Club experience and application booth

Software Design Specification

2021.11.21

Introduction to Software Engineering 41

Team 6(EEA-VRChat)

Team Leader	2019314659	Minje Kim
Team Member	2018312292	Dongwon Kim
Team Member	2017314527	Myeongmin Kim
Team Member	2017311725	Doyeol Kim
Team Member	2017313156	Junyoung Lee
Team Member	2019312351	Chung Juwon

Contents

List of Tables

List of Figures

1. Preface

This section deals with readership, scope, objective of the project EEV-VRChat(Club Experience and Application Booth). It also contains document structure of this design specification document.

1.1 Readership

In this document, mainly 10 sections exist and there are some sub sections for each section if needed. Total structure of the Software Design Document (SDD) is shown in 1.4 Document Structure.

Main reader of the SDD is team 6 which is developing team of the project. The document can be used for having idea about design of the system. Professor and TAs are included in reader's scope.

1.2 Scope

The document can be used to give information about total architecture of the Club experience and application booth. This document deals with system architecture, protocol, database design and testing plan for each component. The software design document also contains developing plan and additional information such as document history and software design specification.

1.3 Objective

The main goal of the SDD is to give design information used by Software Engineering and Software Quality Engineering. The document has system architecture for frontend and backend of the system. For each part, objectives of the part, subcomponents related to the part are defined. Organization of the system architecture is explained with several diagrams such as class diagram, sequence diagram and entity-relationship diagram. Protocol and database design are also included. In protocol design since there is no protocol to send data because the Club experience and application booth system runs on the VR Chat, the section usually takes about synchronization between users. Database design is about primitive database in VR Chat since it is impossible to use off-the-shelf database. After explaining about designs, testing and development plans are given.

1.4 Document Structure

1. Preface

   This section is about things included in the document, main readers, and objective of the Software Design Document.

2. Introduction

   Diagrams, tools that are used in the document are introduced. Also, project scope and reference of the SDD is defined.

3. Overall System Architecture

   Using several diagrams such as context diagram and sequence diagram, total system architecture is given in the section.

4. Frontend System Architecture

   System architecture of frontend is given in this section with class diagram and sequence diagram.

5. Backend System Architecture

   System architecture of backend is given in this section with class diagram and sequence diagram.

6. Protocol Design

   This part usually takes about communication between client and server. However, since the project uses VR Chat, there is no specific protocol to communicate with server. Therefore, in this section, synchronization between users is included instead.

7. Database Design

   This section has several ER diagrams for database.

8. Testing Plan

   This section is about how the system will be tested. Testing policy is defined as a part of the plan.

9. Development Plan

   Development plan is given in this section. Developing plan contains tools to be used for design and implementing the system, constraints, assumption, and dependencies for the system.

10. Supporting Information

History of the software design document and design specification are contained in the section.

2. Introduction

2.1 Objectives

This section is for introducing diagrams and tools that are used for the project. System's scope and references of the whole document are also included in this part.

2.2 Applied Diagrams

2.2.1 UML

According to Microsoft's homepage [1], UML is a standard for software developers, project managers, company owners. UML diagrams are helpful for visualizing systems or process's operation and structures.

There are two types of diagrams in the UML: structural and behavioral diagrams. Structural diagram contains class, object, component, composite structure, deployment, package, and profile diagrams. Structural diagram is about static structure of systems, and each diagram shows different levels of abstraction and implementation. Behavioral diagrams focus on dynamic aspects of systems such as functionality. Activity diagram, use case diagram, interaction overview diagram, timing diagram, state machine diagram, sequence diagram, and communication diagram are kinds of behavioral diagrams.

There are several advantages for using UML.

a) Simplify complexities

b) Easy communication

c) Produce software and processes automatically

d) Easy to solve persistent architectural problems

e) Quality of work increases

f) Save time and cost

Followings are diagrams used in this SDD.

2.2.2 Context diagram

According to the website about context diagram [2], although context diagram is not UML diagram, since context model is useful for visualizing the operational context of a system instead of deployment diagram. Context diagram shows the other systems in the environment rather than the system being developed. The diagram focuses on showing

boundary between or part of systems. By context diagram, interacting environment can be shown.

2.2.3 Use case diagram

Focusing on what system does (rather than how system does), the diagram illustrates the set of sequences and represents the functional requirements of the system. Use case diagram consists of actors and events related to the actors. The diagram is useful to get what actors do clearly form customers.

2.2.4 Class diagram

The diagram looks like flow chart because each class is represented in box. The diagram shows logical and physical design of systems with classes. Since UML focus on object-oriented method, visualizing class and its relationship with class diagram can be helpful. The box is composed of class name, class attributes, class methods and operations.

2.2.5 Sequence diagram

The sequence diagram is useful for showing app types of business processes. The diagram visualizes the structure of system with sequence of messages and interactions between actors and objects. It is also helpful for multi-tasking.

2.2.6 Entity-Relationship diagram

The diagram is composed of entity types and relationships between them. People, places, or things can be entities. The diagram shows the logical structure of databases.

2.3 Applied Tools

2.3.1 Draw.io

Draw.io is a free online diagram software. With the software, drawing flowchart, network diagrams, UML diagrams, ER diagrams, database schema, BPMN, circuit diagrams ext. can be drawn. [3]

2.3.2 Visio

It is a tool for diagramming of Microsoft. With the software, it is possible to draw diagrams such as flowcharts, org charts, floor plans, brainstorming, network diagrams, and UML diagrams. Collaborating with coauthors using Microsoft Teams is also possible. [4]

2.4 Project Scope

The Club experience and application booth system provides a way to experience, explore and apply to club activities even if it is hard for people to meet face-to-face such as pandemic situation in 2020 and 2021. The system let students look around club booths in VR Chat as they did before covid-19. Furthermore, there are maps in the system that help people who are interested in the club activities by providing similar experience that they can do when enter the club. It is also possible to apply the club in the system at the interview rooms with interviewers of the specific club. The goal of the project is providing another way for people to enjoy club activities who cannot experience college life.

2.5 Reference

Using IEEE reference style,

[1] Microsoft, "Guide to UML diagramming and database modeling," Microsoft, 24 11 2018. [Online]. Available: https://www.microsoft.com/en-us/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling?rtc=1. [Accessed 10 11 2021].

[2] C. Adams, "What is a Context Diagram and what are the benefits of creating one?," Modern analyst.com, [Online]. Available: https://www.modernanalyst.com/Careers/InterviewQuestions/tabid/128/ID/1433/What-is-a-Context-Diagram-and-what-are-the-benefits-of-creating-one.aspx. [Accessed 10 11 2021].

[3] google, "draw.io," Draw.io, [Online]. Available: https://g.co/kgs/eM41RJ. [Accessed 10 11 2021].

[4] Microsoft, "Visio," Microsoft, [Online]. Available: https://www.microsoft.com/en-us/microsoft-365/visio/flowchart-software. [Accessed 10 11 2021].

3. System Architecture – Overall

3. 1 Objective

This chapter represents the overall configuration of the system, including the front end and back-end structure of the system. The focus is on briefly explaining the structure and flow of the system rather than describing the detailed front end and back-end architecture.

3. 2 System Organization

Both the user interface corresponding to the front end of the system and the database corresponding to the back end are controlled through VRChat.

The user interface corresponding to the front end is made by entering the map implemented in the VRChat. The map consists of several club booths in the world that embody the appearance of Sungkyunkwan University's Natural Science Campus. Each club booth has a portal to the world where you can experience clubs, and there is also an interview room for interviews to support the club, and a club manager room accessible only to managers who manage the club.

The way the backend of the system is managed is different from the general case. Unlike the general method of managing a database by creating a separate database server, database management of the system is performed by storing the database in an object inside the VRChat and accessing it. This is possible because the system has a less complex backend structure compared to the front end. The system can easily access the database within the object by utilizing coding within the VRChat.

The user's access to the corresponding system begins with the user's authentication. During the Authentication process, the backend operation is performed, and the user's data stored in the object and the user's data attempting access are compared. Users with successful Authentication can enter the world and freely travel around Club Boots and take a portal to Experience World if they want. Inside the club booth, there is a gallery, an interview room, and a space for club management. Experience World is divided into board game clubs, astronomical clubs, and band clubs, respectively. In the case of Othello game experience, the Othello game management system manages game play. Users who have become interested in clubs through club experience can conduct interviews with club managers in the interview room to apply for the club.

