Space for School Festival

Software Design Specification

2021.11.21

Introduction to Software Engineering 41

TEAM 16

Team Leader Hyejin Yoon
Team Member Gyeongun Kang
Team Member Jeongbok An
Team Member Seunggu Kang
Team Member Soo Namgoong
Team Member Yuyeong Seo

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Preface

This chapter contains the readership information, readership, scope, objective of this document and the document structure of this Software Design Document for this project, Space for School Festival.

1.1. Readership

This Software Design Document is divided into 7 sections with various subsections. The structure of the Software Design Document can be found as listed below, in the Document Structure subsection of this SDD. In this document, Team 16 is the main reader. Additionally, professors, TAs, team members in the Introduction to Software Engineering class and all relevant stakeholders can be the main readers. Developers should implement the software according to the software design described in this document, and stakeholders can use this document to confirm that this project meets the requirements.

1.2. Scope

This Design Specification is to be used by Software Engineering and Software Quality Engineering as a definition of the design to be used to implement the Space for School Festival where users can participate in performances, watching, events, and booth activities in VRChat.

1.3. Objective

The primary purpose of this Software Design Document is to provide a description of the technical design aspects for this project, Space for School Festival. This document describes the software architecture and software design decisions for the implementation of Space for School Festival. It also provides an architectural overview of the system to describe different aspects of the system. Furthermore, it specifies the structure and design of some of the modules discussed in the Software Requirement Specification document and displays some of the use cases that have been transformed into sequential and class diagrams which show how the programming team would implement the specific module. The intended audience of this document is, but not limited to, the stakeholders, developers, designers, and software testers of this project, Space for School Festival.

1.4. Document Structure

- **1. Preface**: this chapter describes readership, scope of this document, object of this system, and structure of this document.
- **2. Introduction**: this chapter describes several tools used for this document, several diagrams used in this document and the references, and object of this project.
- **3. Overall System Architecture**: this chapter describes overall architecture of the system using context diagram, sequence diagram, and use case diagram.
- **4. System Architecture - Frontend**: this chapter describes architecture of the frontend system using class diagram and sequence diagram.
- **5. Testing Plan**: this chapter describes testing plan for our system.
- **6. Development Plan**: this chapter describes which tools to use to develop the system, constraints, assumption, and dependencies for developing this system.
- **7. Supporting Information**: this chapter describes baseline of this document and history of this document.

2. Introduction

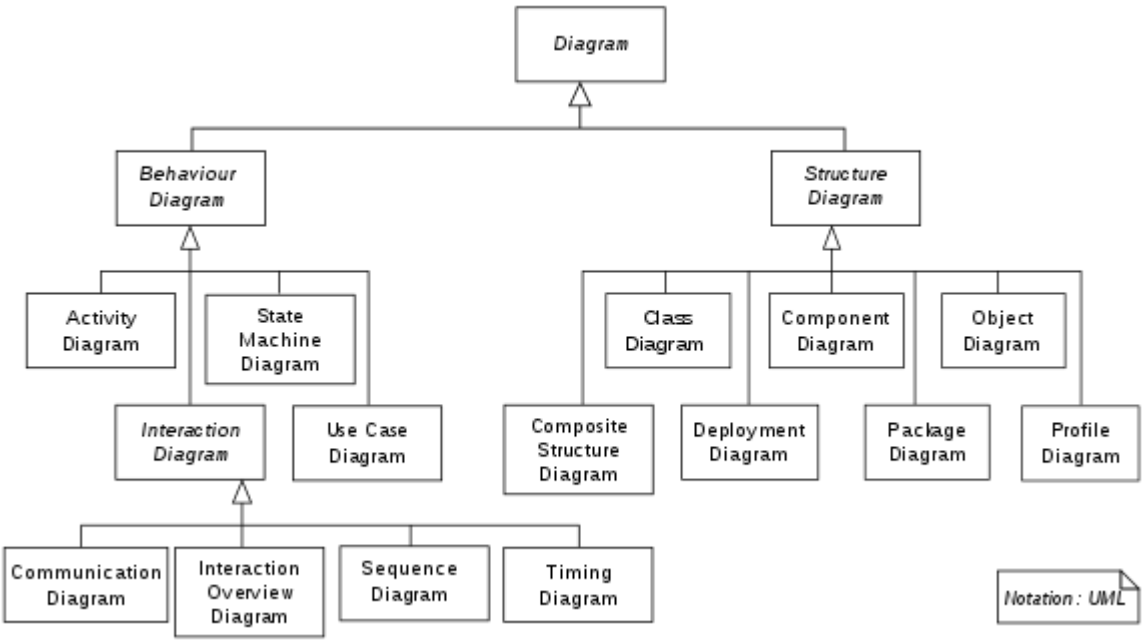
The project is to design and implement VRChat World so that people who cannot enjoy offline school festival due to COVID-19 can enjoy online festival in VRChat. The host can hold the festival by opening the World, and users can enter the World and participate in performances, watching, events, and booth activities. Also, users can interact with other users and objects through functions implemented in the World in addition to VRChat internal functions. This design document presents the designs used or intended to be used in implementing the project. The designs described, follow the requirements specified in the Software Requirement Specification document prepared earlier for the project.

2.1. Objectives

In this chapter, we describe the various tools and diagrams which we have applied to this project in the design phase.

2.2. Applied Diagrams

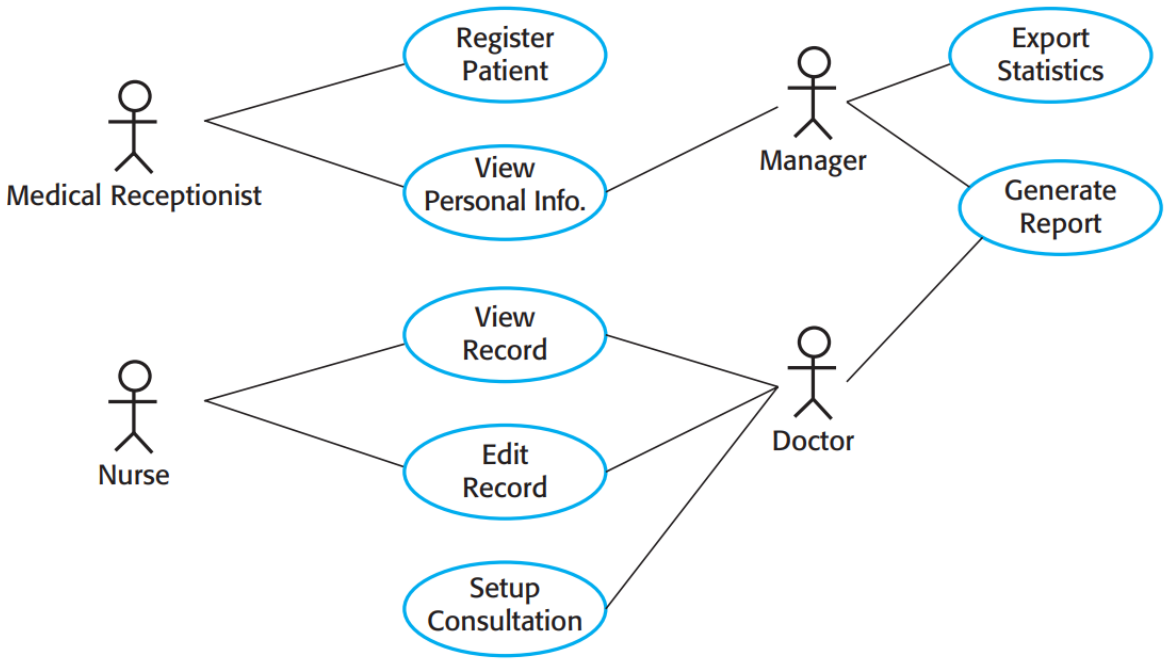
2.2.1. UML



[Figure 1] Hierarchy of UML 2.5 Diagrams

System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system. It has generally come to mean representing the system using graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML). The UML is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system. UML has many diagram types and supports the creation of many different types of system model. It provides a powerful and logical expression for modeling and shows the entire structure of the system in an easy-to-understand form through diagrams. This enables efficient communication between developers, and makes it easy to understand, review and modify the modeling structure. Also, UML can be applied regardless of the project scale, and it provides rich analysis and design tools based on diagram for object-oriented software development.

