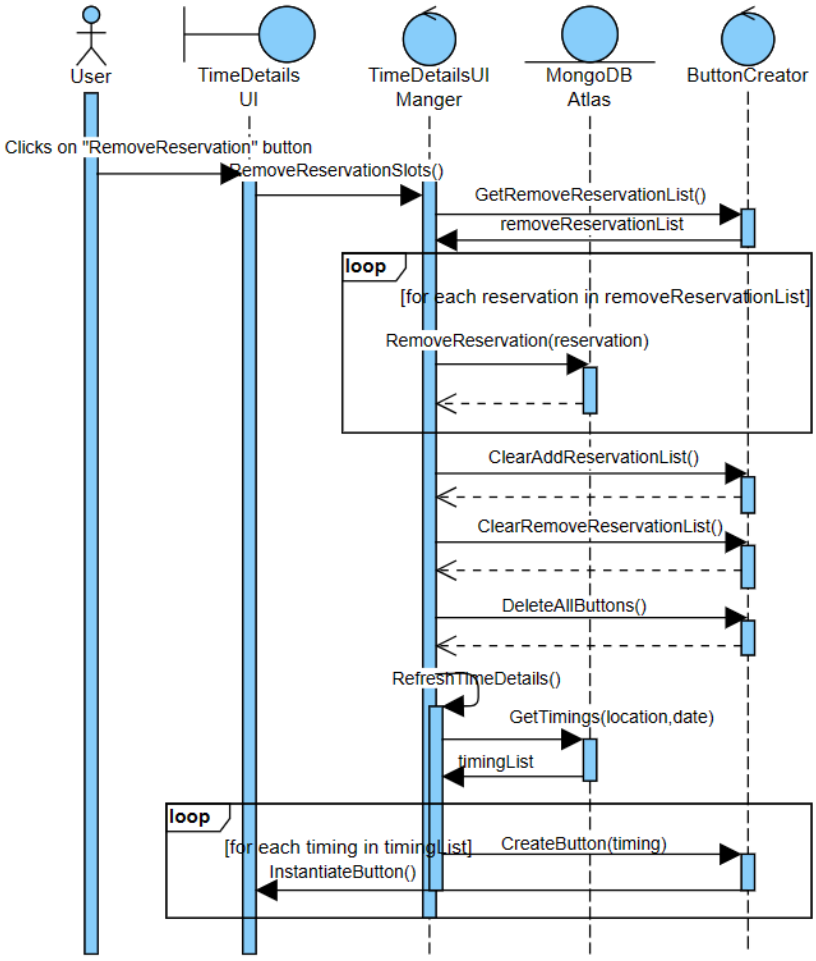

Figure 45: Sequence diagram for cancelling reserved time slots

Similar to the TimeDetailsProf UI, the TimeDetailsStaff UI also contains a list of clickable reservation time slot buttons created by the dynamic button creator (See Figure 39). If a reservation time slot is still available for reservation, the time slot button will be in white. However, if a time slot has already been reserved by Professors/TAs, the time slot button will be in grey (See Figure 46).

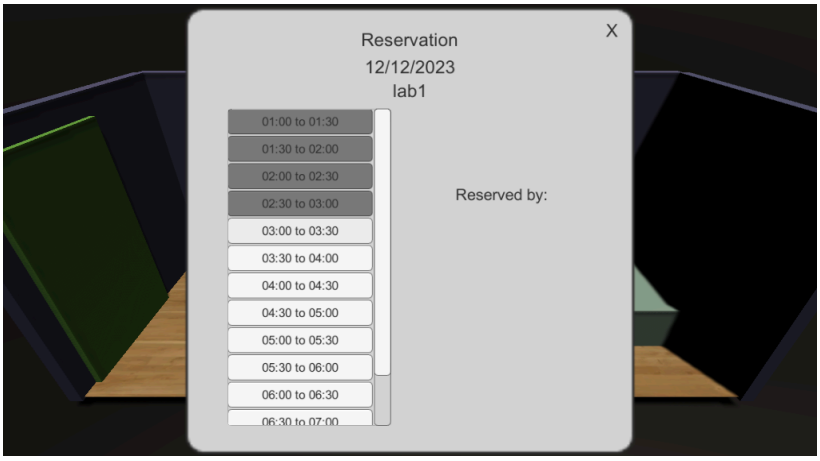

Figure 46: Screenshot of TimeDetailsStaff UI

When the user clicks on a grey time slot button, the details of the reservation will be shown on the right-hand side of the TimeDetailsStaff UI. Details such as the selected time slot and the email of the user who reserved the time slot will be shown (See Figure 47).

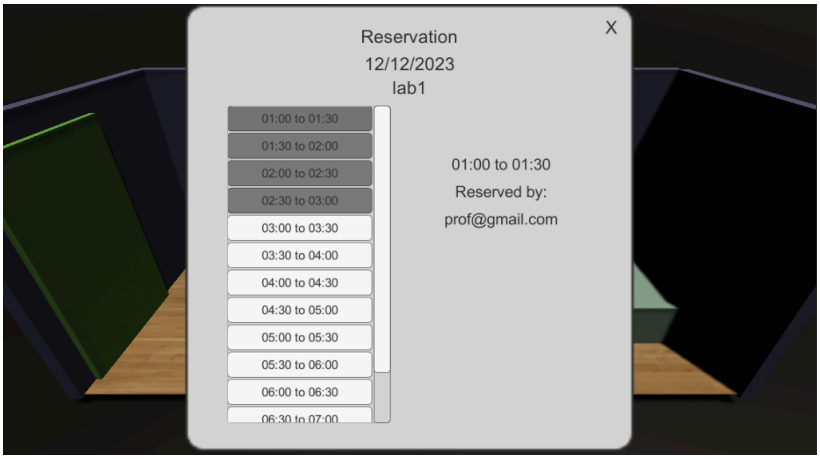

Figure 47: Screenshot of TimeDetailsStaff UI when a grey button is clicked

Ultimately, we can choose to close the respective TimeDetails UI by clicking on the “X” button in the top right-hand corner (See Figures 40 & 46). Clicking on the “X” button will set the respective TimeDetails UI as inactive and the respective DateDetails UI as active.

4.2.11 JoinMeeting UI

When “Player” comes into contact with the 3D object “Door”, the JoinMeeting UI will be set as active and “Player” will be disabled. The JoinMeeting UI consists of a “Join” button that will allow users to join an ongoing meeting when a JoinCode is entered (See Figure 48).

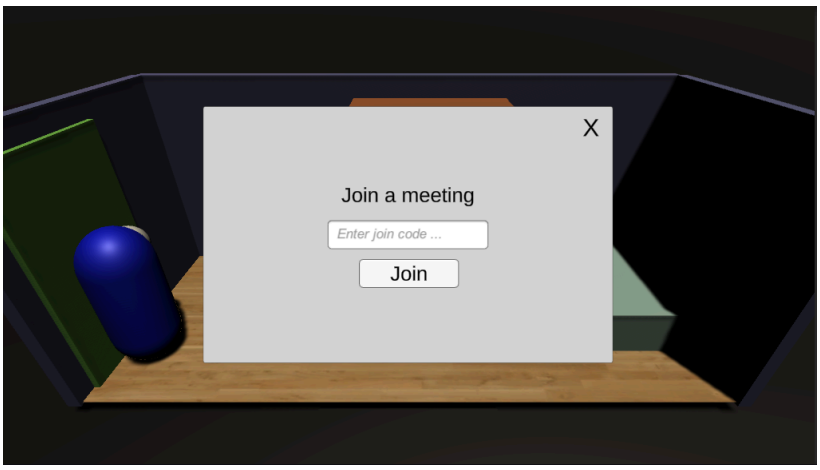

Figure 48: Screenshot of JoinMeeting UI

As shown in Figure 49 below, when a user clicks on the “Join” button, we will attempt to update the JoinCode and MeetingStatus using a PlayFab API call. When the information is updated successfully, the ClassRoom Scene will be loaded.

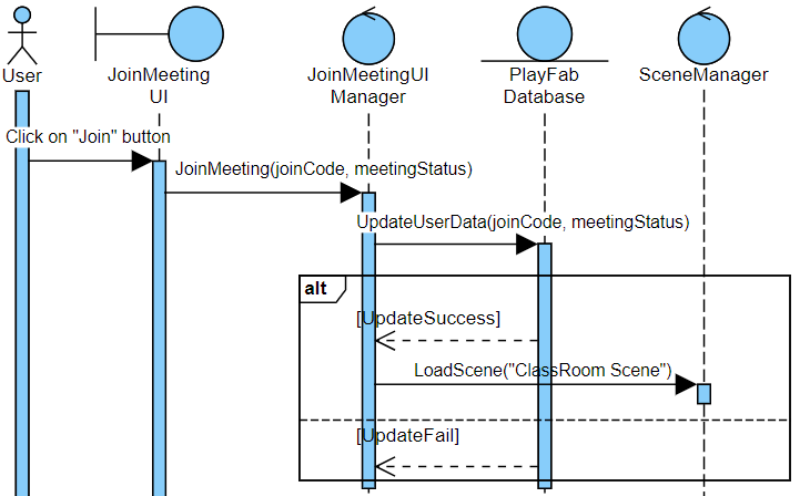

Figure 49: Sequence diagram for joining a scheduled meeting

Ultimately, we can choose to close the JoinMeeting UI by clicking on the “X” button in the top right-hand corner (See Figure 48). Clicking on the “X” button will set the JoinMeeting UI as inactive and enable the “Player”.

4.3 ClassRoom Scene

This subsection discusses the design and implementation of the ClassRoom Scene. This scene consists of a 3D environment containing 3D objects representing the “Stage” (See Figure 50) and a 2D Canvas containing the NetworkManager, EnableQuiz, SampleQuiz, EnableClass and JoinCodeError UIs. This scene will be loaded when the host of a meeting starts the session from MeetingDetails UI (See Section 4.2.5) or when the participant of a meeting joins the session from JoinMeeting UI (See Section 4.2.11). When the ClassRoom Scene is loaded, only the NetworkManager UI will be set as active.

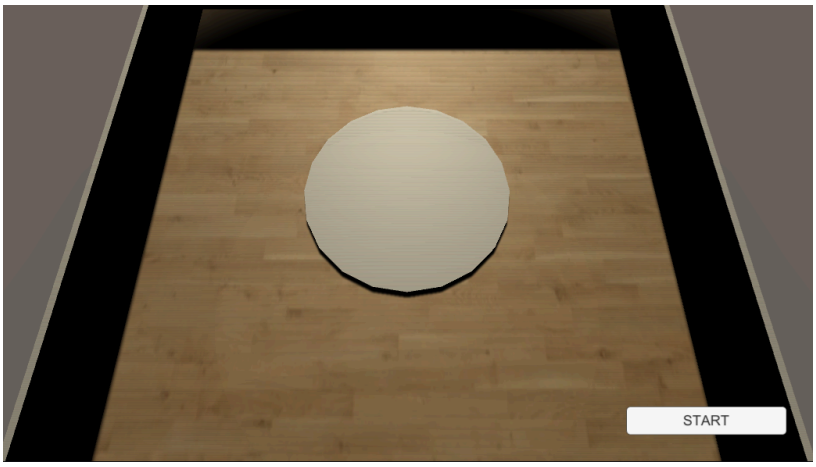

Figure 50: Screenshot of ClassRoom Scene with NetworkManager UI

To ensure that the user has access to the multiplayer functions in this scene, we will first initialize all Unity Gaming Services at once, followed by an anonymous sign-in to the session and initialization of the voice chat service (See Figure 51) when the ClassRoom Scene is loaded.

```
await UnityServices.InitializeAsync();

try
{
    await AuthenticationService.Instance.SignInAnonymouslyAsync();
    Debug.Log("Relay Start AuthenticationService SignInAnonAsync");
}
catch (AuthenticationException ex)
{
    Debug.Log(ex);
}
catch (RequestFailedException ex)
{
    Debug.Log(ex);
}

try
{
    await VivoxService.Instance.InitializeAsync();
    Debug.Log("Relay Start VivoxService InitializeAsync");
}
catch (Exception ex)
{
    Debug.Log(ex);
}
```

Figure 51: Screenshot of function ensuring access to multiplayer functions in ClassRoom Scene

The “NetworkManager” GameObject in the ClassRoom Scene has components “Network Manager” and “Unity Transport” that are required for the multiplayer functions to work in the ClassRoom Scene. The “Network Manager” component is a Netcode component that contains all Netcode related settings and the “Unity Transport” component is a low-level networking library used for Netcode.