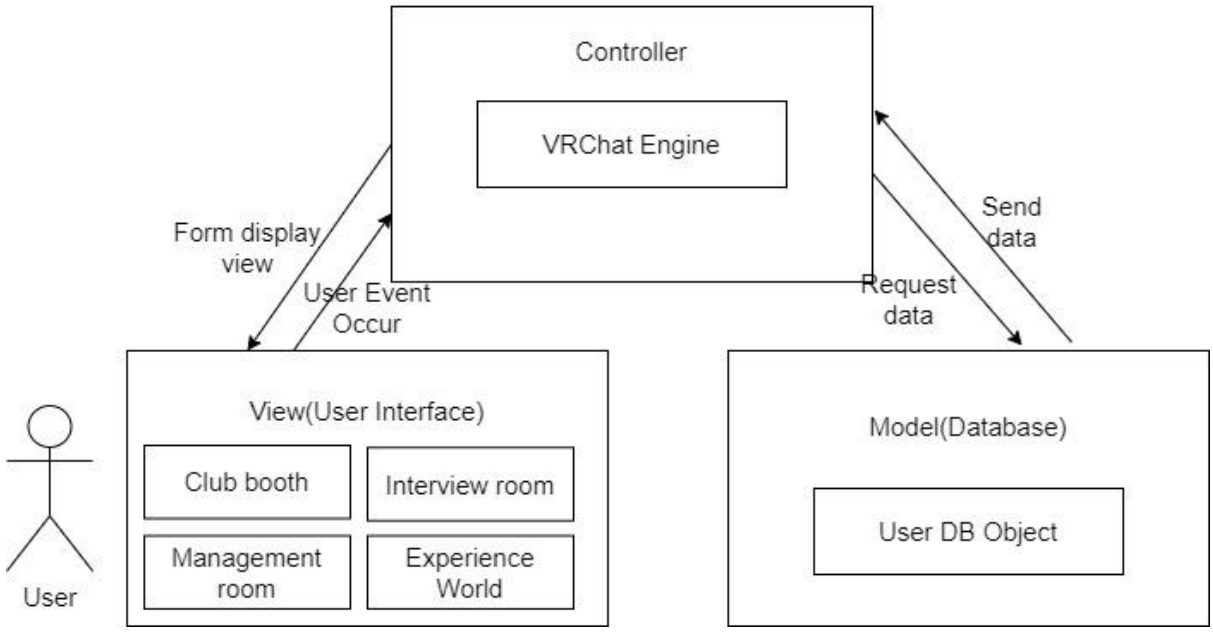

Figure 1 Overall system architecture

3.2.1 Context Diagram

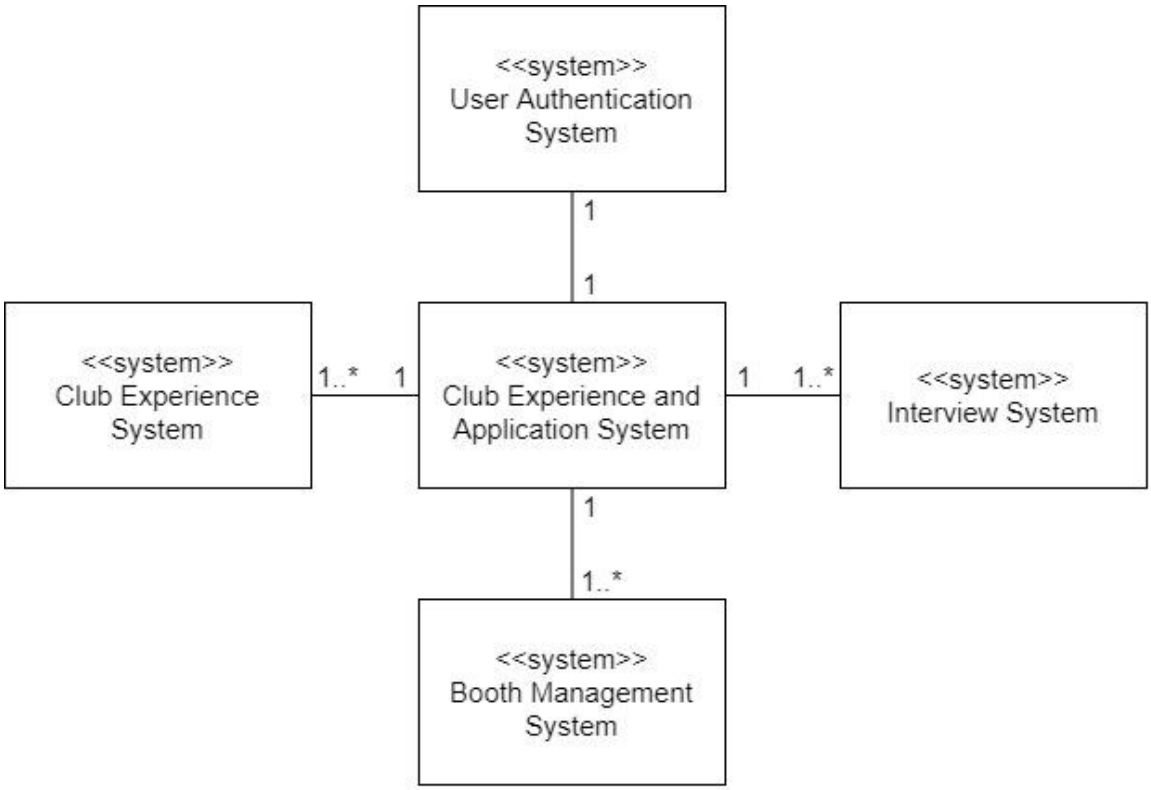

Figure 2 Overall Context Diagram

3.2.2 Sequence Diagram

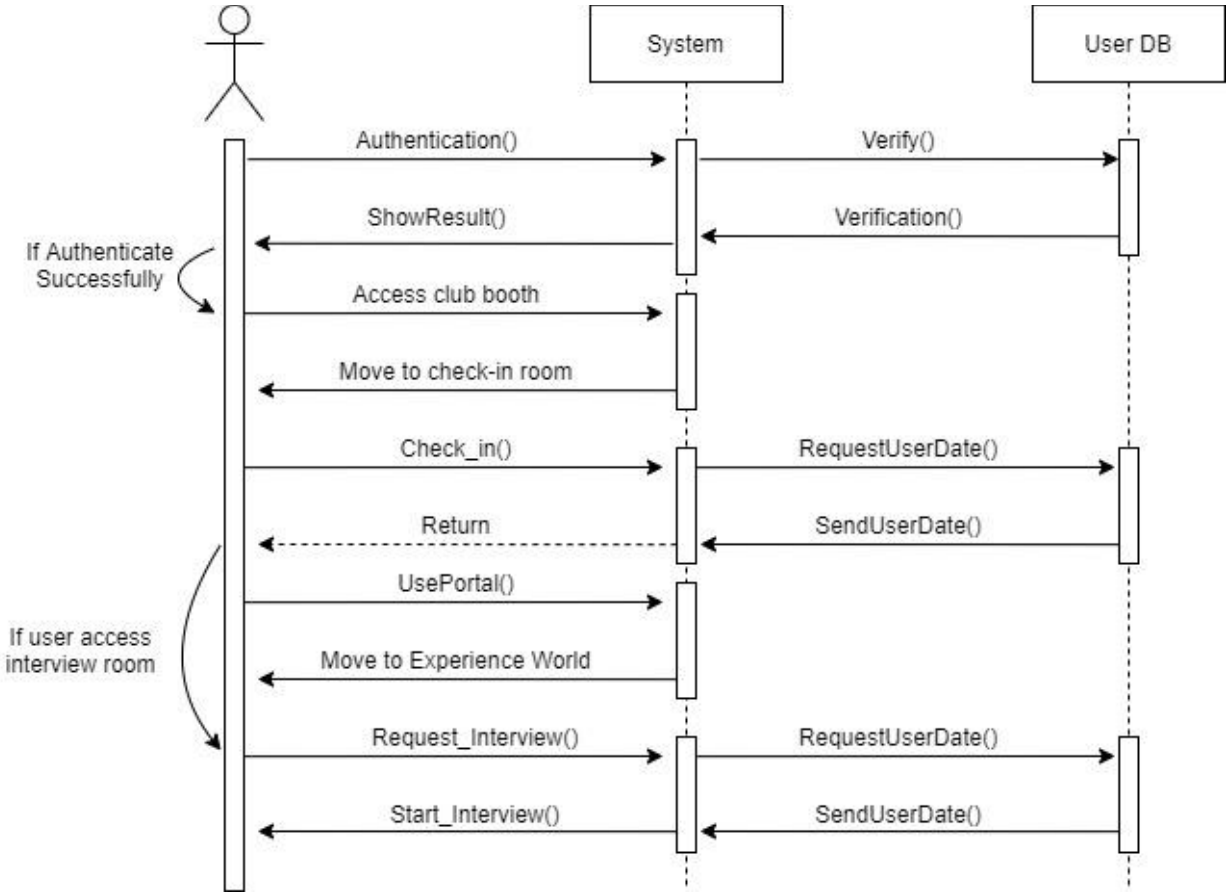

Figure 3 Overall Sequence Diagram

3.2.3 Use Case Diagram

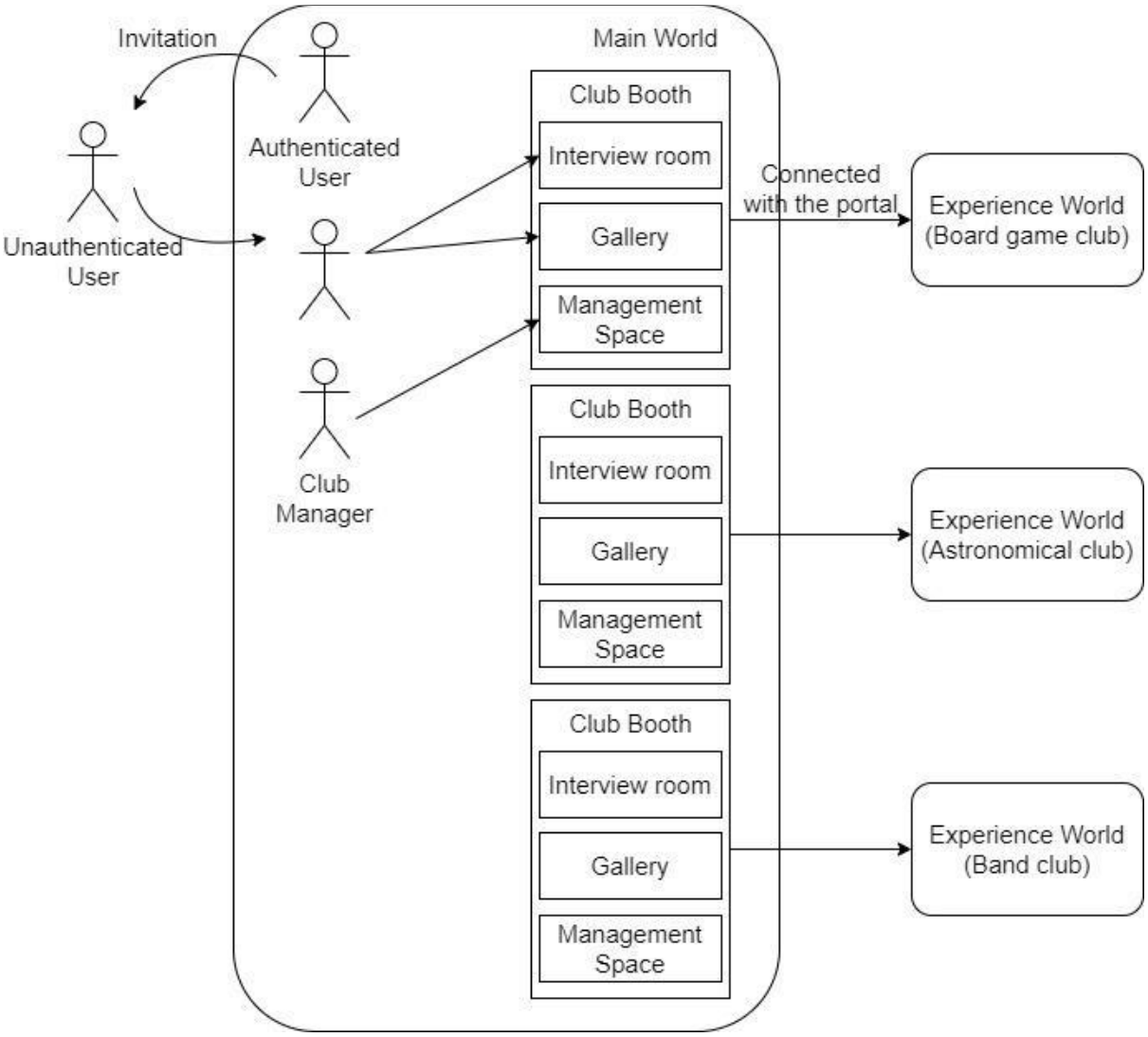

Figure 4 Use Case Diagram

4. System Architecture – Frontend

4.1 Objectives

This chapter describes user-interactive game objects' structure, attributes, and functionalities of appearance.