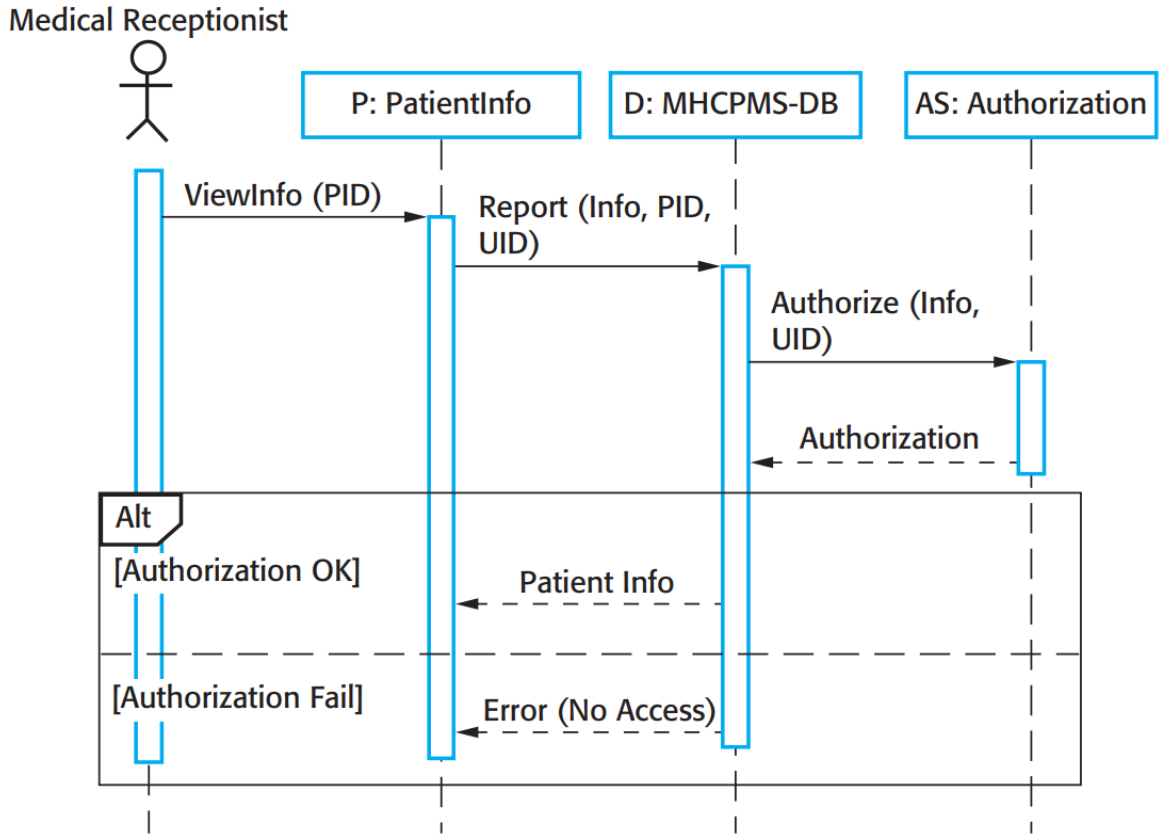
2.2.2. Use Case Diagram



[Figure 2] Use cases for the MHC-PMS

A Use case Diagram of UML is a graphical depiction of a user's possible interactions with a system. Use cases identify the individual interactions between the system and its users or other systems. Each use case should be documented with a textual description. These can then be linked to other models in the UML that will develop the scenario in more detail. The set of use cases represents all possible interactions that will be described in the system requirements. Use cases are documented using a high-level use case diagram because it is difficult to provide detailed information about all possible cases between the system and the users with only use cases. In Use case Diagram, actors in the process, who may be human or other systems, are represented as stick figures. Each class of interaction is represented as a named ellipse. Lines link the actors with the interaction. Optionally, arrowheads may be added to lines to show how the interaction is initiated. A Use case Diagram shows various use cases and different types of users. The main purpose of Use case Diagram is to visualize the functional requirements of the system and it allows customers and developers to coordinate their opinions on the requirements.

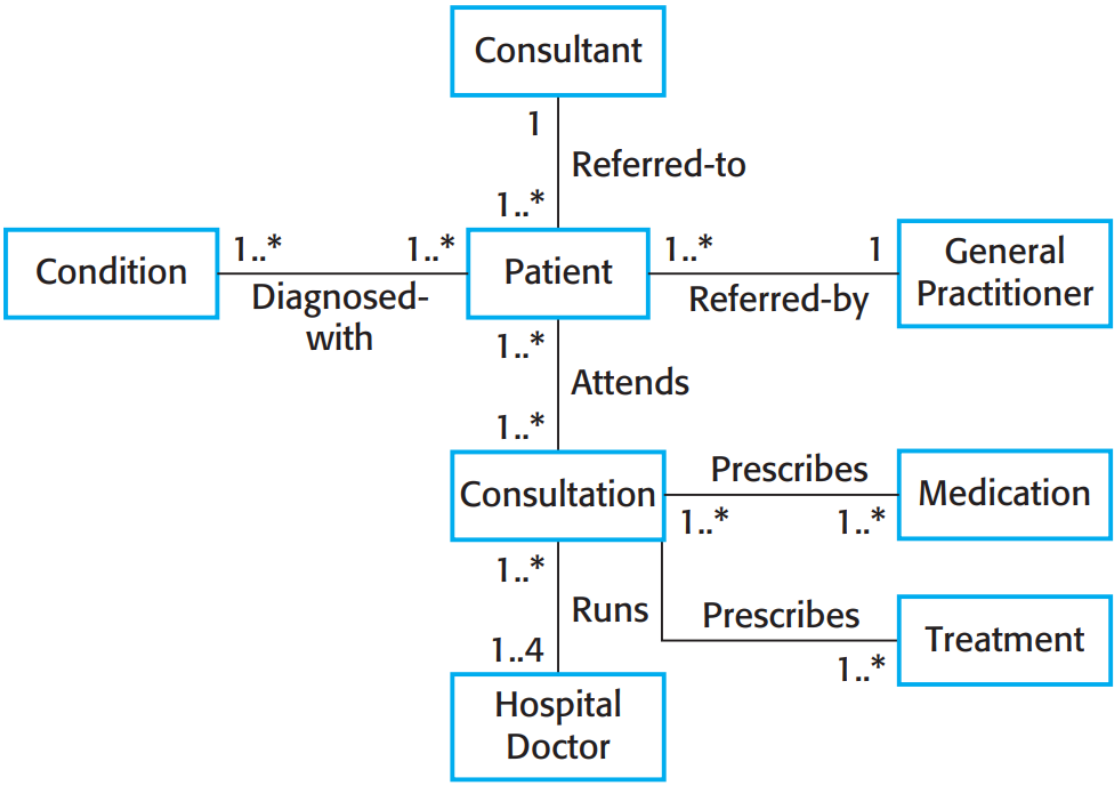
2.2.3. Sequence Diagram



[Figure 3] Sequence diagram for View patient information

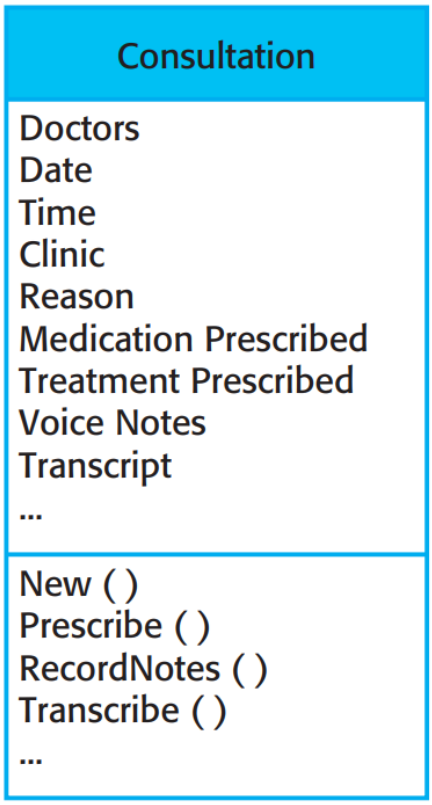
Sequence diagrams in the UML are primarily used to model the interactions between the actors and the objects in a system and the interactions between the objects themselves. As the name implies, a sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance. That is it shows objects interactions arranged in time sequence The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these. Interactions between objects are indicated by annotated arrows. The rectangle on the dotted lines indicates the lifeline of the object concerned. You read the sequence of interactions from top to bottom. The annotations on the arrows indicate the calls to the objects, their parameters, and the return values. Sequence Diagram can be used to model dynamic aspects of the system by depicting data exchanges, message sequences, etc. between each object for a specific use case or part of a specific use case, and it is useful for viewing the flow of the system at runtime.