To automatically spawn a “Player” GameObject that belongs to a user in the session, we created a “Player” prefab and placed it into the “Network Manager” component of the “Network Manager” GameObject. This “Player” prefab has components “Network Object” and “Client Network Transform” that allow synchronisation of the “Player” GameObject’s position across the network. The “Player” prefab is also placed into the NetworkPrefabsList used by NetworkManagers to pick up any changes made.

Similar to the “Player” GameObject in the Main Scene, the user will be able to control the “Player” using the “W”, “A”, “S” and “D” keys and will be able to rotate the camera left and right by pressing on the “Q” and “E” keys respectively in the 1st-person perspective.

4.3.1 NetworkManager UI

The NetworkManager UI consists of a “Start” button (See Figure 50) that allows users to enter the meeting room. When the user clicks on the “Start” button and the user is the host of the meeting, the user will start the NetworkManager as a host, spawn a “Player” GameObject and then the “Enable Audio”, “Quizzes”, “Classes” and “End” buttons will be set as active. The JoinCode generated for this session will also be shown at the top right-hand corner (See Figure 52).

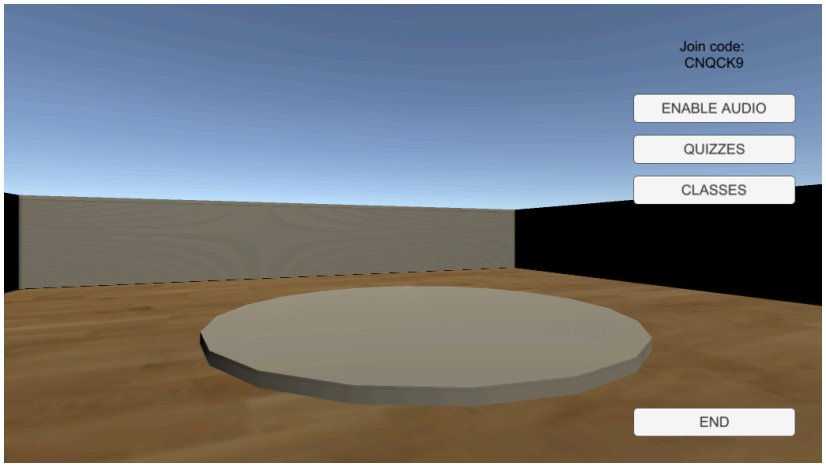

Figure 52: Screenshot of NetworkManager UI when the “Start” button is clicked as a host

On the other hand, when the user clicks on the “Start” button and the user is a participant of the meeting, the user will start the NetworkManager as a client, spawn a “Player” GameObject and then only the “Enable Audio” and “End” button will be set as active (See Figure 53).

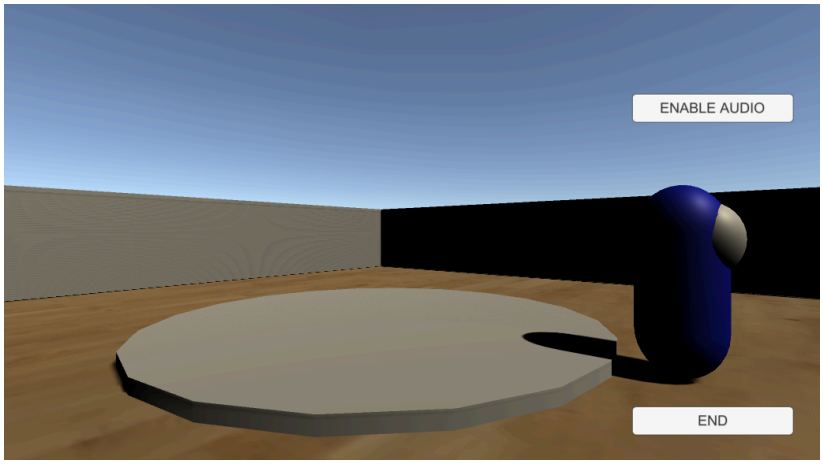

Figure 53: Screenshot of NetworkManager UI when the "Start" button is clicked as a client

As shown in Figure 54 below, when a user clicks on the "Start" button in the NetworkManager UI, we will first attempt to get the user's data using a PlayFab API call. When the retrieval of the data is successful, we will check for the MeetingStatus of the user.

If the user is the host of the meeting, we will start a new meeting session by calling StartHost(). Then, we will update the JoinCode generated to MongoDB Atlas. This JoinCode can be seen in the MeetingDetails UI when a button is clicked (See Figures 24 & 25) and is used by participants of the meeting in JoinMeeting UI (See Figure 48). Then we will store the email of the host to MongoDB Atlas, set the respective buttons as active and then join the voice chat channel of the meeting session.

If the user is a participant of the meeting, we will join the existing meeting session by calling StartClient(). Then, we will get the MeetingID using the JoinCode, store the email of the participant to MongoDB Atlas, set the respective buttons as active and then join the voice chat channel of the meeting session.

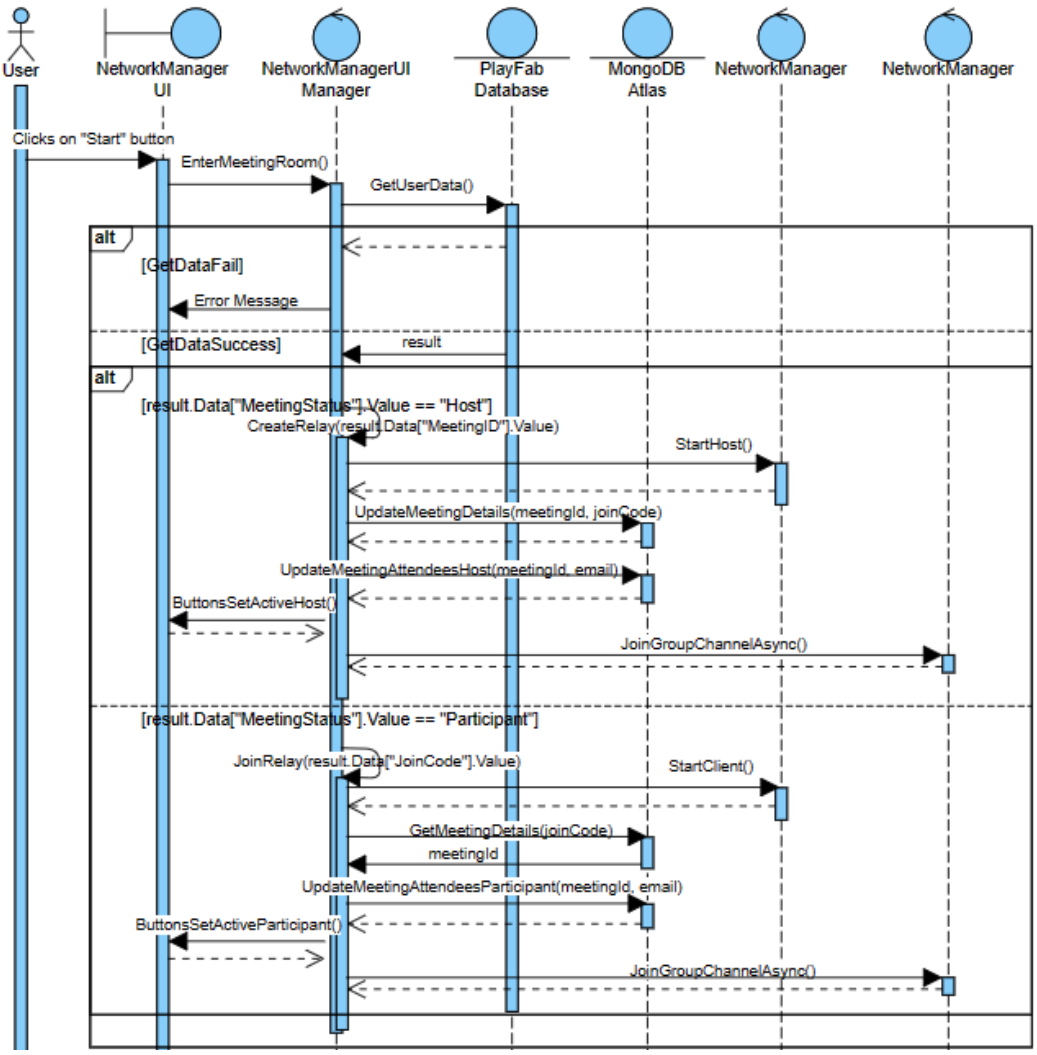

Figure 54: Sequence diagram for starting or joining a meeting session

The schema of data stored in the Meeting Attendees collection in MongoDB Atlas is seen in Figure 55 below. This information is collected for personalization and performance improvement of the application.


```
_id: ObjectId('65a6309d5315a3ba6ef6f774')
_partition: "FYP"
host_email: "staff@gmail.com"
meetingId: "65a48cb1c006bf12265c470a"
▾ participant_emails: Array (1)
  ▾ 0: Object
      participant_email: "prof@gmail.com"
```


Figure 55: Meeting Attendees stored in MongoDB Atlas (Meeting Attendees)

For both the host and the client, clicking on the “Enable Audio” button will unmute the user and then set the “Disable Audio” button as active (See Figure 56). Similarly, clicking on the “Disable Audio” will mute the user and then set the “Enable Audio” button as active.