4.2 Subcomponents

4.2.1 Entrance

Entrance is a game object where user first encounter after world entrance. User can see their avatar by mirror object and can jump into school imitated world by getting out of this game object. Users are located inside this object at first due to VRC world object – which is position of where user first locate after world entrance.

4.2.1.1 Attributes

These are attributes of entrance game object.

1. Walls: Structure of entrance building, including mirror object
2. Mirror Button: Button to activate or deactivate mirror
3. Door: door object, including animation of opening or closing
4. Door Button: Button to open or close door
5. VRC World: Default VRC SDK prefab game object, which determine entered user's initial location.

4.2.1.2 Methods

These are methods to support functionalities of interactive objects – mirror, door button.

1. Start() : Initializing local variables of game objects with Udon script
2. setActive(): Set activeness of game object with Boolean argument
3. Interact(): Triggered method of interactive game object
4. onDoorOpen(): Triggered method of Door button by interact

4.2.1.3 Class Diagram

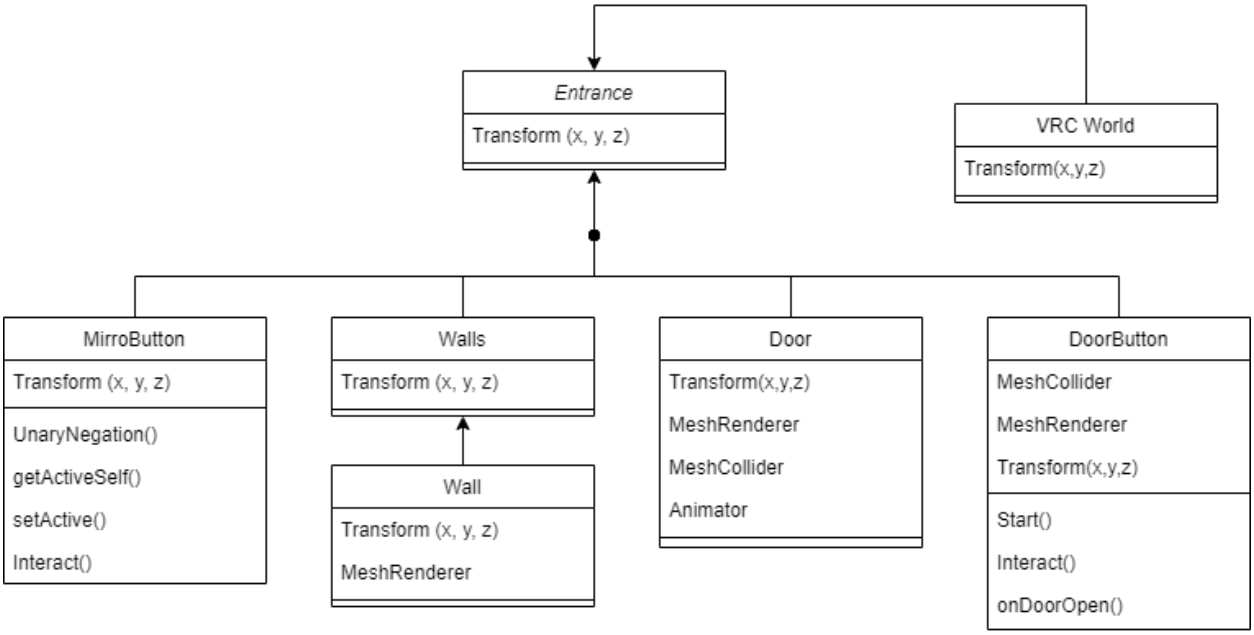

Figure 5 Class Diagram of Entrance Room

4.2.1.4 Sequence Diagram

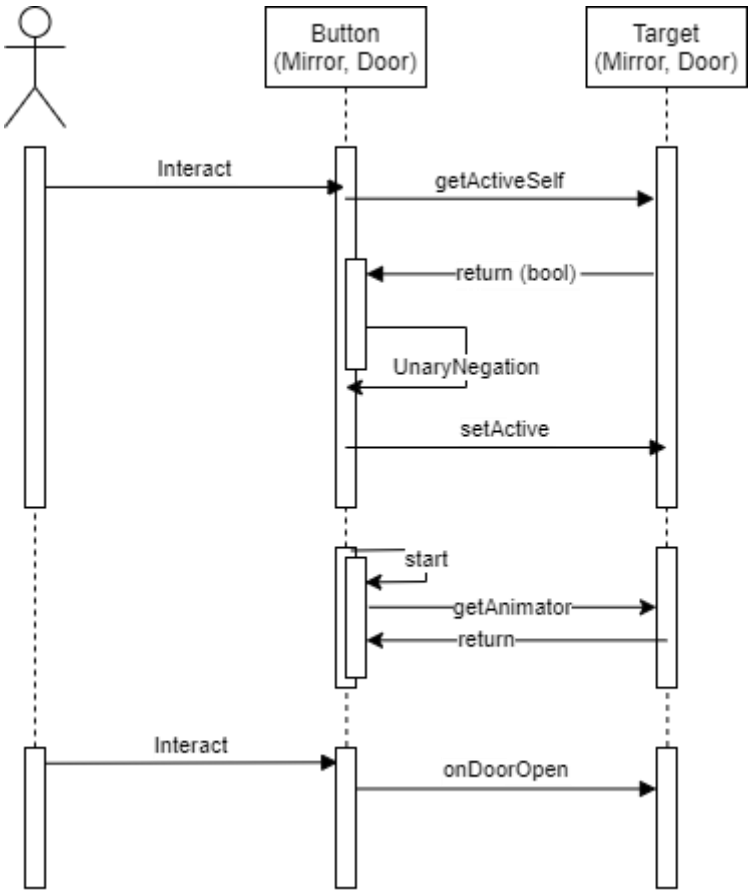

Figure 6 Sequence Diagram of Entrance Room

4.2.2 Admin Room

This game object is for world super admin. Super admin can change password of booth admin room's password, so that each booth admins – club member for booth promotion – can get desired booth admin room password by requesting to super admin. Super admin of the world is decided by the first entered user.

4.2.2.1 Attributes

These are attributes of Admin Room game object.

1. Walls: Structure of entrance building, including mirror object
2. Door: Door object of admin room
3. Admin Database: Database object to save booth admins password information
4. Password UI: UI object to enter booth admins information by super admin

4.2.2.2 Methods

These are methods to support functionalities of interactive objects in hierarchy of admin room – Admin database, UI

1. Start() : Initializing local variables including game object, Udon script
2. onClick(): event-driven method for door button, password submit button, set next button
3. onSubmit(): triggered method in database by onClick method of submit button
4. setNext(): triggered method in database by onClick method of set next button
5. getTarget(): getting target door of currently corresponding – target from booth.
6. setTarget(): setting target door for entered booth admin password in database – target from booth.
7. setPassword(): setting password of corresponding target door in database
8. setText(): changing value of text field especially for description