2.2.4. Class Diagram



[Figure 4] Classes and associations in the MHC-PMS

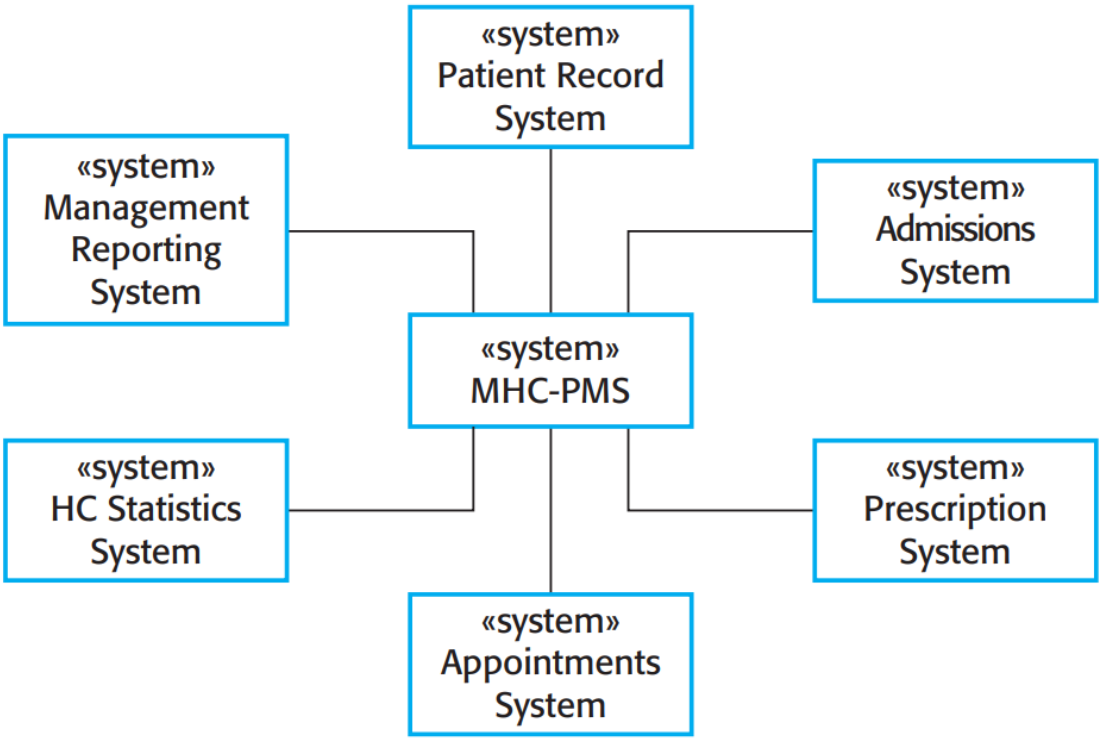
In software engineering, the Class Diagram of UML is a type of static structure diagram that describes the structure of the system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes. An object class can be thought of as a general definition of one kind of system object and an association is a link between classes that indicates that there is a relationship between these classes. A line between classes indicates that there is an association between classes. If each end of the connection is annotated with 1, this means there is a 1:1 relationship between objects of these classes. The classes in a Class Diagram represent both the main elements, interactions in the application and the classes to be programmed and Class Diagram is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code and data modeling.



[Figure 5] The consultation class

In the Class Diagram, classes are represented with boxes that contain three sections: The name of the object class is in the top section. The class attributes are in the middle section. This must include the attribute names and, optionally, their types. The operations (or methods) associated with the object class are in the lower section of the rectangle.

2.2.5. Context Diagram



[Figure 6] The context of the MHC-PMS

Context Diagrams are tools for visualizing context models. Context models show how a system that is being modeled is positioned in an environment with other systems and processes. They help define the boundaries of the system to be developed. Context Diagrams normally show that the environment includes several other systems. However, they do not show the types of relationships between the systems in the environment and the system that is being specified. Context Diagrams represent all external entities that may interact with the system, and it is a high level view of the system. The objective of the Context Diagram is to focus attention on external factors and events that should be considered in developing a complete set of systems requirements and constraints.

2.3. Applied Tools

2.3.1. Microsoft PowerPoint

Microsoft PowerPoint is well known as a tool for creating presentations. It is convenient to draw various types of diagrams by freely placing text and figures. It has the advantages of being easy to access and convenient to use and edit.

2.3.2. GitMind

GitMind is an online mind mapping and brainstorming tool written in JavaScript. GitMind support to create mind maps, flowcharts, decision tree, UML diagrams, Entity-Relationship Model, and so on. Commonly used for knowledge management, meeting note, project management, and other creative tasks.

2.4. Project Scope

Over the past few years, almost all festivals have been postponed or canceled due to COVID-19, but the Space for School Festival project provides a service that allows students to have a similar experience to an offline festival by holding an online school festival and participating in the festival through the VRChat platform. By implementing the main activities of offline school festival inside the VRChat World, through interaction with other users and objects, this service can provide an experience similar to participating in an offline festival to those who participate in the World. This project is a Unity-based project and provides services through the VRChat platform. The project scope is as follows.

The person who opens the world in VRChat becomes the host for the festival, and after that, users can enter the world. Users can perform on stage or sit and watch the performance and participate in lottery events and booth activities such as treasure hunts, quizzes, and simple sports games that are pre-implemented inside the world. In addition, users can check the festival schedule through the schedule table on either side of the central screen. The center screen is connected to the camera facing the stage to mirror the stage, and YouTube video can be played on the screen through URL input. Performers prepare for the performance on the backstage, and when the performance is ready, the host can activate the stage lights and open and close the curtains covering the stage through the event control object.

2.5. References

The user of this SDD may need the following documents for reference:

- Team 1, 2020 Spring. Software Design Document, SKKU.
- Ian Sommerville, *Software Engineering.*
- Wikipedia

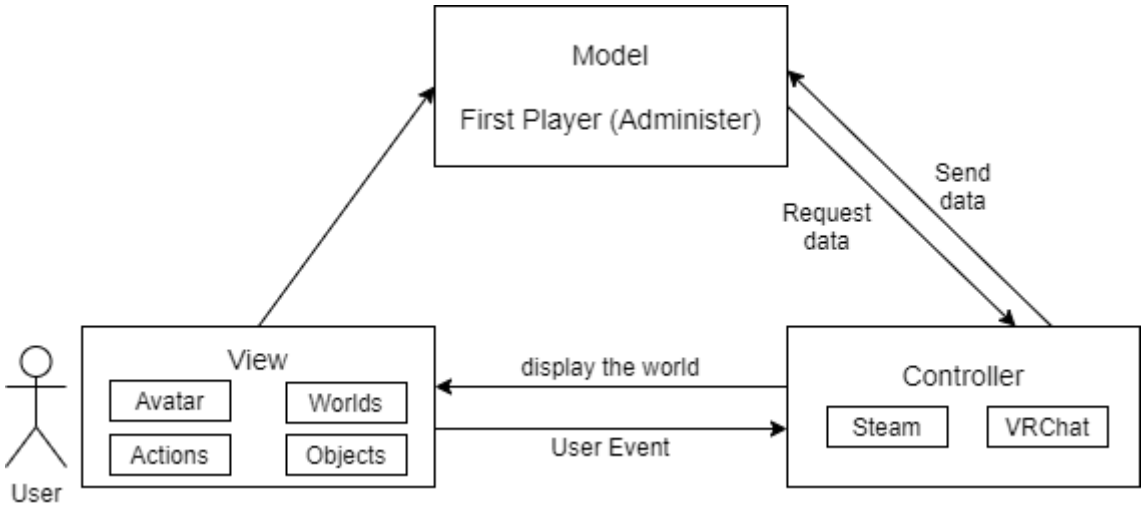
3. System Architecture - Overall

3.1. Objectives

We describe and show the overall system architecture, context diagram, sequence diagram and use case diagram in this chapter.