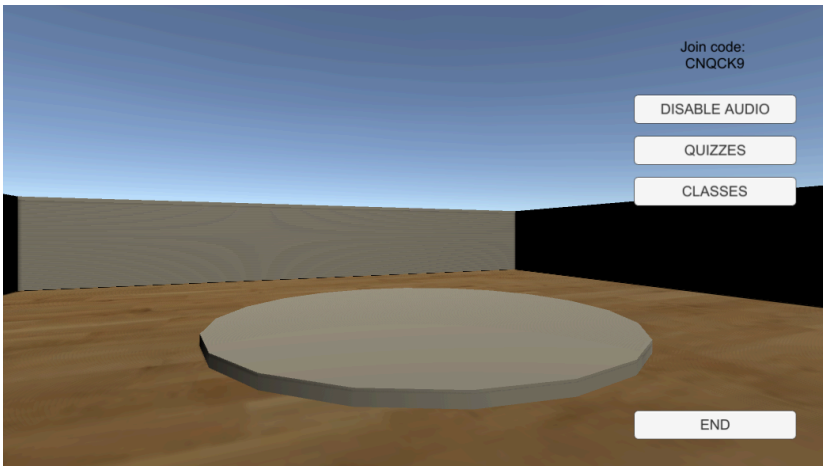

Figure 56: Screenshot of NetworkManager UI when the “Enable Audio” button is clicked

For the host, when the user clicks on the “Quizzes” button, the EnableQuiz UI will be set as active, and when the user clicks on the “Classes” button, the EnableClass UI will be set as active. When the user clicks on the “End” button and the user is the host of the meeting, the user will leave the meeting session, shut down the NetworkManager and destroy it. Then, we will update the JoinCode to “Meeting Ended” and load the Main Scene. On the other hand, when the user clicks on the “End” button and the user is a participant of the meeting, the user will only leave the meeting session and destroy the NetworkManager. Then, we will load the Main Scene.

4.3.2 EnableQuiz UI

By clicking on the “Quizzes” button in the NetworkManager UI (See Figure 52), the EnableQuiz UI will be set as active. The EnableQuiz UI consists of an “Enable Sample” button (See Figure 57) that allows the host to enable the sample quiz for all users in the meeting session.

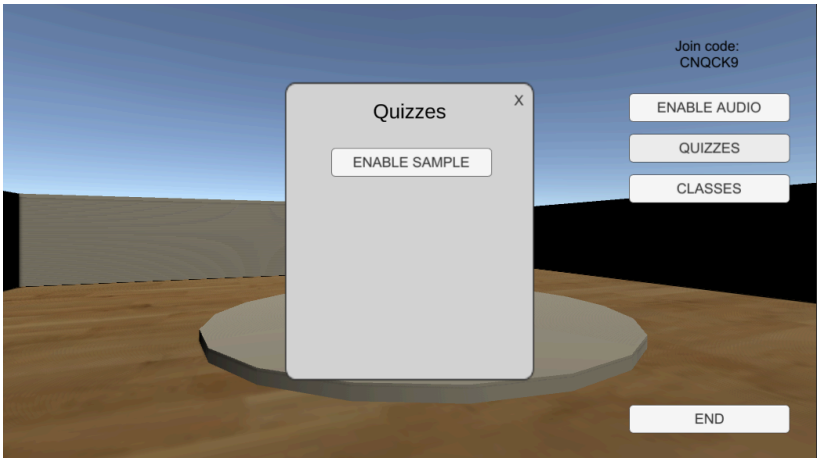

Figure 57: Screenshot of EnableQuiz UI of the host when the "Quizzes" button is clicked

When the host clicks on the "Enable Sample" button, the host will invoke a client RPC which will set the "Join Quiz" button as active at the top left-hand corner on all clients (See Figure 58), including the host. The "Disable Sample" button in EnableQuiz UI will also be set as active on the host side (See Figure 59). Similarly, when the host clicks on the "Disable Sample" button, the host will invoke a client RPC which will set the "Join Quiz" button as inactive on all clients, including the host. The "Enable Sample" button in EnableQuiz UI will then be set as active on the host side.

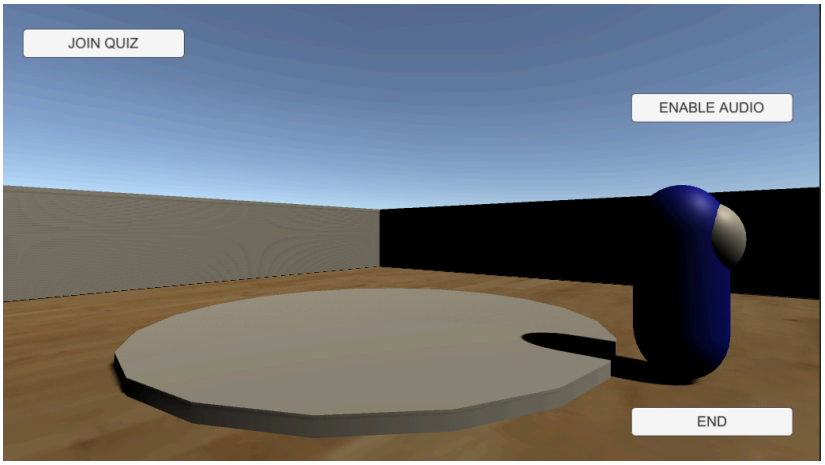

Figure 58: Screenshot of the "Join Quiz" button on the client

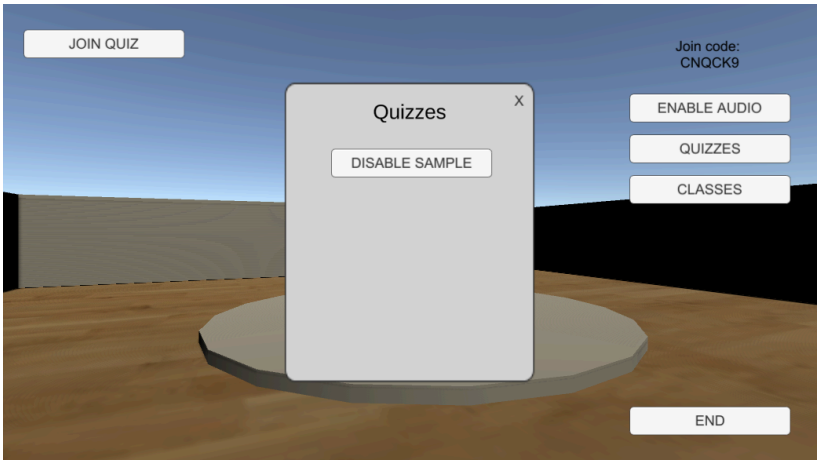

Figure 59: Screenshot of the "Join Quiz" button and EnableQuiz UI on the host

When the user clicks on the "Join Quiz" button, the SampleQuiz UI will be as active on the user's side only, regardless of whether they are the client or the host.

Ultimately, the host can choose to close the EnableQuiz UI by clicking on the "X" button in the top right-hand corner (See Figure 57). Clicking on the "X" button will set the EnableQuiz UI as inactive.

4.3.3 SampleQuiz UI

By clicking on the "Join Quiz" button (See Figures 58 & 59), the SampleQuiz UI will be set as active. The SampleQuiz UI consists of sample questions that can be toggled by clicking on the left and right arrows located at the bottom right-hand side of the SampleQuiz UI (See Figure 60).

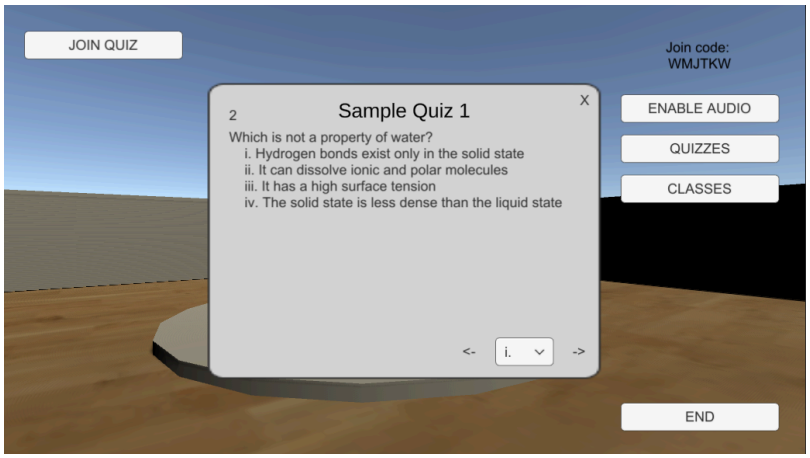

Figure 60: Screenshot of the SampleQuiz UI

Ultimately, the user can choose to close the SampleQuiz UI by clicking on the "X" button in the top right-hand corner (See Figure 60). Clicking on the "X" button will set the SampleQuiz UI as inactive.

4.3.4 EnableClass UI

By clicking on the "Classes" button in the NetworkManager UI (See Figure 52), the EnableClass UI will be set as active. The EnableClass UI consists of an "Enable Sample" button (See Figure 61) that allows the host to enable the sample class for all users in the meeting session.

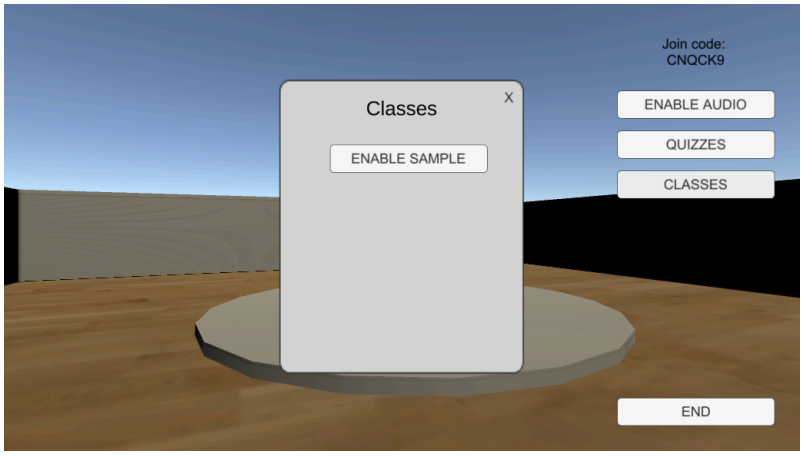

Figure 61: Screenshot of EnableClass UI of the host when the "Classes" button is clicked

When the host clicks on the “Enable Sample” button, the host will first instantiate the “ClassSample” prefab as a GameObject and then spawn the “ClassSample” GameObject on all clients (See Figure 62), including the host. The “Disable Sample” button in EnableClass UI will also be set as active on the host side (See Figure 63). Similarly, when the host clicks on the “Disable Sample” button, the host will destroy the “ClassSample” GameObject on all clients, including the host. The “Enable Sample” button in EnableClass UI will then be set as active on the host side.