4.2.2.3 Class Diagram

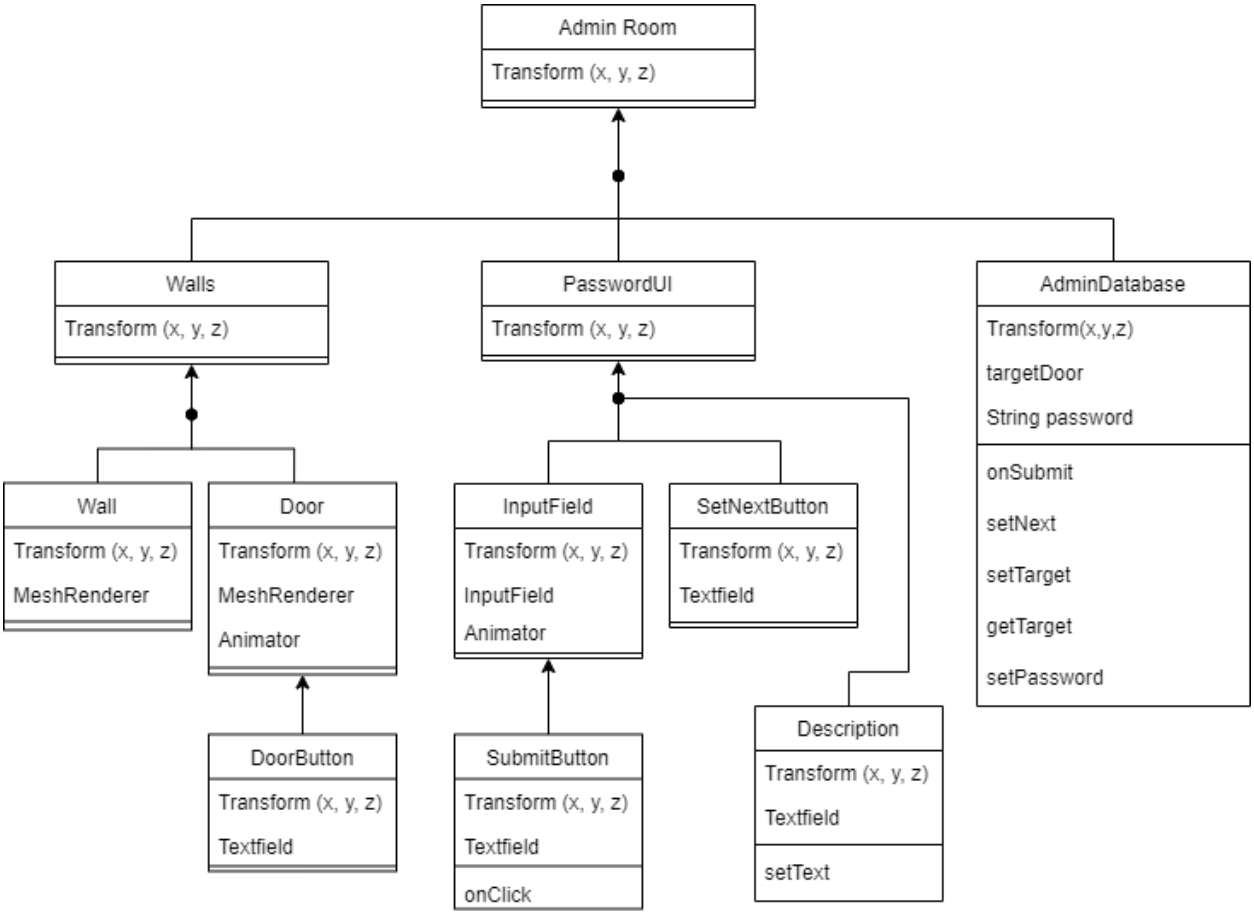

Figure 7 Class Diagram of Admin Room

4.2.2.4 Sequence Diagram

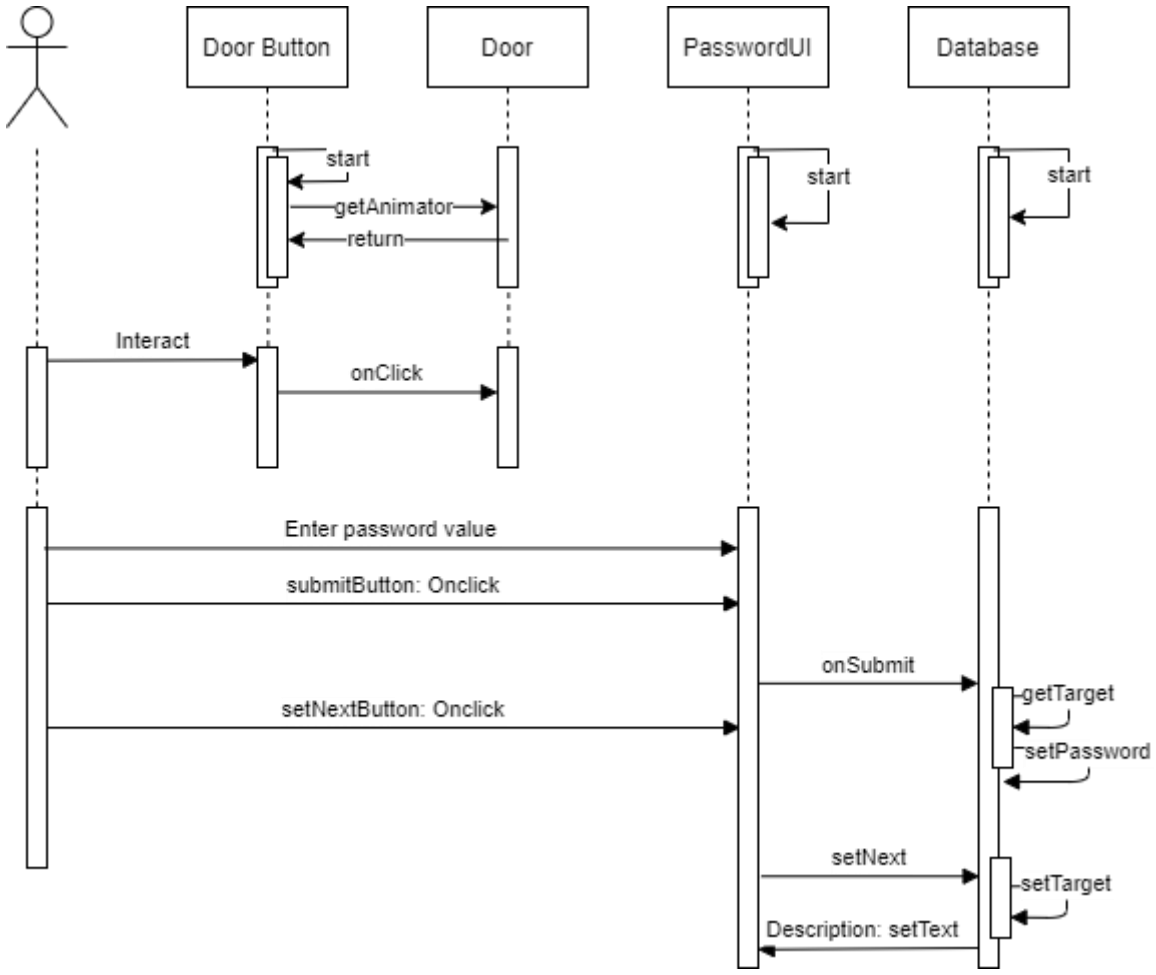

Figure 8 Sequence Diagram of Admin Room

4.2.2 Booth

Booth is where all clubs stay in and where every user can look around club activities. Manager can set Booth setting such as youtube video url or, some gallery.

4.2.2.1 Attribute

These are attribute that is used in Booth component.

1. VRC World: Default VRC SDK prefab game object, which determine entered user's initial location.
2. Walls: Structure of entrance building, including mirror object
3. Door: door object, including animation of opening or closing
4. Door Button: Button to open or close door
5. Input fields: User can write there information and manager can input youtube url.
6. Audio Source: Audio related to youtube video sound
7. Screen: Screen where Video shows up
8. Portal: Portal that can move to another world (e,g. Experience World)

4.2.2.2 Methods

These are methods that is used in Booth component.

1. Start() : Initializing local variables of game objects with Udon script
2. setActive(): Set activeness of game object with Boolean argument
3. Interact(): Triggered method of interactive game object
4. onDoorOpen(): Triggered method of Door button by interact
5. playYoutube(): Screen shows youtube video coworking with audio source.

4.2.2.3 Class Diagram

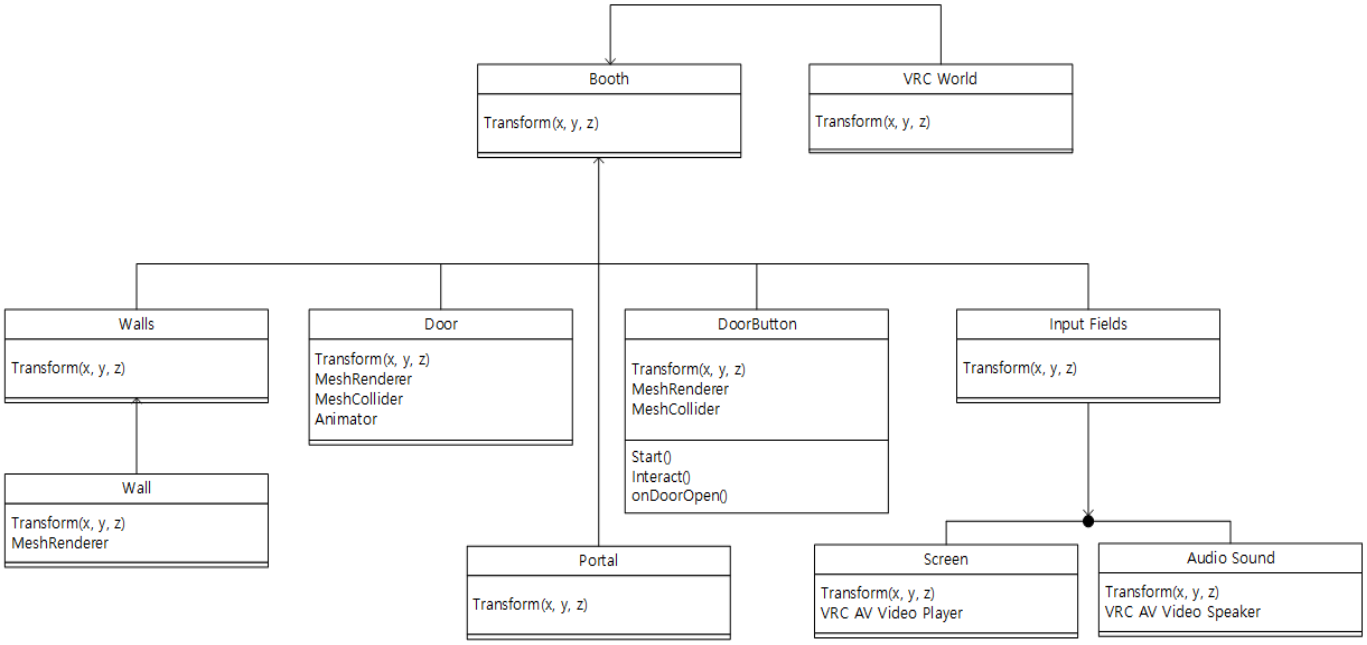

Figure 9 Class Diagram of Booth

4.2.2.4 Sequence Diagram

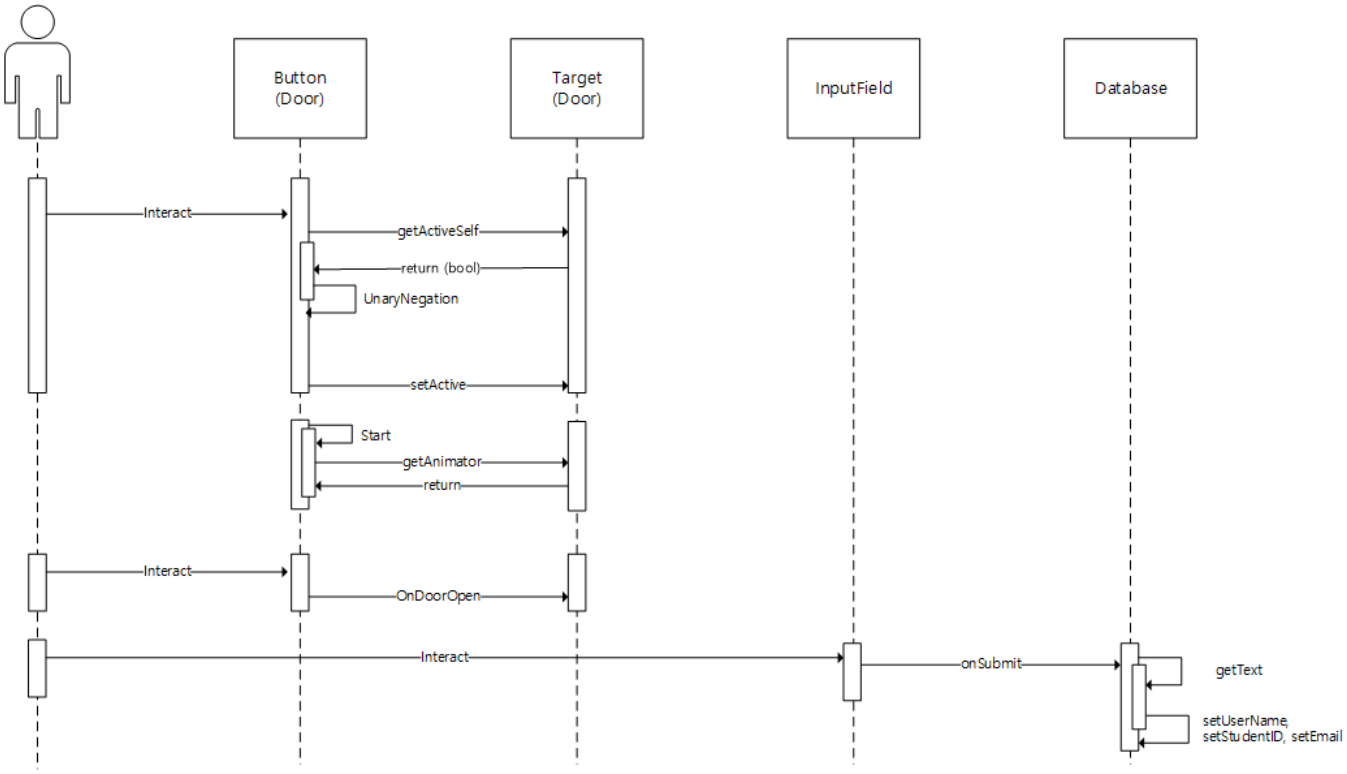

Figure 10 Sequence Diagram of Booth

4.2.3 Othello Table

4.2.3.1 Attributes

These are attributes of Othello table

1. Table Structure: Game object mainly targeting to mesh and render to show game table
2. Othello Board: Game object of Othello board, showing appearance and control game service
3. Stone: Game object of Othello, showing appearance of Othello game status
4. Reset, Start Button: Buttons to start or reset game

4.2.3.2 Methods

These are methods to support functionalities of interactive objects – mirror, door button.