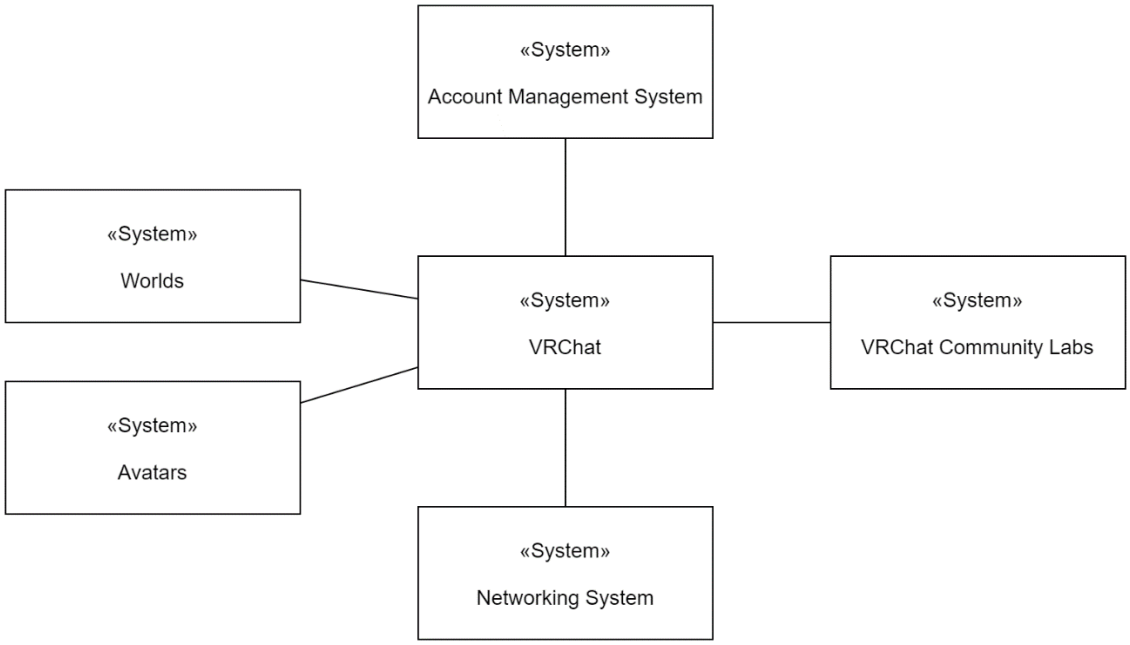
3.2. System Organization

This service uses VRChat's client-server model to make the VRChat service a server and a VRChat user as a client. When one of the clients opens the world, it becomes the administer. The controller recognizes the action and sends that data to the first player (the owner of the network object). The first player changes the object variable and sends data to the controller. Other users access the open world and interact with various objects in the world. The interaction becomes a user event and sends data to the Controller. The controller updates to show the world to all players.



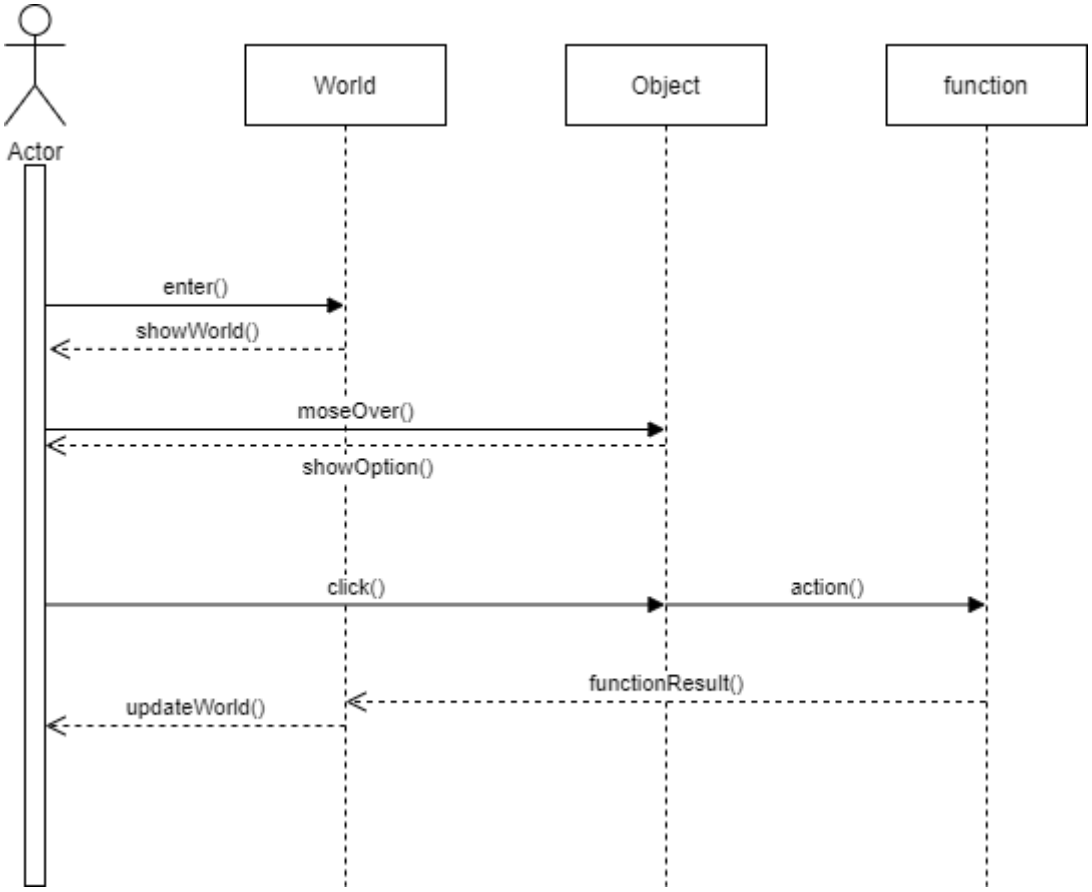
[Figure 7] Overall system architecture

3.2.1. Context Diagram



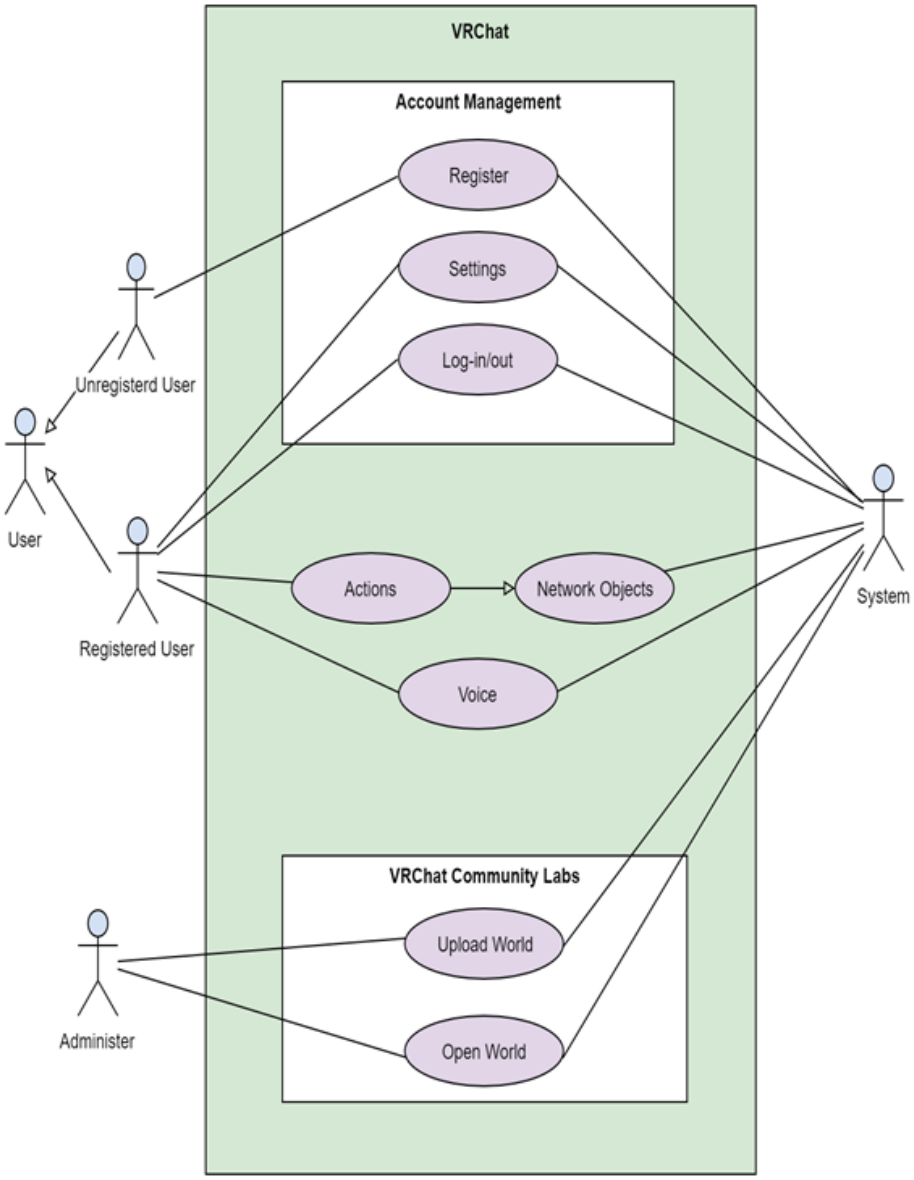
[Figure 8] Overall context diagram

3.2.2. Sequence Diagram



[Figure 9] Overall Sequence Diagram

3.2.3. Use Case Diagram



[Figure 10] Use case diagram

4. System Architecture - Frontend

4.1. Objectives

This chapter describes structure, attributes and function of the frontend system and describe the relation of each component.