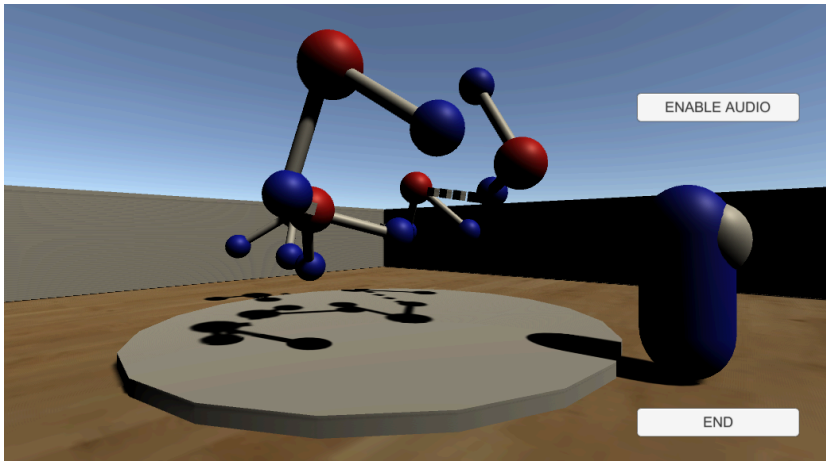

Figure 62: Screenshot of the “ClassSample” GameObject on the client

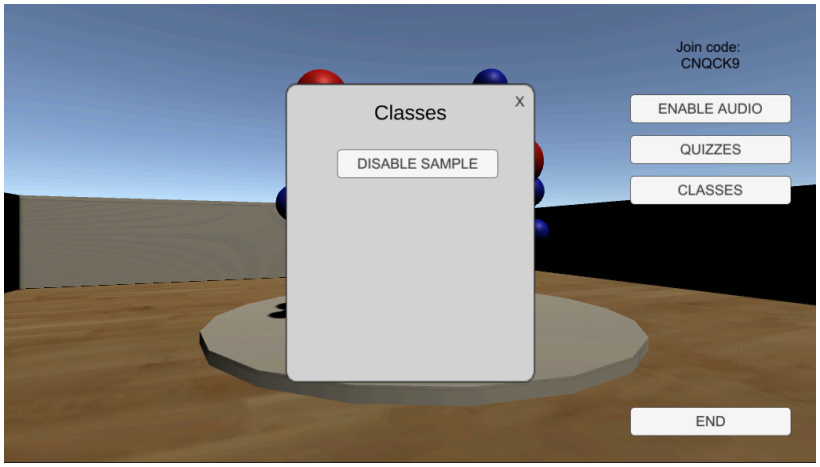

Figure 63: Screenshot of the “ClassSample” GameObject and the EnableClass UI on the host

Ultimately, the host can choose to close the EnableClass UI by clicking on the “X” button in the top right-hand corner (See Figure 61). Clicking on the “X” button will set the EnableClass UI as inactive.

4.3.5 JoinCodeError UI

When the user clicks on the “Start” button in NetworkManager UI (See Figure 50) and the JoinCode is not valid, the JoinCodeError UI will be set as active. The JoinCodeError UI consists of a “Return” button that allows the user to return to the Main Scene (See Figure 64).

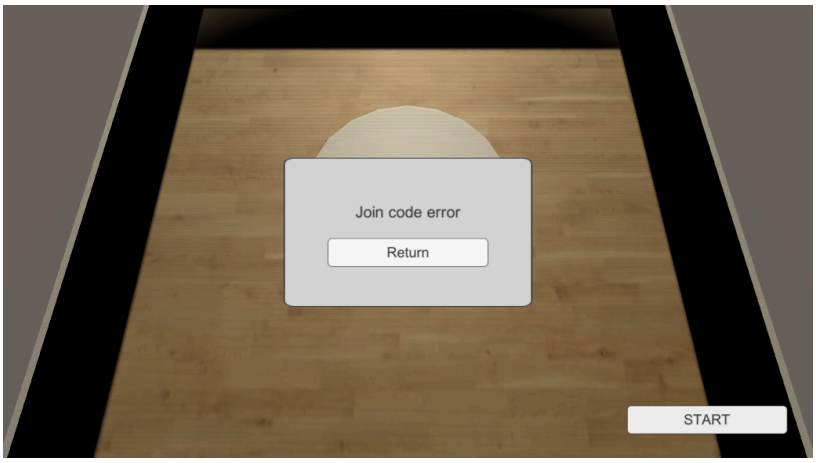

Figure 64: Screenshot of the JoinCodeError UI

5 Evaluation

This section contains the evaluations of the Metaverse prototype by considering the possible limitations of the project.

5.1 Possible Limitations

This subsection contains the possible limitations of the project. By identifying its limitations, we will be able to find out how to better improve the application. As seen in Table 4 below, we have listed possible limitations that our application has and have also provided possible solutions for each limitation.

Table 4: Possible limitations and solutions

	Possible Limitations	**Possible Solutions**
1	The current plan of the PlayFab account is the “Free To Start” plan option, where there is an upper limit of 100,000 unique players. This means that there can only be a maximum of 100,000 registered accounts for the application.	Explore other third-party services that offer a similar functionality and/ or combine multiple third-party services.
2	The MongoDB Atlas App Services is currently free to use below the following monthly free tier thresholds: 1,000,000 requests or 500 hours of compute or 10,000 hours of sync runtime (whichever occurs first) and 10GB of data transfer. If it exceeds any monthly free threshold, it will start billing for additional usage of any kind.	Explore other third-party services that offer a similar functionality and/ or combine multiple third-party services.
3	The Unity Gaming Services Multiplayer, which is used during meeting sessions, is currently free to use. The Relay, used for peer-to-peer, listen-server UDP communications between players, has a limit of approximately 2,160,000 connectivity minutes per month, and exceeding the limit would result in	Explore other third-party services that offer a similar functionality and/ or combine multiple third-party services.

	payment required. The Vivox Voice Chat, used for the audio chat, has a limit of up to 5000 PCU.	
4	The Metaverse application was tested using only two laptops. This means that there could be performance and scalability issues when there is a large number of concurrent users, leading to system failures or performance bottlenecks.	Test the application with more devices through alpha and beta testing to ensure that the application can handle a large number of users concurrently.
5	The Metaverse application was only tested in Microsoft Windows 11 Home. This could mean that there could be compatibility issues where the application may not be compatible with other operating systems or devices, limiting the users' access to the application.	Test the application with more devices through alpha and beta testing to ensure that the application is compatible with other operating systems or devices.
6	The Metaverse application does not support the importing or creation of the quizzes and 3D classes in a meeting session by a user.	Explore ways that allow users to import or create quizzes and 3D classes to ensure that users are able to customize the meeting sessions according to their requirements.
7	The Metaverse application does not support the customization of the "Player" GameObject by a user.	Explore ways that allow users to customize the "Player" GameObject according to their preferences.

6 Conclusion

As mentioned earlier, this project attempts to develop a Metaverse prototype using Unity to provide users with functions such as the scheduling and reservation of VR equipment available in an educational institute. Furthermore, we attempt to collect data efficiently on the usage of virtual meeting rooms for personalization and performance improvement and data such as the emails of the participants in a meeting will be collected and stored in a database.

The objective of this project was met where we were able to provide users with functions such as, adding and removing available time slots in ManageSlots UI in the Main Scene for users with a "Lab Admin" account, and reserving an available and cancelling a reserved time slot in TimeDetailsProf UI in the Main Scene for users with a "Professor/TA" account, for the scheduling and reservation of the VR equipment available in an educational institute. In addition, we were also able to collect data efficiently on the usage of virtual meeting rooms by storing the emails of the participants of a meeting session when a user successfully enters the session in MongoDB Atlas.

With these functions provided through the Metaverse prototype, educational institutes will now be able to manage better the use of the limited quantity of VR equipment, contributing to a smoother adoption of immersive Metaverse-based education in educational institutes.

One significant limitation of the Metaverse application was that due to the third-party services used for the development of this application, there is an upper limit of 100,000 registered accounts for PlayFab, an upper limit of 1,000,000 requests or 500 hours of compute or 10,000

hours of sync runtime (whichever occurs first) and 10GB of data transfer for MongoDB Atlas, an upper limit of approximately 2,160,000 connectivity minutes per month for Relay and an upper limit of 5000 PCU for Vivox Voice Chat.

Hence, there should be explorations of other third-party services that offer a similar functionality and combine multiple third-party services to ensure that the application can cater to a larger number of users in future studies. Some other third-party services could include Oracle Cloud for database, Mirror for multiplayer and Agora for voice chat functionalities.

There should also be explorations on how to use the data collected on the usage of virtual meeting rooms for personalization and performance improvement in future studies. For example, since both the emails of the participants invited to a meeting and the emails of the participants who joined the meeting session were stored in MongoDB Atlas, future studies could develop an algorithm to find out which user often misses a meeting and send reminders to them through their email.

Before the further growth of the Metaverse Education market around the world, developers of Metaverse-based education applications should consider exploring ways where they can help educational institutes manage the use of the expensive hardware. They can do so by developing features that can provide educational institutes with scheduling and reservation functionalities.

References

[1] B. Kye, N. Han, E. Kim, Y. Park, and S. Jo, “Educational applications of metaverse: Possibilities and limitations,” Journal of Educational Evaluation for Health Professions, vol. 18, p. 32, Dec. 2021. doi:10.3352/jeehp.2021.18.32

[2] Statista, “ Metaverse: market data & analysis report,” *Statista*, Sep. 2023. Available: https://www.statista.com/study/132822/metaverse-market-report/

[3] L. Zonaphan, K. Northus, J. Wijaya, S. Achmad, and R. Sutoyo, “Metaverse as a future of education: A systematic review,” 2022 8th International HCI and UX Conference in Indonesia (CHIuXiD), Nov. 2022. doi:10.1109/chiuxid57244.2022.10009854

[4] M. A. Camilleri, “Metaverse applications in education: A systematic review and a cost-benefit analysis,” Interactive Technology and Smart Education, Jun. 2023. doi:10.1108/itse-01-2023-0017

[5] Y. Qian, J. Wang, and Y. Cai, “Revolutionizing educational landscapes: A systematic review of metaverse applications, Paradigms and emerging technologies,” *Cogent Education*, vol. 10, no. 2, Oct. 2023. doi:10.1080/2331186x.2023.2264006

[6] H.-M. Huang, U. Rauch, and S.-S. Liaw, “Investigating learners’ attitudes toward virtual reality learning environments: Based on a constructivist approach,” *Computers & Education*, vol. 55, no. 3, pp. 1171–1182, Nov. 2010. doi:10.1016/j.compedu.2010.05.014

Appendices

Appendix A

A.1 LoginUI Scene

This subsection contains the testing and results of the functionalities in the LoginUI Scene.

A.1.1 Login UI

Test Case ID:	LoginUI-001		
Test Case Name:	Successful login in Login UI		
Test Case Description:	The user successfully logs into the application.		
Pre-Conditions:	1. The user is currently at the Login UI of the application. 2. The user already has a registered email and password, and enters them correctly.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Log in” button.	The scene would change from the LoginUI Scene to the Main Scene.	The scene was changed from the LoginUI Scene to the Main Scene.

Test Case ID:	LoginUI-002
Test Case Name:	Failed login in Login UI due to wrong email or password
Test Case Description:	The user fails to log into the application. The reasons for failure could be invalid email or password.
Pre-Conditions:	1. The user is currently at the Login UI of the application. 2. The user already has a registered email and password.

Test Case No.	Description	Expected Outcome	Actual Output
1	The user keys in their registered email and an **invalid password**, then click on the “Log in” button.	An error message would be shown in Login UI.	Error message “Invalid email address or password” was shown in Login UI.
2	The user keys in their registered password and an **invalid email**, then click on the “Log in” button.	An error message would be shown in Login UI.	Error message “User not found” was shown in Login UI.

Test Case ID:	LoginUI-003		
Test Case Name:	Toggle from Login UI to Register UI		
Test Case Description:	The user toggles from Login UI to Register UI by clicking on the “Register here” button.		
Pre-Conditions:	1. The user is currently at the Login UI of the application.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Register here” button.	The UI would change from Login UI to Register UI.	The UI was changed from Login UI to Register UI.

A.1.2 Register UI

Test Case ID:	RegisterUI-001
Test Case Name:	Successful registration in Register UI
Test Case Description:	The user successfully registers for an account.
Pre-Conditions:	1. The user is currently at the Register UI of the application. 2. The user keys in a valid email and password.