1. setByStartButton(): Method called at Othello board, triggered by start button interaction. This is called especially when two start buttons are all interacted.
2. setByResetButton(): Method called at Othello board, triggered by reset button interaction. This initializes game status.
3. setTurn(): Method called at Othello board, triggered by stone interaction or game start. It processes the previous player's action if exists – putting stone by interacting with stone.
4. setAvailability(): Method called at Othello board, triggered by setTurn method, getting the available stone positions for next turn player.
5. setInteractiveness(): Setting interactiveness of stone. This is triggered after turn setting from Othello board.
6. setActiveness(): Setting activeness of stone, start button, reset button. This is triggered by turn setting from Othello board or button interaction.
7. setMaterial(): Setting material of stone, to use texture of white or black. This is triggered by turn setting from Othello board.
8. Interact(): Interact method of stone, start button, reset button. This method triggers other methods of Othello related objects to set appearance.
9. getStatus(), setStatus(): setter or getter of status board. This is required as game status is in bitmap structure, as syncing variables processing are expensive.

4.2.3.3 Class Diagram

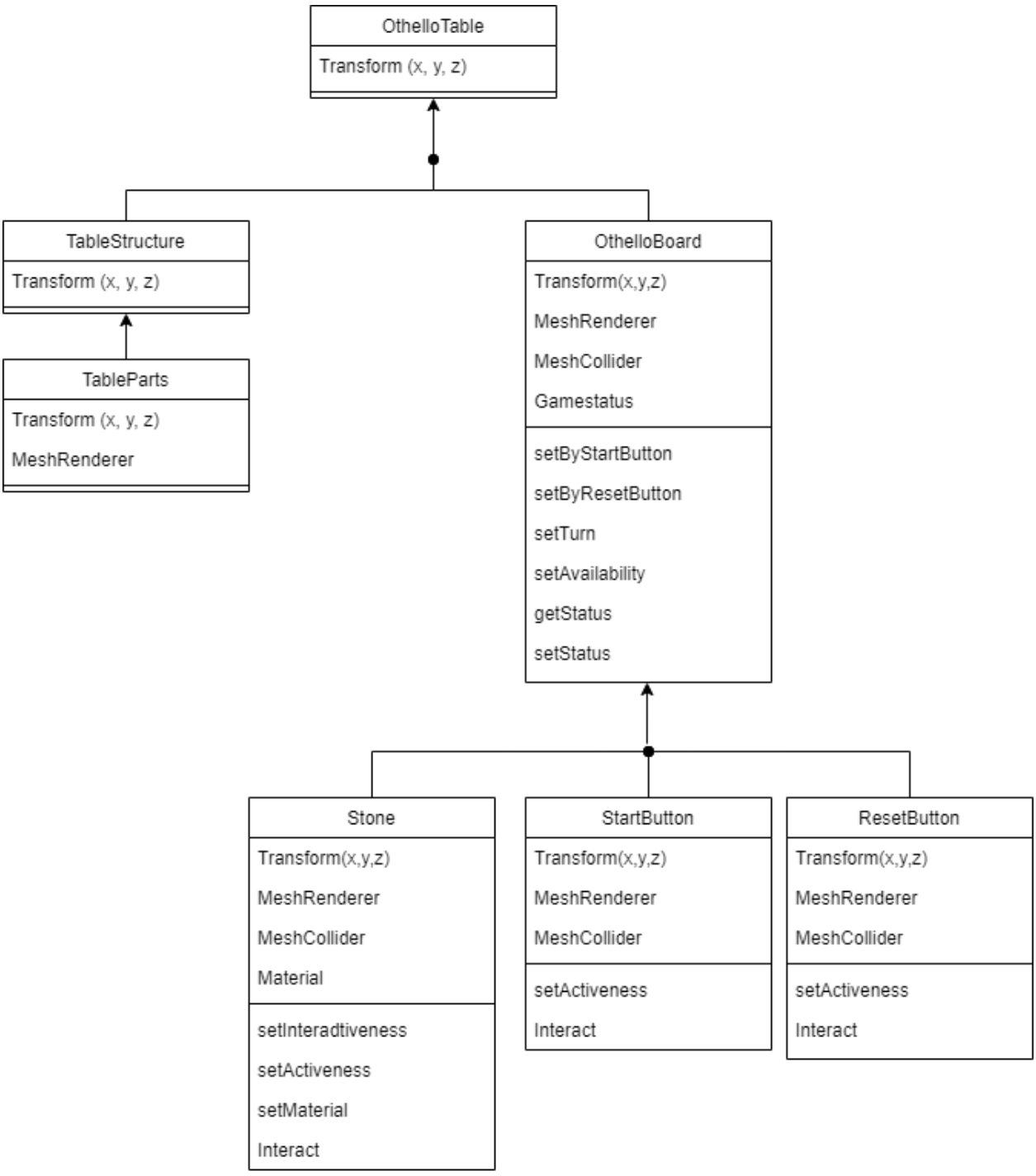

Figure 11 Othello Table Class diagram

4.2.3.4 Sequence Diagram

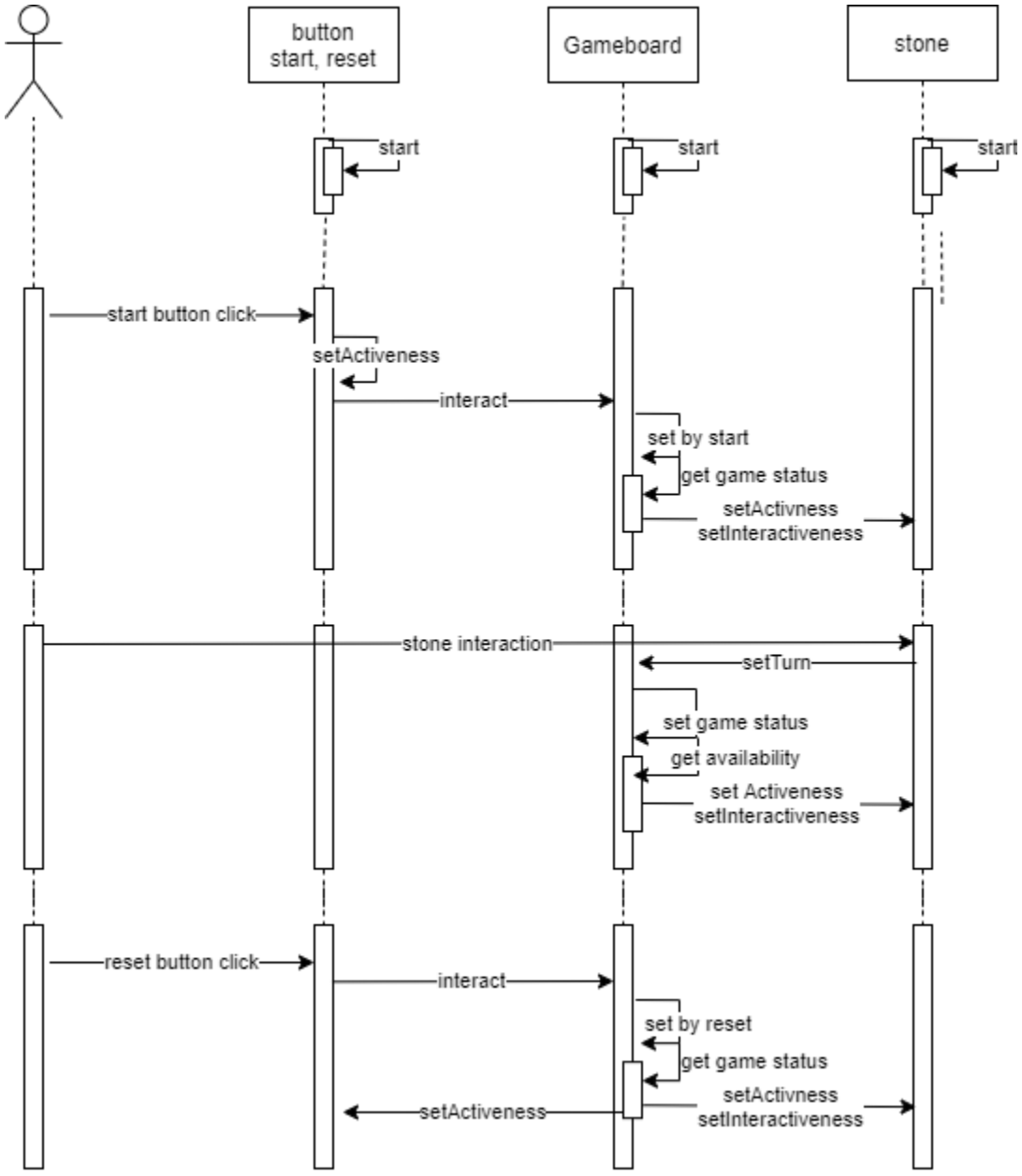

Figure 12 Othello table Sequence Diagram

4.2.4 Astro

Experience World where user can experience Astro club activity.