4.2. Subcomponents

4.2.1. Stage

The stage class deals with the main performance and event. The showrunner hosts a show and the participants perform. The microphone volume is amplified so that sounds can be heard from far away, the lighting can be adjusted, and a lottery event is held.

4.2.1.1. Attributes

These are the attributes that mic object has.

- **Mic Volume**: the volume of mic

These are the attributes that light object has.

- **On Off**: the on/off status of light
- **Direction**: direction of light. coordinate(x, y, z)

These are the attributes that lottery object has.

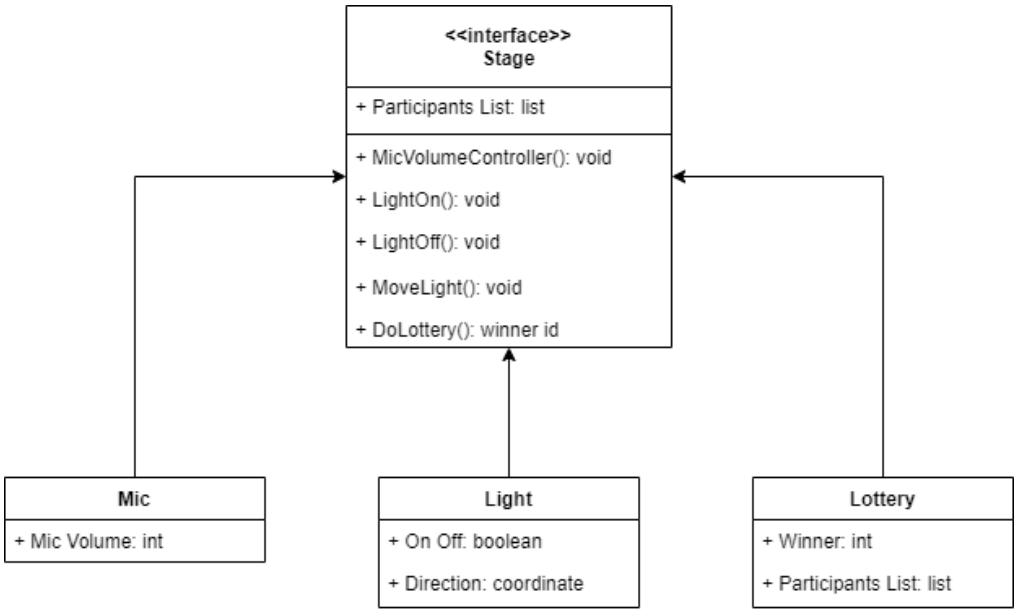
- **Participants List**: participants of lottery
- **Winner**: winner of lottery

4.2.1.2. Methods

These are the methods that stage class has.

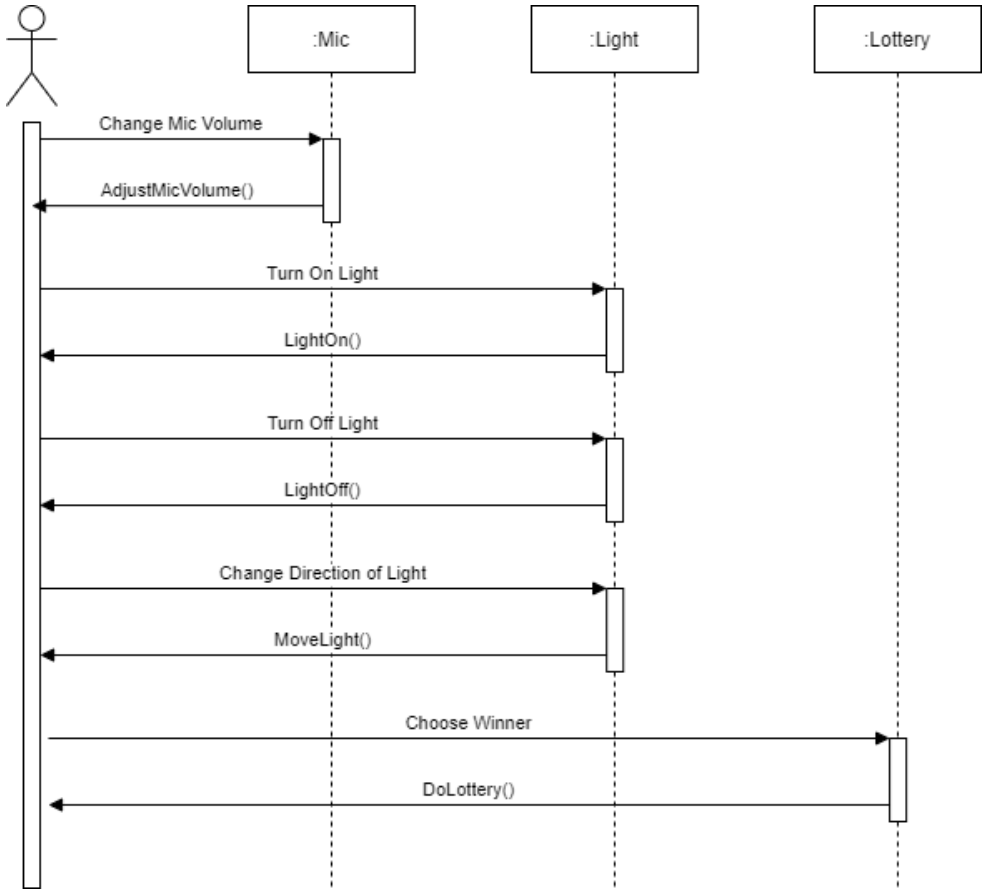
- AdjustMicVolume()
- LightOn()
- LightOff()
- MoveLight()
- DoLottery()

4.2.1.3. Class Diagram



[Figure 11] Class diagram - Stage

4.2.1.4. Sequence Diagram



[Figure 12] Sequence diagram - Stage

4.2.2. Backstage

The show participants wait for stage in backstage. Participants can see their avatars by mirror and can see show stage by a small screen like the main screen on the stage.

4.2.2.1. Attributes

These are the attributes that backstage object has.

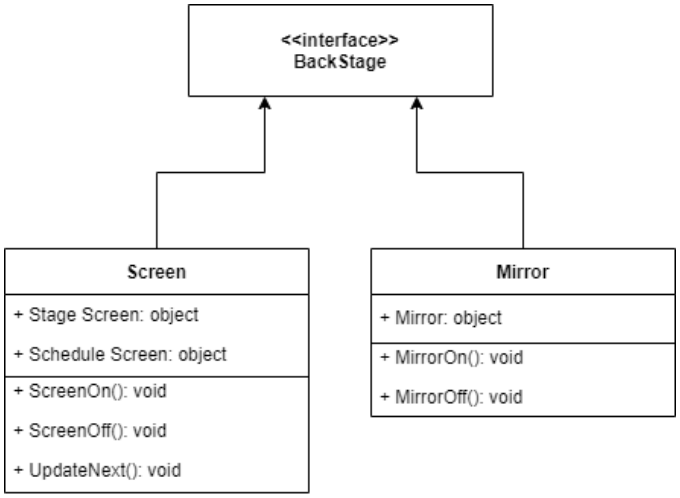
- **Mirror**: mirror that participants can see their avatars
- **Screen**: show stage, schedule and who to wait for next stage