Test Case No.	Description	Expected Outcome	Actual Output
1	The user chooses “Student” in the dropdown, then clicks on the “Register” button.	The UI would change from Register UI to SetUserProfileStudent UI.	The UI was changed from Register UI to SetUserProfileStudent UI.
2	The user chooses “Professor/TA” in the dropdown, then clicks on the “Register” button.	The UI would change from Register UI to SetUserProfileOthers UI.	The UI was changed from Register UI to SetUserProfileOthers UI.
3	The user chooses “Lab Admin” in the dropdown, then clicks on the “Register” button.	The UI would change from Register UI to SetUserProfileOthers UI.	The UI was changed from Register UI to SetUserProfileOthers UI.

Test Case ID:	RegisterUI-002		
Test Case Name:	Failed registration in Register UI		
Test Case Description:	The user fails to register for an account. The reasons for failure could be invalid email, invalid password or not indicating identity.		
Pre-Conditions:	1. The user is currently at the Register UI of the application.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user keys in an **invalid email**, valid password and indicates their identity in the dropdown, then click on the “Register” button.	An error message would be shown in Register UI.	Error message “Email address is not valid” was shown in Register UI.
2	The user keys in an **invalid password**, valid email and indicates their	An error message would be shown in Register UI.	Error message “Password must be between 6 and 100

	identity in the dropdown, then click on the “Register” button.		characters” was shown in Register UI.
3	The user keys in n valid email, valid password and but **does not choose any option in the dropdown**, then click on the “Register” button.	An error message would be shown in Register UI.	Error message “Missing input(s). Unable to register.” was shown in Register UI.

Test Case ID:	RegisterUI-003		
Test Case Name:	Toggle from Register UI to Login UI		
Test Case Description:	The user toggles from Register UI to Login UI by clicking on the “Log in here” button.		
Pre-Conditions:	1. The user is currently at the Register UI of the application.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Log in here” button.	The UI would change from Register UI to Login UI.	The UI was changed from Register UI to Login UI.

A.1.3 SetUserProfile UIs

Test Case ID:	SetUserProfileStudentUI-001
Test Case Name:	Successfully complete registration process in SetUserProfileStudent UI.
Test Case Description:	The user successfully completes registration.
Pre-Conditions:	1. The user is currently at the SetUserProfileStudent UI of the application. 2. The user has provided inputs such as the Preferred Display Name, School, Course and Year.

Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Continue” button.	The scene would change from the LoginUI Scene to the Main Scene.	The scene was changed from the LoginUI Scene to the Main Scene.

Test Case ID:	SetUserProfileStudentUI-002		
Test Case Name:	Failed to complete registration process in SetUserProfileStudent UI		
Test Case Description:	The user fails to completes registration. The reasons for failure could be no inputs to the Preferred Display Name, School, Course or Year.		
Pre-Conditions:	1. The user is currently at the SetUserProfileStudent UI of the application.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user provides inputs to School, Course and Year, but **not Preferred Display Name**, then clicks on the “Continue” button.	An error message would be shown in SetUserProfileStudent UI.	Error message “Unable to continue. Missing input(s).” was shown in SetUserProfileStudent UI.
2	The user provides inputs to Preferred Display Name, Course and Year but **not School**, then clicks on the “Continue” button.	An error message would be shown in SetUserProfileStudent UI.	Error message “Unable to continue. Missing input(s).” was shown in SetUserProfileStudent UI.
3	The user provides inputs to Preferred Display Name, School and Year, but **not Course**, then clicks on the “Continue” button.	An error message would be shown in SetUserProfileStudent UI.	Error message “Unable to continue. Missing input(s).” was shown in SetUserProfileStudent UI.

4	The user provides inputs to Preferred Display Name, School and Course, but **not Year**, then clicks on the "Continue" button.	An error message would be shown in SetUserProfileStudent UI.	Error message "Unable to continue. Missing input(s)." was shown in SetUserProfileStudent UI.

Test Case ID:	SetUserProfileOthersUI-001		
Test Case Name:	Successfully complete registration process in SetUserProfileOthers UI.		
Test Case Description:	The user successfully completes registration.		
Pre-Conditions:	1. The user is currently at the SetUserProfileOthers UI of the application. 2. The user has provided inputs such as the Preferred Display Name and School.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the "Continue" button.	The scene would change from the LoginUI Scene to the Main Scene.	The scene was changed from the LoginUI Scene to the Main Scene.

Test Case ID:	SetUserProfileOthersUI-002		
Test Case Name:	Failed to complete registration process in SetUserProfileOthers UI		
Test Case Description:	The user fails to completes registration. The reasons for failure could be no inputs to the Preferred Display Name or School.		
Pre-Conditions:	1. The user is currently at the SetUserProfileOthers UI of the application.		
Test Case No.	Description	Expected Outcome	Actual Output

1	The user provides inputs to School but **not Preferred Display Name**, then clicks on the “Continue” button.	An error message would be shown in SetUserProfileOthers UI.	Error message “Unable to continue. Missing input(s).” was shown in SetUserProfileOthers UI.
2	The user provides inputs to Preferred Display Name but **not School**, then clicks on the “Continue” button.	An error message would be shown in SetUserProfileOthers UI.	Error message “Unable to continue. Missing input(s).” was shown in SetUserProfileOthers UI.

A.2 Main Scene

This subsection contains the testing and results of the functionalities in the Main Scene.

Test Case ID:	ThirdPersonCamera-001		
Test Case Name:	ThirdPersonCamera functions		
Test Case Description:	The user is moving “Player” around in the ThirdPersonCamera. The user also tries to switch to the FirstPersonCamera.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. No UI is currently set as active. 3. The ThirdPersonCamera is set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user controls the “Player” using the “W”, “A”, “S” and “D” keys.	The “Player” would be able to move around the scene in ThirdPersonCamera.	The “Player” was able to move around the scene in ThirdPersonCamera.

2	The user switches to the FirstPersonCamera by pressing on the “C” key.	The camera would switch to the FirstPersonCamera.	The camera was switched to the FirstPersonCamera.

Test Case ID:	FirstPersonCamera-001		
Test Case Name:	FirstPersonCamera functions		
Test Case Description:	The user is moving “Player” around in the FirstPersonCamera. The user also tries to switch to the ThirdPersonCamera.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. No UI is currently set as active. 3. The FirstPersonCamera is set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user controls the “Player” using the “W”, “A”, “S” and “D” keys.	The “Player” would be able to move around the scene in FirstPersonCamera.	The “Player” was able to move around the scene in FirstPersonCamera.
2	The user rotates the FirstPersonCamera using the “Q” and “E” keys.	The FirstPersonCamera would be able to rotate left and right in FirstPersonCamera.	The FirstPersonCamera was able to rotate left and right in FirstPersonCamera.
3	The user switches to the ThirdPersonCamera by pressing on the “C” key.	The camera would switch to the ThirdPersonCamera.	The camera was switched to the ThirdPersonCamera.

A.2.1 LogOut UI

Test Case ID:	LogOutUI-001
Test Case Name:	Open LogOut UI

Test Case Description:	The user opens the LogOut UI.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. No UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user makes “Player” come into contact with the 3D object “Bed”	The LogOut UI would be set as active and “Player” is disabled.	The LogOut UI was set as active and “Player” was disabled.

Test Case ID:	LogOutUI-002		
Test Case Name:	LogOut UI functions		
Test Case Description:	The user clicks on the buttons in the LogOut UI. The LogOut UI consists of a “Confirm” button and a “Cancel” button.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The LogOut UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Confirm” button.	The application would close.	The application was closed.
2	The user clicks on the “Cancel” button.	The LogOut UI would be set as inactive and “Player” is enabled.	The LogOut UI was set as inactive and “Player” was enabled.

A.2.2 UserProfile UIs

Test Case ID:	UserProfileStudentUI-001
Test Case Name:	Open UserProfileStudent UI

Test Case Description:	The user opens the UserProfileStudent UI.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. No UI is currently set as active. 3. The current account is a “Student” account.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user makes “Player” come into contact with the 3D object “Wardrobe”	The UserProfileStudent UI would be set as active and “Player” is disabled.	The UserProfileStudent UI was set as active and “Player” was disabled.

Test Case ID:	UserProfileStudentUI-002		
Test Case Name:	Successfully updates user profile in UserProfileStudent UI.		
Test Case Description:	The user successfully updates the user profile.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The UserProfileStudent UI is currently set as active. 3. The user has provided inputs such as the Display Name, School, Course and Year.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Update Profile” button.	A success message would be shown in UserProfileStudent UI.	Success message “Saved successfully” was shown in UserProfileStudent UI.

Test Case ID:	UserProfileStudentUI-003
Test Case Name:	Fails to update user profile in UserProfileStudent UI.
Test Case Description:	The user fails to update the user profile. The reasons for failure could be no inputs to the Display Name, School, Course or Year.

Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The UserProfileStudent UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user provides inputs to School, Course and Year, but **not Display Name**, then clicks on the “Update Profile” button.	An error message would be shown in UserProfileStudent UI.	Error message “Error. Display name, School, Course and/or Year of study cannot be empty.” was shown in UserProfileStudent UI.
2	The user provides inputs to Display Name, Course and Year but **not School**, then clicks on the “Update Profile” button.	An error message would be shown in UserProfileStudent UI.	Error message “Error. Display name, School, Course and/or Year of study cannot be empty.” was shown in UserProfileStudent UI.
3	The user provides inputs to Display Name, School and Year, but **not Course**, then clicks on the “Update Profile” button.	An error message would be shown in UserProfileStudent UI.	Error message “Error. Display name, School, Course and/or Year of study cannot be empty.” was shown in UserProfileStudent UI.
4	The user provides inputs to Display Name, School and Course, but **not Year**, then clicks on the “Update Profile” button.	An error message would be shown in UserProfileStudent UI.	Error message “Error. Display name, School, Course and/or Year of study cannot be empty.” was shown in UserProfileStudent UI.

Test Case ID:	UserProfileStudentUI-004
Test Case Name:	Close UserProfileStudent UI

Test Case Description:	The user clicks on the “X” button in the UserProfileStudent UI.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The UserProfileStudent UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “X” button.	The UserProfileStudent UI would be set as inactive and “Player” is enabled.	The UserProfileStudent UI was set as inactive and “Player” was enabled.

Test Case ID:	UserProfileOthersUI-001		
Test Case Name:	Open UserProfileOthers UI		
Test Case Description:	The user opens the UserProfileOthers UI.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. No UI is currently set as active. 3. The current account is a “Professor/TA” or “Lab Admin” account.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user makes “Player” come into contact with the 3D object “Wardrobe”	The UserProfileOthers UI would be set as active and “Player” is disabled.	The UserProfileOthers UI was set as active and “Player” was disabled.

Test Case ID:	UserProfileOthersUI-002
Test Case Name:	Successfully updates user profile in UserProfileOthers UI.
Test Case Description:	The user successfully updates the user profile.
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The UserProfileOthers UI is currently set as active.

	3. The user has provided inputs such as the Display Name and School.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Update Profile” button.	A success message would be shown in UserProfileOthers UI.	Success message “Saved successfully” was shown in UserProfileOthers UI.