4.2.4.1 Attribute

1. VRC World: Default VRC SDK prefab game object, which determine entered user's initial location.
2. Walls: Structure of entrance building, including mirror object
3. Star Frame: Star shape or image is shown on part of frame
4. Button: Button that can show more specific star information related to star frame

4.2.4.2 Methods

1. Start() : Initializing local variables of game objects with Udon script
2. setActive(): Set activeness of game object with Boolean argument
3. Interact(): Triggered method of interactive some object
4. showContent(): When clicking the button, trigger more specific information of star.

4.2.4.3 Class Diagram

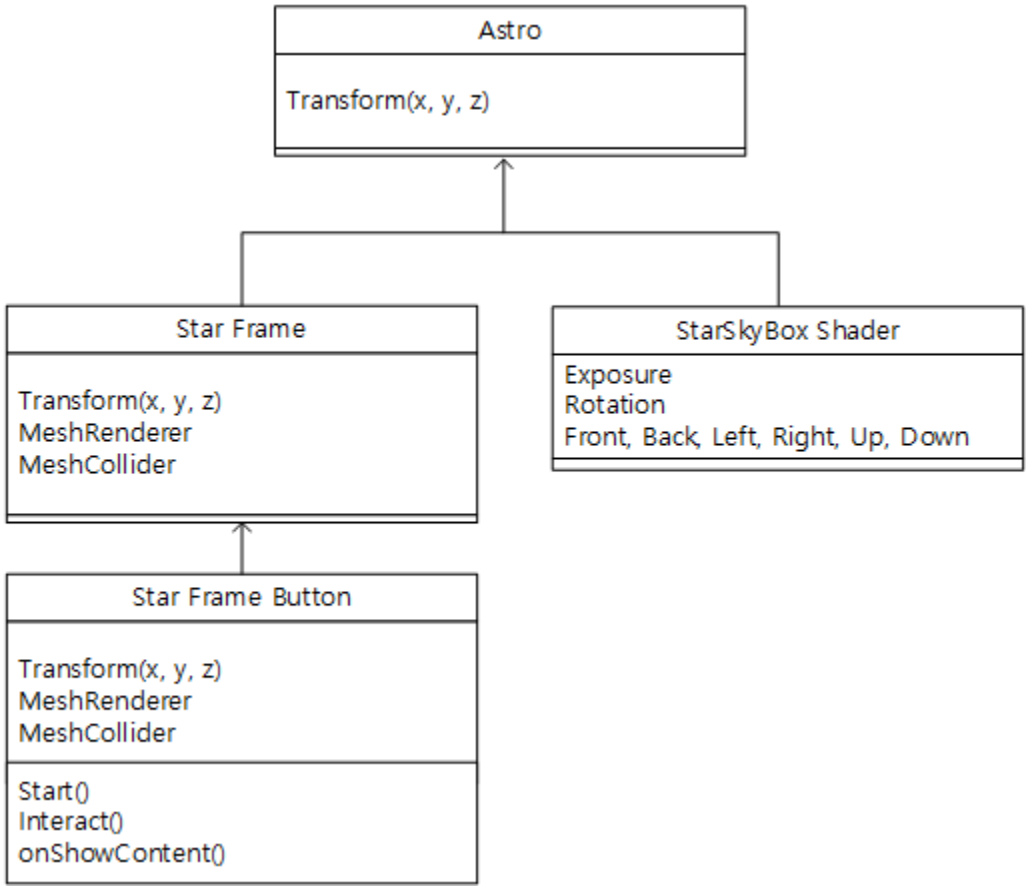

Figure 13 Class Diagram of Astro

4.2.4.4 Sequence Diagram

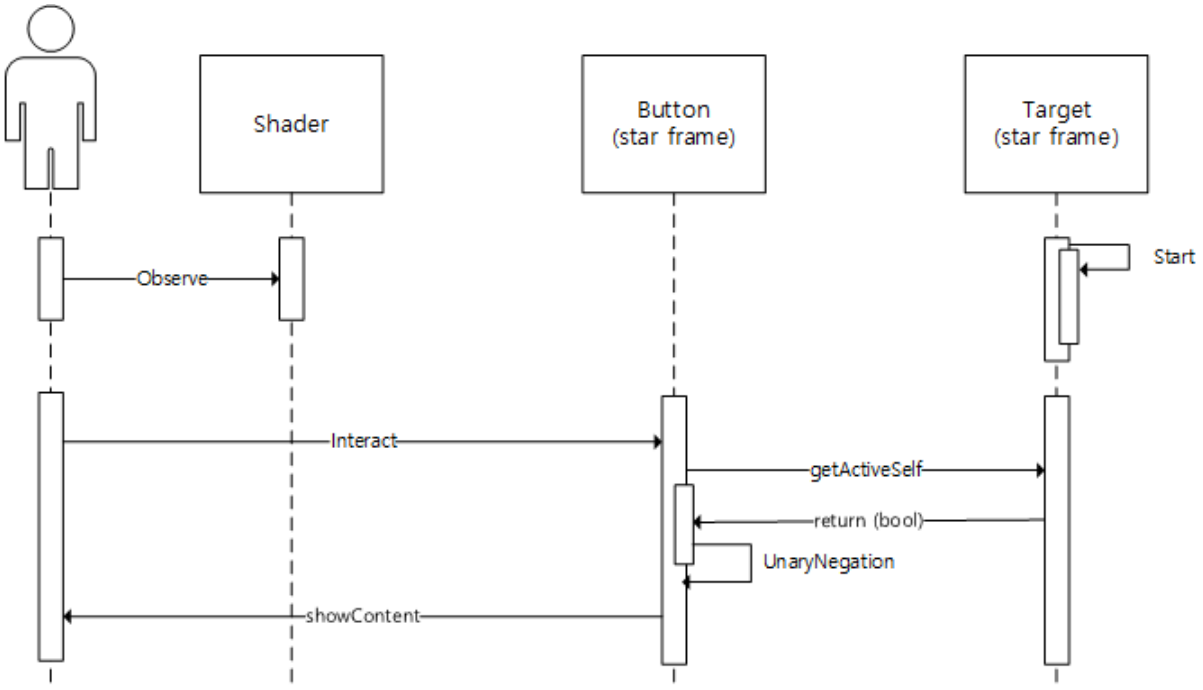

Figure 14 Sequence Diagram of Astro

4.2.5 Band

4.2.5.1 Attributes

1. VRC World: Default VRC SDK prefab game object, which determine entered user's initial location.
2. Walls: Structure of entrance building, including mirror object
3. Stage: Structure where band musician can play instruments
4. Chairs: Structure where audience sit on.
5. Light: Structure that can light musician.
6. Light Button: Buttons to control light movement

4.2.5.2 Methods

1. Start() : Initializing local variables of game objects with Udon script
2. setActive(): Set activeness of game object with Boolean argument
3. Interact(): Triggered method of interactive instrument object
4. moveLight(): Lights can show another spot to highlight musician