4.2.2.2. Methods

These are the methods that backstage class has

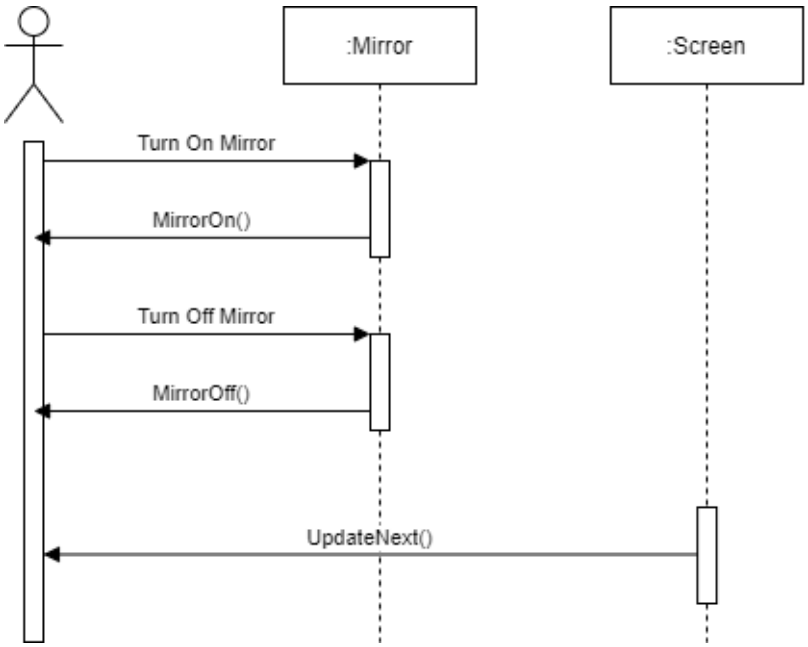
- UpdateNext()
- MirrorOn()
- MirrorOff()

4.2.2.3. Class Diagram



[Figure 13] Class diagram - Backstage

4.2.2.4. Sequence Diagram



[Figure 14] Sequence diagram - Backstage

4.2.3. Screen

Screen shows video from Youtube or shows the stage by camera. Video screen and stage screen is switched as a host pushes the button for main screen.

4.2.3.1. Attributes

These are the attributes that screen object has.

- **Switch Button**: switch a screen for Youtube video and a screen for mirroring stage

These are the attributes that video screen object has.

- **On/Off Status**: turn on screen or turn off screen by switching
- **Video Link**: video link to play on screen

These are the attributes that stage screen object has.

- **On/Off Status**: turn on screen or turn off screen by switching

These are the attributes that sub screen object has.

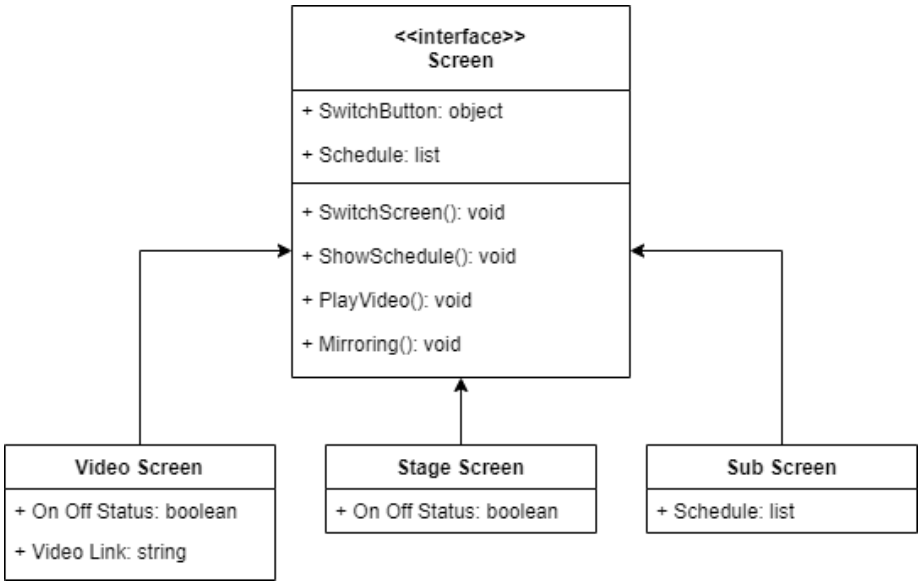
- **Schedule**: schedule of whole show

4.2.3.2. Methods

These are the methods that screen class has.

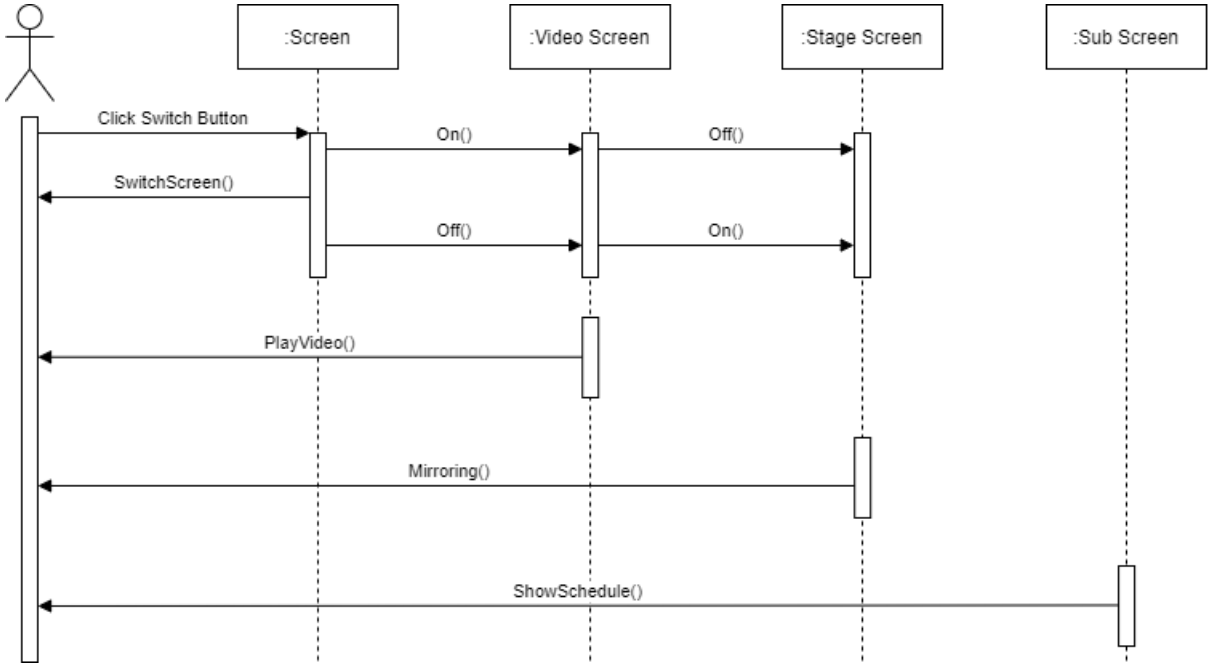
- SwitchScreen()
- ShowSchedule()
- PlayVideo()
- Mirroring()

4.2.3.3. Class Diagram



[Figure 15] Class diagram - Screen

4.2.3.4. Sequence Diagram



[Figure 16] Sequence diagram - Screen

4.2.4. Camera

Camera Class is to capture the stage to show on the main screen of stage and the screen of

backstage. It mirrors stage and send video data to screen. There is a camcorder object that participants can hold and take.

4.2.4.1. Attributes

These are the attributes that camera object has.

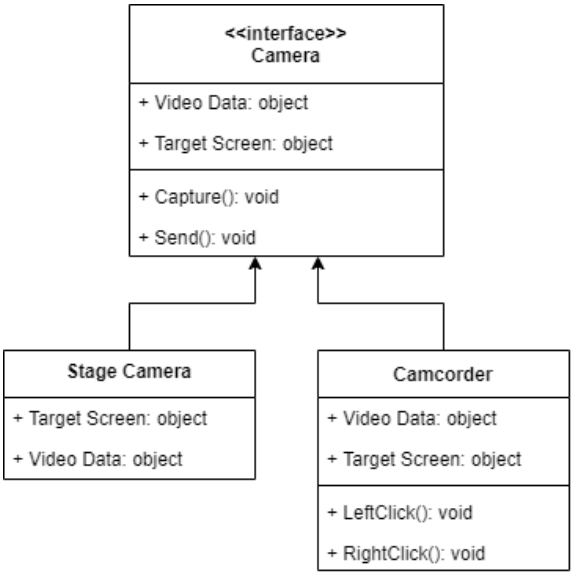
- **Video Data**: video data that camera captures
- **Target Screen**: screen to send video data

4.2.4.2. Methods

These are the attributes that camera class has.