Test Case ID:	UserProfileOthersUI-003		
Test Case Name:	Fails to update user profile in UserProfileOthers UI.		
Test Case Description:	The user fails to update the user profile. The reasons for failure could be no inputs to the Display Name or School.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The UserProfileOthers UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user provides inputs to School but **not Display Name**, then clicks on the “Update Profile” button.	An error message would be shown in UserProfileOthers UI.	Error message “Error. Display name and/or School cannot be empty.” was shown in UserProfileOthers UI.
2	The user provides inputs to Display Name but **not School**, then clicks on the “Update Profile” button.	An error message would be shown in UserProfileOthers UI.	Error message “Error. Display name and/or School cannot be empty.” was shown in UserProfileOthers UI.

Test Case ID:	UserProfileOthersUI-004
Test Case Name:	Close UserProfileOthers UI.
Test Case Description:	The user clicks on the “X” button in the UserProfileOthers UI.

Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The UserProfileOthers UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “X” button.	The UserProfileOthers UI would be set as inactive and “Player” is enabled.	The UserProfileOthers UI was set as inactive and “Player” was enabled.

A.2.3 MainMenu UIs

Test Case ID:	MainMenuStudentUI-001		
Test Case Name:	Open MainMenuStudent UI		
Test Case Description:	The user opens the MainMenuStudent UI.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. No UI is currently set as active. 3. The current account is a “Student” account.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user makes “Player” come into contact with the 3D object “Table”	The MainMenuStudent UI would be set as active and “Player” is disabled.	The MainMenuStudent UI was set as active and “Player” was disabled.

Test Case ID:	MainMenuStudentUI-002
Test Case Name:	MainMenuStudent UI functions
Test Case Description:	The user clicks on the buttons in the MainMenuStudent UI. The MainMenuStudent UI consists of a “Meeting Schedule” button and a “X” button.
Pre-Conditions:	1. The user is currently at the Main Scene of the application.

	2. The MainMenuStudent UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Meeting Schedule” button.	The UI would change from MainMenuStudent UI to MeetingSchedule UI.	The UI was changed from MainMenuStudent UI to MeetingSchedule UI.
2	The user clicks on the “X” button.	The MainMenuStudent UI would be set as inactive and “Player” is enabled.	The MainMenuStudent UI was set as inactive and “Player” was enabled.

Test Case ID:	MainMenuProfUI-001		
Test Case Name:	Open MainMenuProf UI		
Test Case Description:	The user opens the MainMenuProf UI.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. No UI is currently set as active. 3. The current account is a “Professor/TA” account.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user makes “Player” come into contact with the 3D object “Table”	The MainMenuProf UI would be set as active and “Player” is disabled.	The MainMenuProf UI was set as active and “Player” was disabled.

Test Case ID:	MainMenuProfUI-002
Test Case Name:	MainMenuProf UI functions
Test Case Description:	The user clicks on the buttons in the MainMenuProf UI. The MainMenuProf UI consists of a “Meeting Schedule” button, “Resource Reservation” button and a “X” button.

Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The MainMenuProf UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Meeting Schedule” button.	The UI would change from MainMenuProf UI to MeetingSchedule UI.	The UI was changed from MainMenuProf UI to MeetingSchedule UI.
2	The user clicks on the “Resource Reservation” button	The UI would change from MainMenuProf UI to ResourceReservationProf of UI.	The UI was changed from MainMenuProf UI to ResourceReservationProf UI
3	The user clicks on the “X” button.	The MainMenuProf UI would be set as inactive and “Player” is enabled.	The MainMenuProf UI was set as inactive and “Player” was enabled.

Test Case ID:	MainMenuStaffUI-001		
Test Case Name:	Open MainMenuStaff UI		
Test Case Description:	The user opens the MainMenuStaff UI.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. No UI is currently set as active. 3. The current account is a “Lab Admin” account.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user makes “Player” come into contact with the 3D object “Table”	The MainMenuStaff UI would be set as active and “Player” is disabled.	The MainMenuStaff UI was set as active and “Player” was disabled.

Test Case ID:	MainMenuStaffUI-002		
Test Case Name:	MainMenuStaff UI functions		
Test Case Description:	The user clicks on the buttons in the MainMenuStaff UI. The MainMenuStaff UI consists of a “Meeting Schedule” button, “Resource Reservation” button and a “X” button.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The MainMenuStaff UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Meeting Schedule” button.	The UI would change from MainMenuStaff UI to MeetingSchedule UI.	The UI was changed from MainMenuStaff UI to MeetingSchedule UI.
2	The user clicks on the “Resource Reservation” button	The UI would change from MainMenuStaff UI to ResourceReservationStaff UI.	The UI was changed from MainMenuStaff UI to ResourceReservationStaff UI
3	The user clicks on the “X” button.	The MainMenuStaff UI would be set as inactive and “Player” is enabled.	The MainMenuStaff UI was set as inactive and “Player” was enabled.

A.2.4 MeetingSchedule UI

Test Case ID:	MeetingScheduleUI-001
Test Case Name:	MeetingSchedule UI functions
Test Case Description:	The user clicks on the buttons in the MeetingSchedule UI. The MeetingSchedule UI consists of date buttons, a left arrow button, a right arrow button and a “Create Meeting” button.
Pre-Conditions:	1. The user is currently at the Main Scene of the application.

	2. The MeetingSchedule UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on a date button.	The UI would change from MeetingSchedule UI to MeetingDetails UI.	The UI was changed from MeetingSchedule UI to MeetingDetails UI.
2	The user clicks on the left arrow button.	The dates of the MeetingSchedule UI would be updated to that of the previous month.	The dates of the MeetingSchedule UI was updated to that of the previous month.
3	The user clicks on the right arrow button.	The dates of the MeetingSchedule UI would be updated to that of the next month.	The dates of the MeetingSchedule UI was updated to that of the next month.
4	The user clicks on the “Create Meeting” button	The UI would change from MeetingSchedule UI to NewMeeting UI.	The UI was changed from MeetingSchedule UI to NewMeeting UI.

Test Case ID:	MeetingScheduleUI-002		
Test Case Name:	Close MeetingSchedule UI		
Test Case Description:	The user clicks on the buttons in the MeetingSchedule UI. The MeetingSchedule UI also consists of a “X” button.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The MeetingSchedule UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “X” button and the current account is a “Student” account.	The UI would change from MeetingSchedule UI to MainMenuStudent UI.	The UI was changed from MeetingSchedule UI

			to MainMenuStudent UI.
2	The user clicks on the “X” button and the current account is a “Professor/TA” account.	The UI would change from MeetingSchedule UI to MainMenuProf UI.	The UI was changed from MeetingSchedule UI to MainMenuProf UI.
3	The user clicks on the “X” button and the current account is a “Lab Admin” account.	The UI would change from MeetingSchedule UI to MainMenuStaff UI.	The UI was changed from MeetingSchedule UI to MainMenuStaff UI.

A.2.5 MeetingDetails UI

Test Case ID:	MeetingDetailsUI-001		
Test Case Name:	MeetingDetails UI functions		
Test Case Description:	The user clicks on the buttons in the MeetingDetails UI. The MeetingDetails UI consists of meeting buttons, a “Start Meeting” button, a “Delete Meeting” button and a “X” button.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The MeetingDetails UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on a white meeting button.	The details of the meeting would be shown on the right-hand side of the MeetingDetails UI, with “Start Meeting” and “Delete Meeting” buttons disabled.	The details of the meeting was shown on the right-hand side of the MeetingDetails UI, with “Start Meeting” and “Delete Meeting” buttons disabled.

2	The user clicks on a blue meeting button.	The details of the meeting would be shown on the right-hand side of the MeetingDetails UI, with “Start Meeting” and “Delete Meeting” buttons enabled	The details of the meeting was shown on the right-hand side of the MeetingDetails UI, with “Start Meeting” and “Delete Meeting” buttons enabled
3	The user clicks on a blue meeting button, then clicks on the “Start Meeting” button.	The scene would change from the Main Scene to the ClassRoom Scene.	The scene was changed from the Main Scene to the ClassRoom Scene.
4	The user clicks on a blue meeting button, then clicks on the “Delete Meeting” button.	The MeetingDetails UI would refresh to show the updated list of scheduled meetings.	The MeetingDetails UI was refreshed to show the updated list of scheduled meetings.
5	The user clicks on the “X” button.	The UI would change from MeetingDetails UI to MeetingSchedule UI.	The UI was changed from MeetingDetails UI to MeetingSchedule UI.

A.2.6 NewMeeting UI

Test Case ID:	NewMeetingUI-001
Test Case Name:	Successfully schedules a new meeting in NewMeeting UI
Test Case Description:	The user successfully schedules a new meeting.
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The NewMeeting UI is currently set as active. 3. The user keys in information such as the Date, Start time, Duration and Description.

Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Create Meeting” button.	A success message would be shown in NewMeeting UI.	Success message “Meeting created” was shown in NewMeeting UI.

Test Case ID:	NewMeetingUI-002		
Test Case Name:	Failed to schedule a new meeting in NewMeeting UI		
Test Case Description:	The user fails to schedule a new meeting. The reasons for failure could be invalid Date, Start time, Duration or Description.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The NewMeeting UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user keys in an **invalid Date**, valid Start time, Duration and Description, then clicks on the “Create Meeting” button.	An error message would be shown in NewMeeting UI.	Error message “Unable to create meeting. Please make sure date and start time is valid.” was shown in NewMeeting UI.
2	The user keys in an **invalid Start time**, valid Date, Duration and Description, then clicks on the “Create Meeting” button.	An error message would be shown in NewMeeting UI.	Error message “Unable to create meeting. Please make sure date and start time is valid.” was shown in NewMeeting UI.
3	The user keys in an **invalid Duration**, valid Date, Start time and Description, then clicks on the “Create Meeting” button.	An error message would be shown in NewMeeting UI.	Error message “Unable to create meeting. Missing input(s).” was shown in NewMeeting UI.

4	The user keys in an **invalid Description**, valid Date, Start time and Duration , then clicks on the “Create Meeting” button.	An error message would be shown in NewMeeting UI.	Error message “Unable to create meeting. Missing input(s).” was shown in NewMeeting UI.

Test Case ID:	NewMeetingUI-003		
Test Case Name:	Successfully adds a participant in NewMeeting UI		
Test Case Description:	The user successfully adds a participant.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The NewMeeting UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user keys in a valid email that has not already been added to the list of participants, then clicks on the “Add” button.	A success message and the participant’s email would be shown in NewMeeting UI.	Success message “Email successfully added” and participant’s email was show in NewMeeting UI.

Test Case ID:	NewMeetingUI-004		
Test Case Name:	Failed to add a participant in NewMeeting UI		
Test Case Description:	The user fails to add a participant. The reasons for failure could be invalid email, or valid email that is already in the list of participants.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The NewMeeting UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output

1	The user keys in an invalid email, then clicks on the “Add” button.	An error message would be shown in NewMeeting UI.	Error message “Email address is not valid” was shown in NewMeeting UI.
2	The user keys in a valid email that has already been added to the list of participants, then clicks on the “Add” button.	An error message would be shown in NewMeeting UI.	Error message “Email already added” was shown in NewMeeting UI.

Test Case ID:	NewMeetingUI-005		
Test Case Name:	Close NewMeeting UI		
Test Case Description:	The user clicks on the “X” button in NewMeeting UI.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The NewMeeting UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “X” button.	The UI would change from NewMeeting UI to MeetingSchedule UI.	The UI was changed from NewMeeting UI to MeetingSchedule UI.