4.2.5.3 Class Diagram

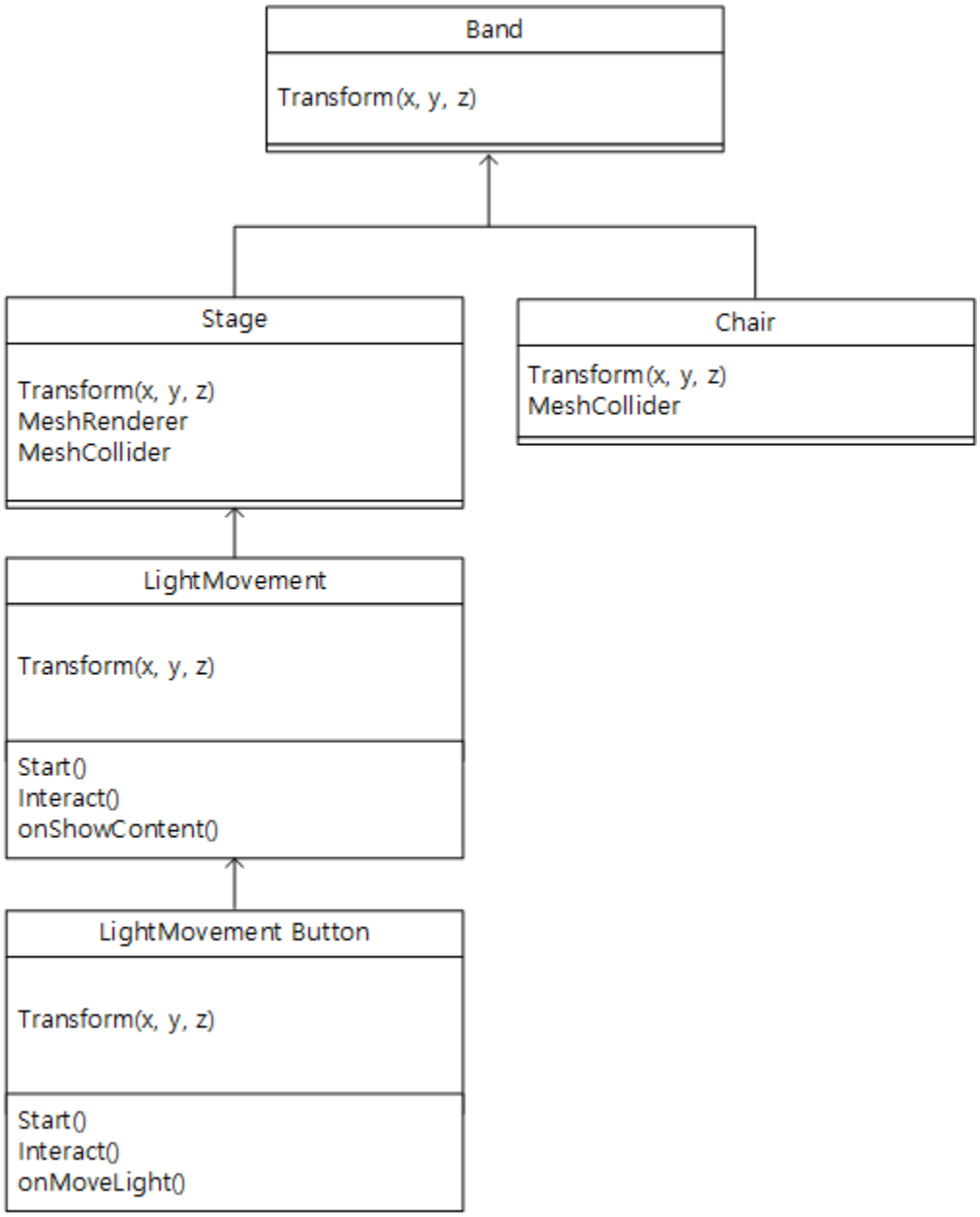

Figure 15 Class Diagram of Band

4.2.5.4 Sequence Diagram

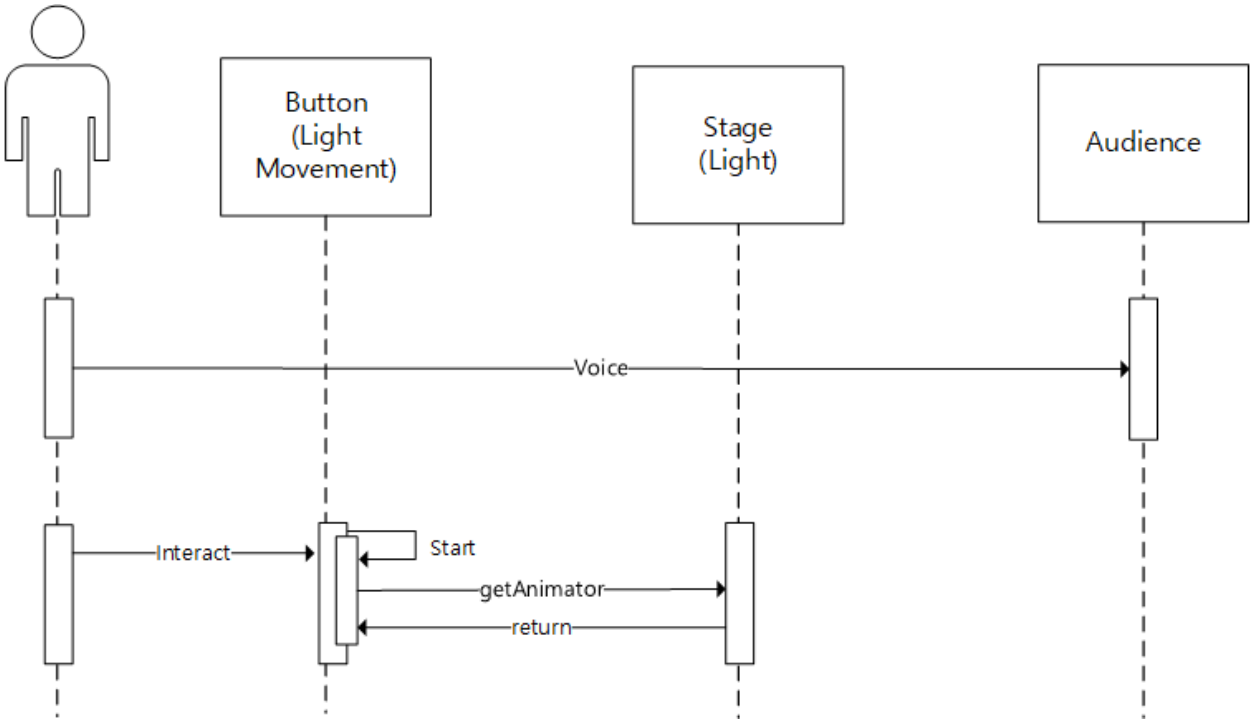

Figure 16 Sequence Diagram of Band

5. System Architectures – Backend

5.1 Objectives

This chapter describes the structure of the backend system.

5.2 Overall Architecture

The overall architecture of the backend system is like above. Since VR Chat limits networking to the outside of the platform, the backend structure is not complicated compared to the frontend system. The booth world contains club application and user authorization functionalities. The data are stored in the game object that serves as a primitive database and would be referenced by the club and world managers.

5.3 Subcomponents

5.3.1 Club application

5.3.1.1 Class Diagram

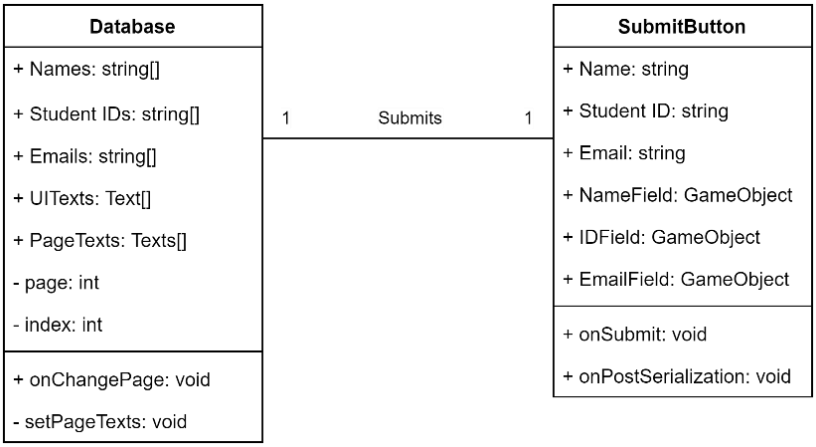

Figure 17 Class Diagram – Club application

- Class Description
  1. **Club application** : It is an interface that handles the user information submission to the club. The database exists as a class to compensate the networking restriction. When a user applies to a club, the system validates the user inputs and submits them to the database. The database is directly connected to the admin UI and the data is displayed through it. Each club would have the classes to manage their application data.

5.3.1.2 Sequence Diagram

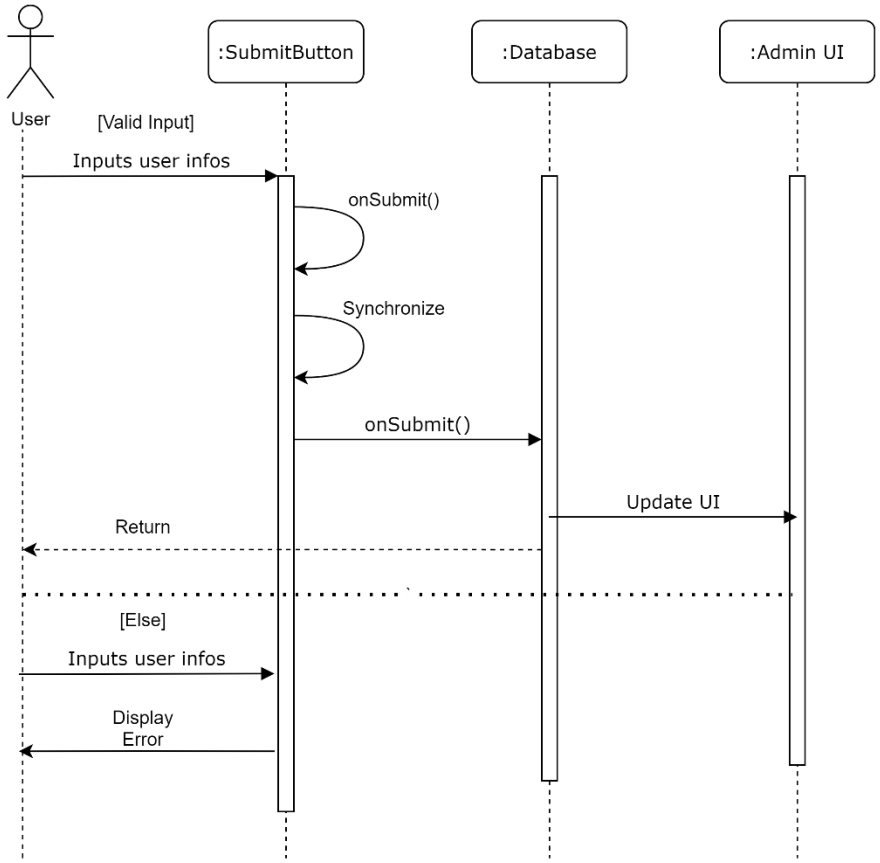

Figure 18 Sequence Diagram – Club Application

5.3.2 User Authorization

5.3.2.1 Class Diagram

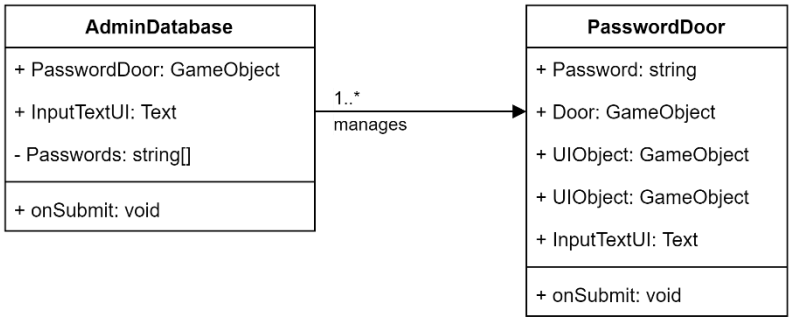

Figure 19 Class Diagram – User Authorization

- Class Description
    1. **User Authorization:** There is a world master who opens the world in VR Chat. The world master manages each club admins by setting password to the club admin room. The passwords would be given beforehand. Then, the club admins can access to the admin room with the password.