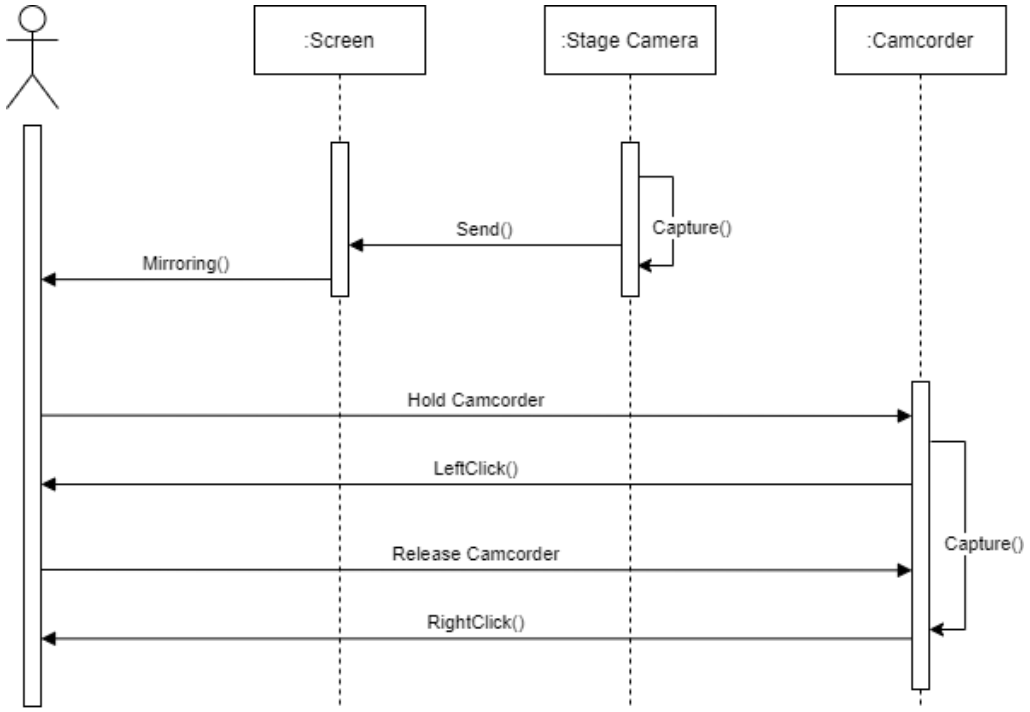
- Capture()
- Send()
- LeftClick()
- RightClick()

4.2.4.3. Class Diagram



[Figure 17] Class diagram - Camera

4.2.4.4. Sequence Diagram



[Figure 18] Sequence diagram - Camera

4.2.5. Quiz Booth

Users can win prizes by challenging the ranking the most in time limit for the quiz presented in this booth.

4.2.5.1. Attributes

These are the attributes that preference object has.

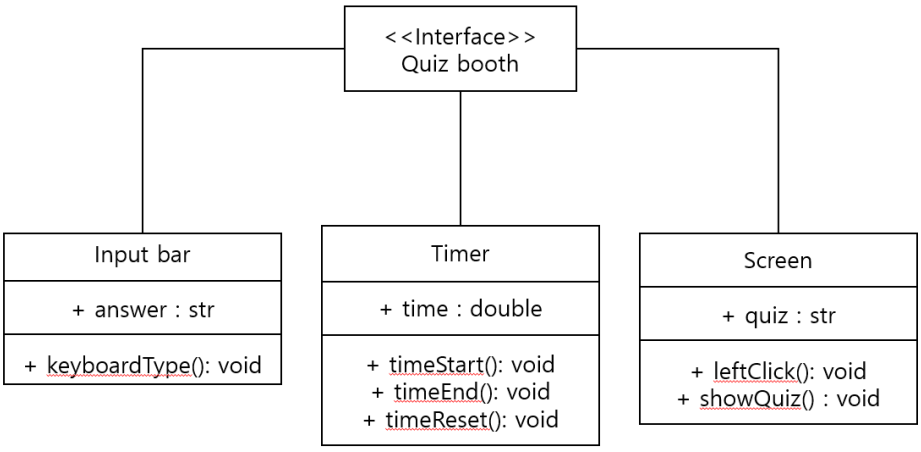
- **Quiz booth**: This booth is to find a treasure in the island for 3 minutes.
- **Timer:** The timer starts when the user clicks the portal object.
- **Screen:** The screen shows the quiz user needs to guess
- **Input bar:** The input bar shows the answer user inputs through keyboard.

4.2.5.2. Methods

There are six methods in the stamp booth.

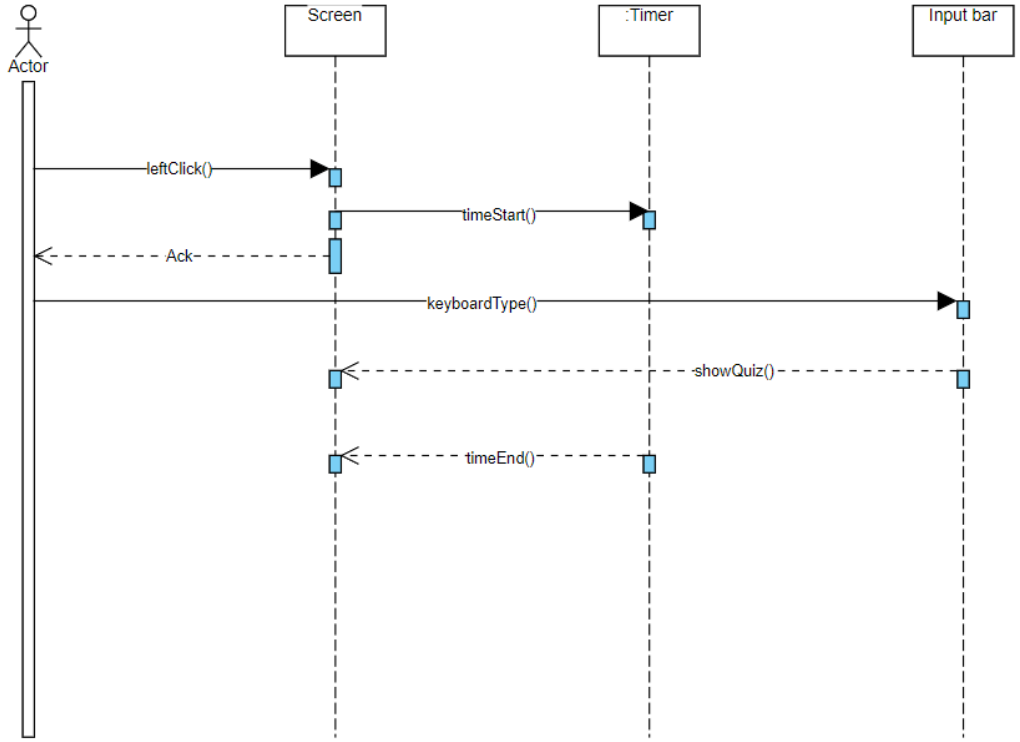
- keyboardType()
- leftClick()
- timeStart()
- timeEnd()
- timeReset()
- showQuiz()

4.2.5.3. Class Diagram



[Figure 19] Class diagram – Quiz booth

4.2.5.4. Sequence Diagram



[Figure 20] Sequence Diagram – Quiz booth

4.2.6. Information Booth

Users can get a variety of festival-related information through this booth. Users can check information such as the performance schedule of the entire festival, the schedule of the prize event, the type of booth, and compensation.

4.2.6.1. Attributes

These are the attributes that preference object has.

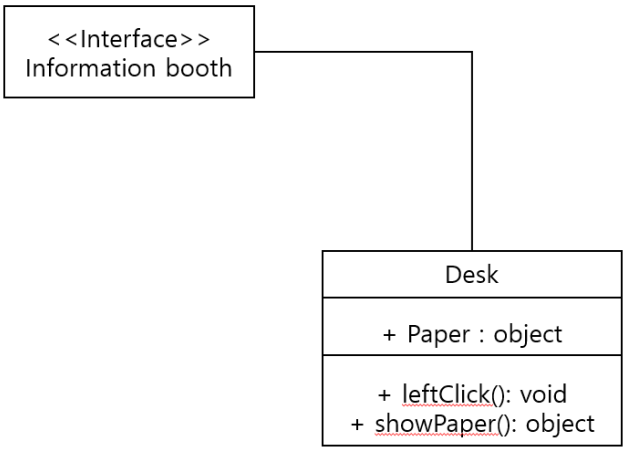
- **Information booth**: This booth is to find a treasure in the island for 3 minutes.
- **Desk**: Users can check out various festival-related information documents on their desks.