A.2.7 ResourceReservation UIs

Test Case ID:	ResourceReservationsProfUI-001
Test Case Name:	ResourceReservationsProf UI functions
Test Case Description:	The user clicks on the buttons in the ResourceReservationsProf UI. The ResourceReservationsProf UI consists of date buttons, a left arrow button, a right arrow button and a “X” button.
Pre-Conditions:	1. The user is currently at the Main Scene of the application.

	2. The ResourceReservationsProf UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on a date button.	The UI would change from ResourceReservationsProf UI to DateDetailsProf UI.	The UI was changed from ResourceReservations Prof UI to DateDetailsProf UI.
2	The user clicks on the left arrow button.	The dates of the ResourceReservationsProf UI would be updated to that of the previous month.	The dates of the ResourceReservations Prof UI was updated to that of the previous month.
3	The user clicks on the right arrow button.	The dates of the ResourceReservationsProf UI would be updated to that of the next month.	The dates of the ResourceReservations Prof UI was updated to that of the next month.
4	The user clicks on the “X” button.	The UI would change from ResourceReservationsProf UI to MainMenuProf UI.	The UI was changed from ResourceReservations Prof UI to MainMenuProf UI.

Test Case ID:	ResourceReservationsStaffUI-001		
Test Case Name:	ResourceReservationsStaff UI functions		
Test Case Description:	The user clicks on the buttons in the ResourceReservationsStaff UI. The ResourceReservationsStaff UI consists of date buttons, a left arrow button, a right arrow button, a “Manage Slots” button and a “X” button.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The ResourceReservationsStaff UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output

1	The user clicks on a date button.	The UI would change from ResourceReservationsStaff UI to DateDetailsStaff UI.	The UI was changed from ResourceReservations Staff UI to DateDetailsStaff UI.
2	The user clicks on the left arrow button.	The dates of the ResourceReservationsStaff UI would be updated to that of the previous month.	The dates of the ResourceReservations Staff UI was updated to that of the previous month.
3	The user clicks on the right arrow button.	The dates of the ResourceReservationsStaff UI would be updated to that of the next month.	The dates of the ResourceReservations Staff UI was updated to that of the next month.
4	The user clicks on the “Manage Slots” button.	The UI would change from ResourceReservationsStaff UI to ManageSlots UI.	The UI was changed from ResourceReservations Staff UI to ManageSlots UI.
5	The user clicks on the “X” button.	The UI would change from ResourceReservationsStaff UI to MainMenuStaffUI.	The UI was changed from ResourceReservations Staff UI to MainMenuStaff UI.

A.2.8 ManageSlots UI

Test Case ID:	ManageSlotsUI-001
Test Case Name:	Successfully creates and removes available time slots in ManageSlots UI.
Test Case Description:	The user successfully creates new available time slots.

Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The ManageSlots UI is currently set as active. 3. The user keys in information such the Location, StartDate, EndDate, StartTime and EndTime.		
Test Case No.	Description	Expected Outcome	
1	The user clicks on the “Add slots” button.	A success message would be shown in ManageSlots UI.	Success message “Adding slots complete” was shown in ManageSlots UI.
2	The user clicks on the “Remove slots” button.	A success message would be shown in ManageSlots UI.	Success message “Removing slots complete” was shown in ManageSlots UI.

Test Case ID:	ManageSlotsUI-002		
Test Case Name:	Failed to create available time slots in ManageSlots UI.		
Test Case Description:	The user fails to create available time slots. The reasons for failure could be invalid Location, StartDate, EndDate, StartTime and EndTime.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The ManageSlots UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user keys in an **invalid Location**, valid StartDate, EndDate, StartTime and EndTime, then clicks on the “Add slots” button.	An error message would be shown in ManageSlots UI.	Error message “Unable to add slots. Missing input(s).” was shown in ManageSlots UI.

2	The user keys in an **invalid StartDate**, valid Location, EndDate, StartTime and EndTime, then clicks on the “Add slots” button.	An error message would be shown in ManageSlots UI.	Error message “Unable to add slots. Please make sure fromdate/time is earlier than todate/time and that they are a valid date.” was shown in ManageSlots UI.
3	The user keys in an **invalid EndDate**, valid Location, StartDate, StartTime and EndTime, then clicks on the “Add slots” button.	An error message would be shown in ManageSlots UI.	Error message “Unable to add slots. Please make sure fromdate/time is earlier than todate/time and that they are a valid date.” was shown in ManageSlots UI.
4	The user keys in an **invalid StartTime**, valid Location, StartDate, EndDate, and EndTime, then clicks on the “Add slots” button.	An error message would be shown in ManageSlots UI.	Error message “Unable to add slots. Please make sure fromdate/time is earlier than todate/time and that they are a valid date.” was shown in ManageSlots UI.
5	The user keys in an **invalid EndTime**, valid Location, StartDate, EndDate, and StartTime, then clicks on the “Add slots” button.	An error message would be shown in ManageSlots UI.	Error message “Unable to add slots. Please make sure fromdate/time is earlier than todate/time and that they are a valid date.” was shown in ManageSlots UI.

Test Case ID:	ManageSlotsUI-003		
Test Case Name:	Failed to remove available time slots in ManageSlots UI.		
Test Case Description:	The user fails to remove available time slots. The reasons for failure could be invalid Location, StartDate, EndDate, StartTime and EndTime.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The ManageSlots UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user keys in an **invalid Location**, valid StartDate, EndDate, StartTime and EndTime, then clicks on the "Remove slots" button.	An error message would be shown in ManageSlots UI.	Error message "Unable to remove slots. Missing input(s)." was shown in ManageSlots UI.
2	The user keys in an **invalid StartDate**, valid Location, EndDate, StartTime and EndTime, then clicks on the "Remove slots" button.	An error message would be shown in ManageSlots UI.	Error message "Unable to remove slots. Please make sure fromdate/time is earlier than todate/time and that they are a valid date." was shown in ManageSlots UI.
3	The user keys in an **invalid EndDate**, valid Location, StartDate, StartTime and EndTime, then clicks on the "Remove slots" button.	An error message would be shown in ManageSlots UI.	Error message "Unable to remove slots. Please make sure fromdate/time is earlier than todate/time and that they are a valid date." was shown in ManageSlots UI.

4	The user keys in an **invalid StartTime**, valid Location, StartDate, EndDate, and EndTime, then clicks on the “Remove slots” button.	An error message would be shown in ManageSlots UI.	Error message “Unable to remove slots. Please make sure fromdate/time is earlier than todate/time and that they are a valid date.” was shown in ManageSlots UI.
5	The user keys in an **invalid EndTime**, valid Location, StartDate, EndDate, and StartTime, then clicks on the “Remove slots” button.	An error message would be shown in ManageSlots UI.	Error message “Unable to remove slots. Please make sure fromdate/time is earlier than todate/time and that they are a valid date.” was shown in ManageSlots UI.

Test Case ID:	ManageSlotsUI-004		
Test Case Name:	Close ManageSlots UI.		
Test Case Description:	The user clicks on the “X” button in NewMeeting UI.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The ManageSlots UI is currently set as active.		
Test Case No.	Description	Expected Outcome	
1	The user clicks on the “X” button.	The UI would change from ManageSlots UI to ResourceReservationStaff UI.	The UI was changed from ManageSlots UI to ResourceReservationStaff UI.

A.2.9 DateDetails UIs

Test Case ID:	DateDetailsProfUI-001		
Test Case Name:	DateDetailsProf UI functions		
Test Case Description:	The user clicks on the buttons in the DateDetailsProf UI. The DateDetailsProf UI consists of location buttons and a “X” button.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The DateDetailsProf UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on a location button.	The UI would change from DateDetailsProf UI to TimeDetailsProf UI.	The UI was changed from DateDetailsProf UI to TimeDetailsProf UI.
2	The user clicks on the “X” button.	The UI would change from DateDetailsProf UI to ResourceReservationsProf UI	The UI was changed from DateDetailsProf UI to ResourceReservations Prof UI.

Test Case ID:	DateDetailsStaffUI-001		
Test Case Name:	DateDetailsStaff UI functions		
Test Case Description:	The user clicks on the buttons in the DateDetailsStaff UI. The DateDetailsStaff UI consists of location buttons and a “X” button.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The DateDetailsStaff UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on a location button.	The UI would change from DateDetailsStaff UI to TimeDetailsStaff UI.	The UI was changed from DateDetailsStaff UI to TimeDetailsStaff UI.

2	The user clicks on the “X” button.	The UI would change from DateDetailsStaff UI to ResourceReservationsStaff UI	The UI was changed from DateDetailsStaff UI to ResourceReservations Staff UI.

A.2.10 TimeDetails UIs

Test Case ID:	TimeDetailsProfUI-001		
Test Case Name:	TimeDetailsProf UI functions		
Test Case Description:	The user clicks on the buttons in the TimeDetailsProf UI. The TimeDetailsProf UI consists of time slots buttons, a “Add Reservation” button, a “Remove Reservation” button and a “X” button.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The TimeDetailsProf UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on a white time slot button.	The time slot button selected in TimeDetailsProf UI would turn from white to green.	The time slot button selected in TimeDetailsProf UI was turned from white to green.
2	The user clicks on a blue time slot button.	The time slot button selected in TimeDetailsProf UI would turn from blue to red.	The time slot button selected in TimeDetailsProf UI was turned from blue to red.
3	The user clicks on a white time slot button, then clicks on the “Add Reservation” button.	The time slot button selected in TimeDetailsProf UI would turn from green to blue.	The time slot button selected in TimeDetailsProf UI was turned from green to blue.

4	The user clicks on a blue time slot button, then clicks on the “Remove Reservation” button.	The time slot button selected in TimeDetailsProf UI would turn from red to white.	The time slot button selected in TimeDetailsProf UI was turned from red to white.
5	The user clicks on the “X” button.	The UI would change from TimeDetailsProf UI to DateDetailsProf UI.	The UI was changed from TimeDetailsProf UI to DateDetailsProf UI.

Test Case ID:	TimeDetailsStaffUI-001		
Test Case Name:	TimeDetailsStaff UI functions		
Test Case Description:	The user clicks on the buttons in the TimeDetailsStaff UI. The TimeDetailsStaff UI consists of time slots buttons and a “X” button.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The TimeDetailsStaff UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on a grey time slot button.	The details of the time slot would be shown on the right-hand side of the TimeDetailsStaff UI.	The details of the time slot was shown on the right-hand side of the TimeDetailsStaff UI.
2	The user clicks on the “X” button.	The UI would change from TimeDetailsStaff UI to DateDetailsStaff UI.	The UI was changed from TimeDetailsStaff UI to DateDetailsStaff UI.

A.2.11 JoinMeeting UIs

Test Case ID:	JoinMeetingUI-001		
Test Case Name:	Open JoinMeeting UI		
Test Case Description:	The user opens the JoinMeeting UI.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. No UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user makes “Player” come into contact with the 3D object “Door”	The JoinMeeting UI would be set as active and “Player” is disabled.	The JoinMeeting UI was set as active and “Player” was disabled

Test Case ID:	JoinMeetingUI-002		
Test Case Name:	JoinMeeting UI functions		
Test Case Description:	The user clicks on the buttons in the JoinMeeting UI. The JoinMeeting UI consists of a “Join” button and a “X” button.		
Pre-Conditions:	1. The user is currently at the Main Scene of the application. 2. The JoinMeeting UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user keys in a JoinCode and then clicks on the “Join” button.	The scene would change from Main Scene to ClassRoom Scene.	The scene was changed from Main Scene to ClassRoom Scene.
2	The user clicks on the “X” button.	The JoinMeeting UI would be set as inactive and “Player” is enabled.	The JoinMeeting UI was set as inactive and “Player” was enabled.