5.3.2.2 Sequence Diagram

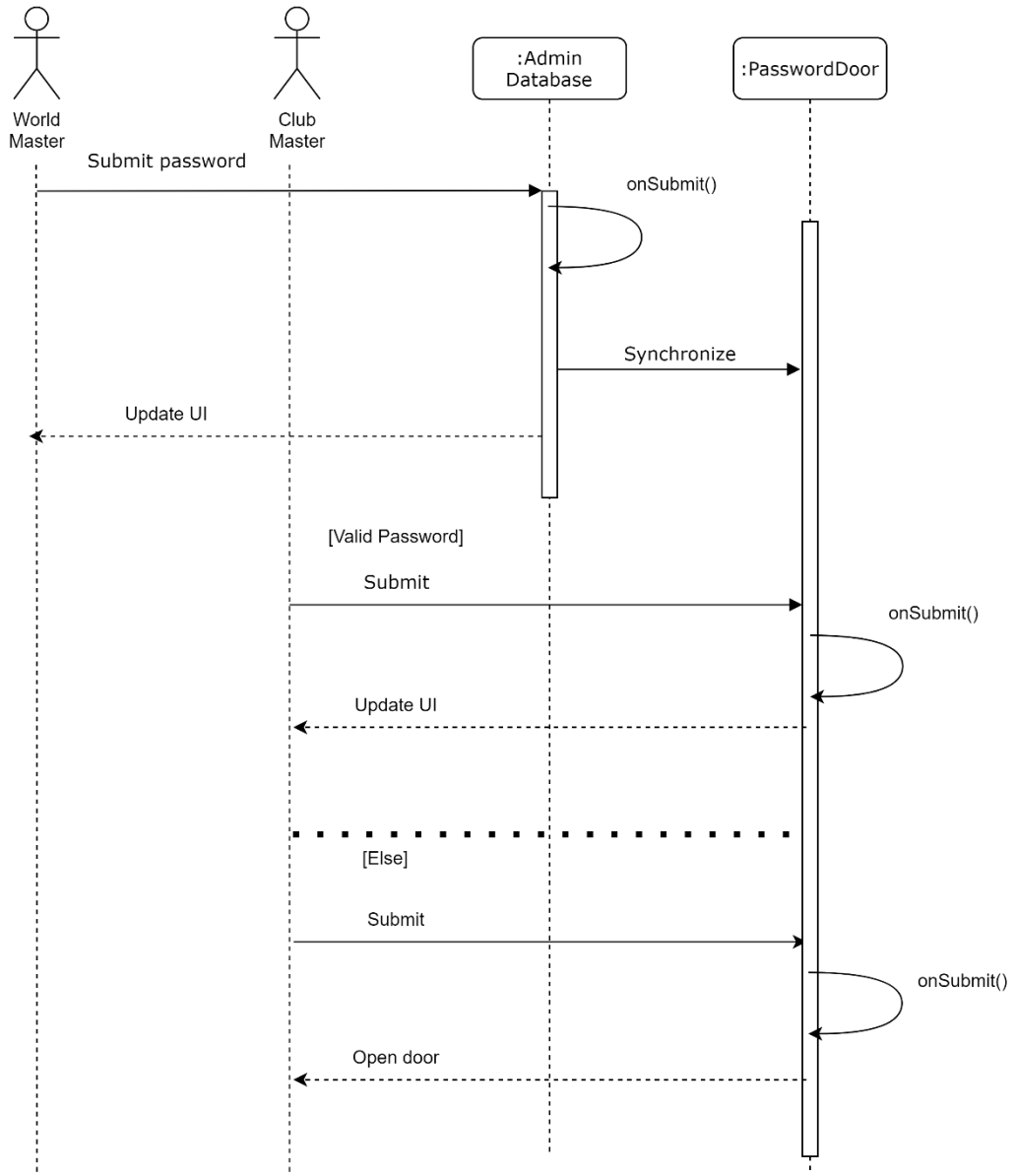

Figure 20 Sequence Diagram – User Authorization

6. Protocol Design

6.1 Objectives

This chapter describes the protocol used for the system interaction. Since the system runs on the VR chat, there is not a protocol to send data. However, the variables would be synced through networking inside the world. The synchronization is important since the users might see different things unless the synchronization is properly done. Thus, this chapter will describe how the variables are synchronized.

6.2 Synchronization

The Udon APIs, which supports programmable graphs and scripts for VR chat, introduce synchronization. The synchronization can be done manually or automatically through sync attribute of the value or network event. The sync attribute forces every player in the world to have the same value with the owner of the object. There are two types of the sync attribute: manual and continuous. The manual sync mode does not synchronize the variables unless the synchronization is requested. The continuous mode keeps synchronization in a lazy way, which can be delayed for short time. The owner of an object is initially set to the world master who opened the world for the first time. The owner can be handed over to other player with scripts. Since the variables are synced with the owner of the object, it can be used to authorization of a user.

The network event is another way to synchronize the variables. Unlike the sync attribute, the synchronization can be done with users who are not the owner of the object. The network event is occurred when a user satisfies some conditions that are scripted. The two methods should be adequately used for the purposes and network synchronization.

7. Database Design

7.1 Objectives

This section describes the objectives of the system. Since any off-the-shelf database system is not usable, the primitive database is implemented to support minimum functionality set. Thus, there is not any join or complicated searching operation. Also, the system is mainly implemented through frontend system, there are few entities (or table) in the database.

7.2 ER Diagram

Since there is not any available join operation and other relationships, there is no relationships between entities.

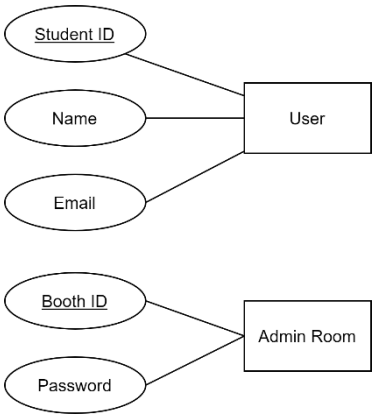

Figure 21 Entity-Relationship Diagram

7.2.1 Entities

7.2.1.1 User

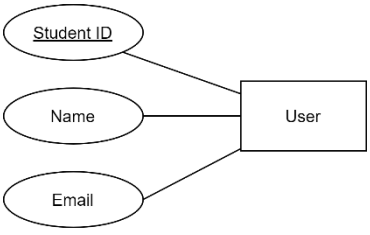

Figure 22 User Entity

User entity represents the users in the VR chat world. It consists of student id, name and email of the user. The student id is a primary key.

7.2.1.2 Admin Room

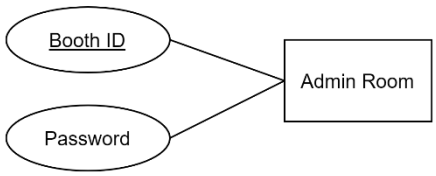

Figure 23 Admin Room Entity

Admin room entity represents the admin rooms in the VR chat world. It consists of booth ID and password. The booth ID is a primary key.

8. Testing Plan

8.1. Objectives

This chapter will explain the testing plan. Testing to proceed is divided into development testing, release testing, and user testing.

8.2. Testing Policy

8.2.1. Development Testing

Development testing includes all test activities performed by the team developing the system. It includes unit tests to test individual objects or object classes, testing for components created by integrating multiple independent units, and testing for systems created by integrating some or all of the components.

At this stage, static code analysis, peer code review, and unit testing are performed. Through this process, we need to ensure that the system developed meets the required performance, stable behavior without failure, and that there is no security problem.

8.2.1.1. Performance

In our system, it is important to support users to have experience with the club as positive as possible, so as specified in the requirements specification, the resource load for all

activities required in the experience process must be completed within 5 seconds. The list to be checked and corresponding test plans are as follows.

1. Image and video loading: 2 types of images of 500KB or less, 1MB or more, respectively, and 2 types of 1MB or more.
2. Outside world movement for club experience: 3 example worlds (Othello, Astronomical Observation, Band)

8.2.1.2. Reliability

Individual unit testing should be thoroughly conducted so that the system can operate safely without failure, and repeatedly checked to prevent problems in the integration process. In particular, it is necessary to focus on checking whether images and videos uploaded by individual club managers are displayed normally and do not store applications in the application list of other clubs or omitted.

8.2.1.3. Security

Security issues are particularly important because our system receives and stores applicants' personal information during the application process. Saved applications should be ensured that malicious users or innocent users who accidentally accessed them cannot see them. To this end, we must identify security vulnerabilities, supplement them, and record them as reports by repeated attempts to invade the system assuming malicious users after final system integration.

8.2.2. Release Testing

The main goal of Release testing is to ensure that the corresponding version of the system is sufficient for customers to use. Prior to release, we need to go through the process of testing whether the system of the version we intend to release operates normally in the recommended specifications specified in the requirements specification. It is desirable that a separate team that is not involved in the development of the system is in charge of this testing, but due to conditions, the teams that developed each sub-system test other sub-system to see if there are any defects and if the sub-system meets the requirements.

8.2.3. User Testing

Through the User testing step, we can receive input from actual users and receive advice on the system. Even after the release testing is performed, user testing is an essential

process.
We proceed with user testing assuming a realistic situation. Beta tests are conducted on 20 users, the minimum concurrent accessible condition specified in the requirements specification, and user reviews are collected and recorded.

8.2.4. Testing Case

The testing case is based on the test plan for development testing described in 8.2.1. Testing for performance, reliability, and security is carried out sequentially in the unit and integration stages, and the same content is carried out by different testers, evaluations, and cross-verification in the Release testing and user testing stages, increasing the reliability of the system.

9. Development Plan

9.1. Objectives

This chapter will describe the technologies and environments to be used for the development of the system.

9.2. Frontend Environment

9.2.1. VRChat

Figure 24 VR Chat logo

It is virtual reality-oriented voice chat software developed by VRChat Inc. and provided through the Steam platform. Our system operates on VRChat's world, and activities such as world composition and connection for external experiences, and conversations between club managers and users are implemented and provided based on functions provided by VRChat.

9.2.2. Unity

Figure 25 unity logo

It is a game engine developed and provided by Unity Technologies. It helps develop 2D and 3D games targeting various platforms such as Windows PC, Linux, Mac OS, Android, and iOS. Since world production in VRChat is done through Unity, we will mainly utilize Unity for system development.

In particular, for VRChat World production, Udon, a node-type programming language developed by the VRChat development team, should be used, and we develop it in a more familiar C# environment using Udon Sharp, a compiler that compiles C# into Udon.

9.3. Backend Environment

9.3.1. Github

Figure 26 GitHub logo

It is an open-source web service that supports Git storage hosting, a distributed version management tool. We use GitHub in the process of software version management, unit development, and integration.

9.3.2. Unity

As explained in Section 5.2. above, VRChat restricts networking outside the platform, so we will create game objects in Unity that act as raw databases to store support lists for clubs and allow licensed club managers to access them.

9.4. Constraints

Other details other than those mentioned in this document shall be designed and implemented at the discretion of the developer. In that case, the following must be observed.

1. Use a technology that has been verified and widely used.
2. Avoid using technologies and software that require purchasing a separate license or paying a fee as much as possible.
3. Decide in the direction of improving the overall performance of the system.
4. The future scalability of the system must be considered.
5. Faithfully provide annotations for ease of maintenance.
6. It is necessary to faithfully support the recommended specifications specified in the - requirement specification.

9.5. Assumptions and Dependencies

All systems in this document were created under the assumption that they were designed, implemented, and executed based on VRChat. All content is based on the latest version of VRChat and may not apply.

10. Supporting Information

10.1 Software Deisign Specification

This software requirements specification was written in accordance with the IEEE Recommendation (IEEE Recommended Practice for Software Requirements Specifications, IEEE-Std-830).

10.2 Document History

Date	Description	Writer
11/9	write part 5	Junyoung Lee
11/10	Write part 1, 2	Dongwon Kim
11/12	Write part 4.1, 4.2	Doyeol Kim
11/14	Write part 4.2	Minjie Kim
11/15	write part 3	Chung Ju Won
11/15	write part 6, 7, 8, 9	Myeongmin Kim
11/17	modify whole	Dongwon Kim
11/18	modify whole	Minje Kim

Table 1Document History