A.3 ClassRoom Scene

This subsection contains the testing and results of the functionalities in the ClassRoom Scene.

A.3.1 NetworkManager UI

Test Case ID:	NetworkManagerUIHost-001		
Test Case Name:	Successfully start a new meeting in NetworkManager UI as a host		
Test Case Description:	The user clicks on the buttons in the NetworkManager UI. The NetworkManager UI initially consists of a “Start” button.		
Pre-Conditions:	1. The user is currently at the ClassRoom Scene of the application. 2. The user came to this scene by clicking on the “Start Meeting” in MeetingDetails UI. 3. The NetworkManager UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Start” button.	The “Enable Audio” button, “Quizzes” button, “Classes” button and “End” button would be set as active and a “Player” is spawned.	The “Enable Audio” button, “Quizzes” button, “Classes” button and “End” button was set as active and a “Player” was spawned.

Test Case ID:	NetworkManagerUIHost-002
Test Case Name:	NetworkManager UI functions as a host
Test Case Description:	The user clicks on the buttons in the NetworkManager UI. The NetworkManager UI now contains the “Enable Audio” button, “Quizzes” button, “Classes” button and “End” button. The “Disable Audio” button would also be set as active when the “Enable Audio” button is clicked.
Pre-Conditions:	1. The user is currently at the ClassRoom Scene of the application.

	2. The user came to this scene by clicking on the “Start Meeting” in MeetingDetails UI. 3. The NetworkManager UI is currently set as active. 4. The user has already started the meeting session and another user has entered the session.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Enable Audio” button.	The “Disable Audio” button would be set as active in NetworkManager UI and the other user would be able to hear the user speaking.	The “Disable Audio” button was set as active in NetworkManager UI and the other user was able to hear the user speaking.
2	The user clicks on the “Enable Audio” button, and the clicks on the “Disable Audio” button.	The “Enable Audio” button would be set active in NetworkManager UI and the other user would no longer be able to hear the user speaking.	The “Enable Audio” button was set active in NetworkManager UI and the other user was no longer be able to hear the user speaking.
3	The user clicks on the “Quizzes” button.	The EnableQuiz UI would be set as active.	The EnableQuiz UI was set as active.
4	The user clicks on the “Classes” button.	The EnableClasses UI would be set as active.	The EnableClasses UI was set as active.
5	The user clicks on the “End” button.	The scene would change from ClassRoom Scene to Main Scene.	The scene was changed from ClassRoom Scene to Main Scene.

Test Case ID:	NetworkManagerUIClient-001
Test Case Name:	Successfully join a meeting session in NetworkManager UI as a client

Test Case Description:	The user successfully joins a meeting session.		
Pre-Conditions:	1. The user is currently at the ClassRoom Scene of the application. 2. The user came to this scene by clicking on the “Join” in JoinMeeting UI and entered a valid JoinCode. 3. The NetworkManager UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Start” button.	The “Enable Audio” button and “End” button would be set as active and a “Player” would be spawned.	The “Enable Audio” button and “End” button was set as active and a “Player” was spawned.

Test Case ID:	NetworkManagerUIClient-002		
Test Case Name:	Fails to join a meeting session in NetworkManager UI as a client		
Test Case Description:	The user fails to join a meeting session due to an invalid JoinCode entered in JoinMeeting UI.		
Pre-Conditions:	1. The user is currently at the ClassRoom Scene of the application. 2. The user came to this scene by clicking on the “Join” in JoinMeeting UI and entered an invalid JoinCode. 3. The NetworkManager UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Start” button.	The JoinCodeError UI would be set as active.	The JoinCodeError UI was set as active.

Test Case ID:	NetworkManagerUIClient-003
Test Case Name:	NetworkManager UI functions as a client
Test Case Description:	The user clicks on the buttons in NetworkManager UI. The NetworkManager UI now contains the “Enable Audio” button and “End”

	button. The “Disable Audio” button would also be set as active when the “Enable Audio” button is clicked.		
Pre-Conditions:	1. The user is currently at the ClassRoom Scene of the application. 2. The user came to this scene by clicking on the “Join” in JoinMeeting UI and entered a valid JoinCode. 3. The NetworkManager UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Enable Audio” button.	The “Disable Audio” button would be set as active in NetworkManager UI and the other user would be able to hear the user speaking.	The “Disable Audio” button was set as active in NetworkManager UI and the other user would be able to hear the user speaking.
2	The user clicks on the “Enable Audio” button, and the clicks on the “Disable Audio” button.	The “Enable Audio” button would be set active in NetworkManager UI and the other user would no longer be able to hear the user speaking	The “Enable Audio” button was set active in NetworkManager UI and the other user would no longer be able to hear the user speaking
3	The user clicks on the “End” button.	The scene would change from ClassRoom Scene to Main Scene.	The scene was changed from ClassRoom Scene to Main Scene.

A.3.2 EnableQuiz UI

Test Case ID:	EnableQuizUIHost-001
Test Case Name:	EnableQuiz UI functions as a host
Test Case	The user clicks on the buttons in the EnableQuiz UI. The EnableQuiz UI

Description:	contains the “Enable Sample” button and “X” button. The “Disable Sample” button would also be set active when the “Enable Sample” button is clicked.		
Pre-Conditions:	1. The user is currently at the ClassRoom Scene of the application. 2. The user came to this scene by clicking on the “Start Meeting” in MeetingDetails UI. 3. The NetworkManager UI and EnableQuiz UI is currently set as active. 4. Another user has entered the meeting session.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Enable Sample” button.	The “Join Quiz” button would be set as active in NetworkManager UI for host and client.	The “Join Quiz” button was set as active in NetworkManager UI for host and client.
2	The user clicks on the “Enable Sample” button, then clicks on the “Disable Sample” button.	The “Join Quiz” button would be set as active and then inactive in NetworkManager UI for host and client.	The “Join Quiz” button was set as active and then inactive in NetworkManager UI for host and client.
3	The user clicks on the “X” button.	The EnableQuiz UI would be set as inactive.	The EnableQuiz UI was set as inactive.

A.3.3 SampleQuiz UI

Test Case ID:	SampleQuizUIHost-001
Test Case Name:	Open SampleQuiz UI as a host
Test Case Description:	The user opens the SampleQuiz UI.
Pre-Conditions:	1. The user is currently at the ClassRoom Scene of the application. 2. The user came to this scene by clicking on the “Start Meeting” in MeetingDetails UI. 3. The NetworkManager UI is currently set as active. 4. The “Join Quiz” button is set as active in NetworkManager UI.

Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Join Quiz” button.	The SampleQuiz UI would be set as active.	The SampleQuiz UI was set as active.

Test Case ID:	SampleQuizUIHost-002		
Test Case Name:	SampleQuiz UI functions as a host		
Test Case Description:	The user clicks on the buttons in the SampleQuiz UI. The SampleQuiz UI consists of a right arrow button, a left arrow button and a “X” button.		
Pre-Conditions:	1. The user is currently at the ClassRoom Scene of the application. 2. The user came to this scene by clicking on the “Start Meeting” in MeetingDetails UI. 3. The NetworkManager UI and SampleQuiz UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the right arrow button.	The next question would be shown in SampleQuiz UI.	The next question was be shown in SampleQuiz UI.
2	The user clicks on the left arrow button.	The previous question would be shown in SampleQuiz UI.	The previous question was shown in SampleQuiz UI.
3	The user clicks on the “X” button.	The SampleQuiz UI would be set as inactive.	The SampleQuiz UI was set as inactive.

Test Case ID:	SampleQuizUIClient-001
Test Case Name:	Open SampleQuiz UI as a client
Test Case Description:	The user opens the SampleQuiz UI.
Pre-Conditions:	1. The user is currently at the ClassRoom Scene of the application.

	2. The user came to this scene by clicking on the “Join” in JoinMeeting UI and entered a valid JoinCode. 3. The NetworkManager UI is currently set as active. 4. The “Join Quiz” button is set as active in NetworkManager UI.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Join Quiz” button.	The SampleQuiz UI would be set as active.	The SampleQuiz UI was set as active.

Test Case ID:	SampleQuizUIClient-002		
Test Case Name:	SampleQuiz UI functions as a client		
Test Case Description:	The user clicks on the buttons in the SampleQuiz UI. The SampleQuiz UI consists of a right arrow button, a left arrow button and a “X” button.		
Pre-Conditions:	1. The user is currently at the ClassRoom Scene of the application. 2. The user came to this scene by clicking on the “Join” in JoinMeeting UI and entered a valid JoinCode. 3. The NetworkManager UI and SampleQuiz UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the right arrow button.	The next question would be shown in SampleQuiz UI.	The next question was shown in SampleQuiz UI.
2	The user clicks on the left arrow button.	The previous question would be shown in SampleQuiz UI.	The previous question was shown in SampleQuiz UI.
3	The user clicks on the “X” button.	The SampleQuiz UI would be set as inactive	The SampleQuiz UI was set as inactive

A.3.4 EnableClass UI

Test Case ID:	EnableClassUIHost-001		
Test Case Name:	EnableClass UI functions as a host		
Test Case Description:	The user clicks on the buttons in the EnableClass UI. The EnableQuiz UI contains the “Enable Sample” button and “X” button. The “Disable Sample” button would also be set active when the “Enable Sample” button is clicked.		
Pre-Conditions:	1. The user is currently at the ClassRoom Scene of the application. 2. The user came to this scene by clicking on the “Start Meeting” in MeetingDetails UI. 3. The NetworkManager UI and EnableClass UI is currently set as active. 4. Another user has entered the meeting session.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Enable Sample” button.	The “ClassSample” GameObject would be spawned in the ClassRoom Scene for host and client.	The “ClassSample” GameObject was spawned in the ClassRoom Scene for host and client.
2	The user clicks on the “Enable Sample” button, then clicks on the “Disable Sample” button.	The “ClassSample” GameObject would be spawned and then destroyed in the ClassRoom Scene for host and client.	The “ClassSample” GameObject was spawned and then destroyed in the ClassRoom Scene for host and client.
3	The user clicks on the “X” button.	The EnableClass UI would be set as inactive.	The EnableClass UI was set as inactive.

A.3.5 JoinCodeError UI

Test Case ID:	JoinCodeErrorUIClient-001		
Test Case Name:	JoinCodeError UI functions as a client		
Test Case Description:	The user clicks on the button in JoinCodeError UI. The JoinCodeError UI consists of a “Return” button		
Pre-Conditions:	1. The user is currently at the ClassRoom Scene of the application. 2. The user came to this scene by clicking on the “Join” in JoinMeeting UI and entered an invalid JoinCode. 3. The NetworkManager UI and JoinCodeError UI is currently set as active.		
Test Case No.	Description	Expected Outcome	Actual Output
1	The user clicks on the “Return” button.	The scene would change from ClassRoom Scene to Main Scene.	The scene was changed from ClassRoom Scene to Main Scene.