4.2.6.2. Methods

There are two methods in the stamp booth.

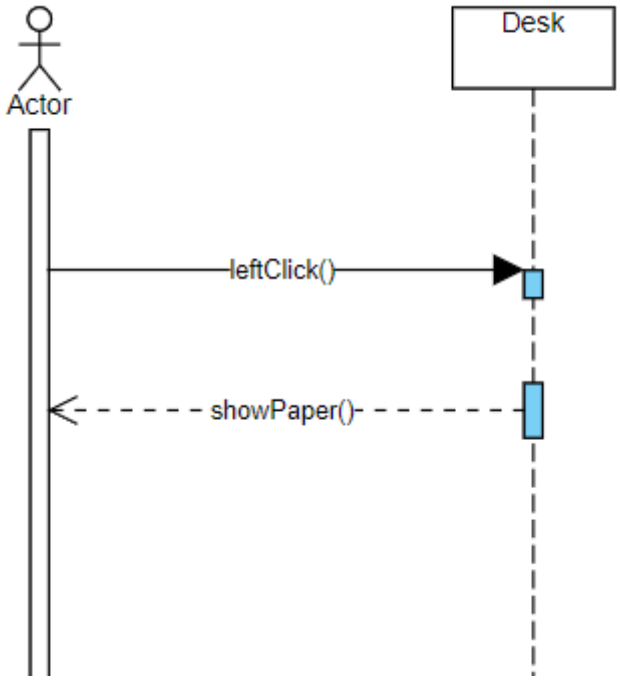
- leftClick()
- showPaper()

4.2.6.3. Class Diagram



[Figure 21] Class Diagram – Information booth

4.2.6.4. Sequence Diagram



[Figure 22] Sequence Diagram - Information booth

4.2.7. Stamp Booth

This booth is a place where you can get a prize after experiencing all the booths.

4.2.7.1. Attributes

These are the attributes that preference object has.

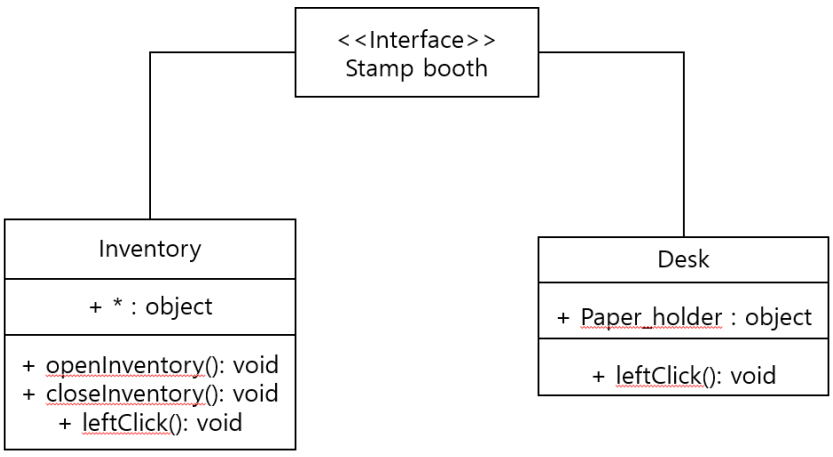
- **Stamp booth**: This booth is to get a prize after experiencing all the booths.
- **Inventory:** Users carry paper that can be stamped by booth through inventory.
- **Desk:** The user can place stamped paper on the desk to check if all stamps have been received.

4.2.7.2. Methods

There are two methods in the stamp booth.

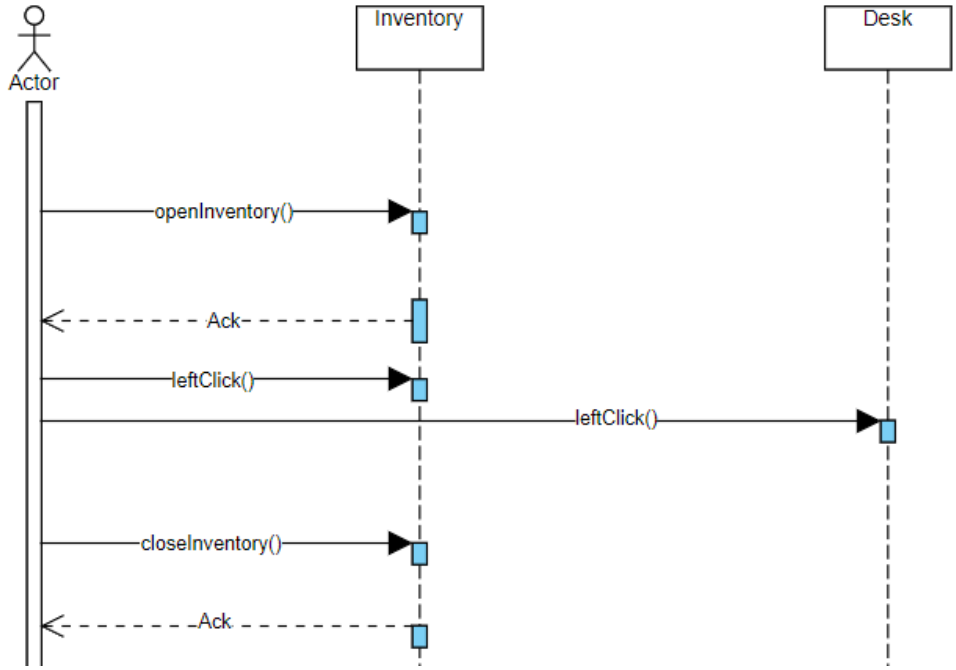
- openInventory()
- closeInventory()
- leftClick()

4.2.7.3. Class Diagram



[Figure 23] Class Diagram – Stamp booth

4.2.7.4. Sequence Diagram



[Figure 24] Sequence Diagram – Stamp booth

4.2.8. Treasure Hunt Booth

This booth is to find a treasure on the new location, the island. The users can teleport in the specific point. The users try to find a treasure in the island. If they find a treasure, then return the booth and get the prize.

4.2.8.1. Attributes

These are the attributes that preference object has.

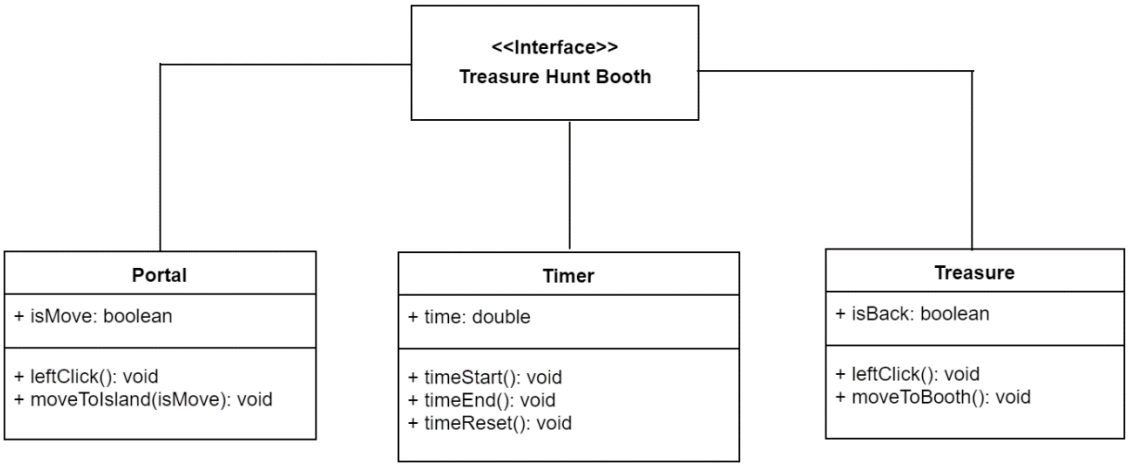
- **Treasure hunt booth**: This booth is to find a treasure in the island for 3 minutes.
- **Portal**: When user clicks this object, the user teleports to the island.
- **Island**: This island is big and the treasure is located in an island.
- **Treasure**: It looks like a treasure chest. If user finds and clicks it, the user can move back to the treasure hunt booth.
- **Timer:** The timer starts when the user clicks the portal object.

4.2.8.2. Methods

There are six methods in the baseball booth.

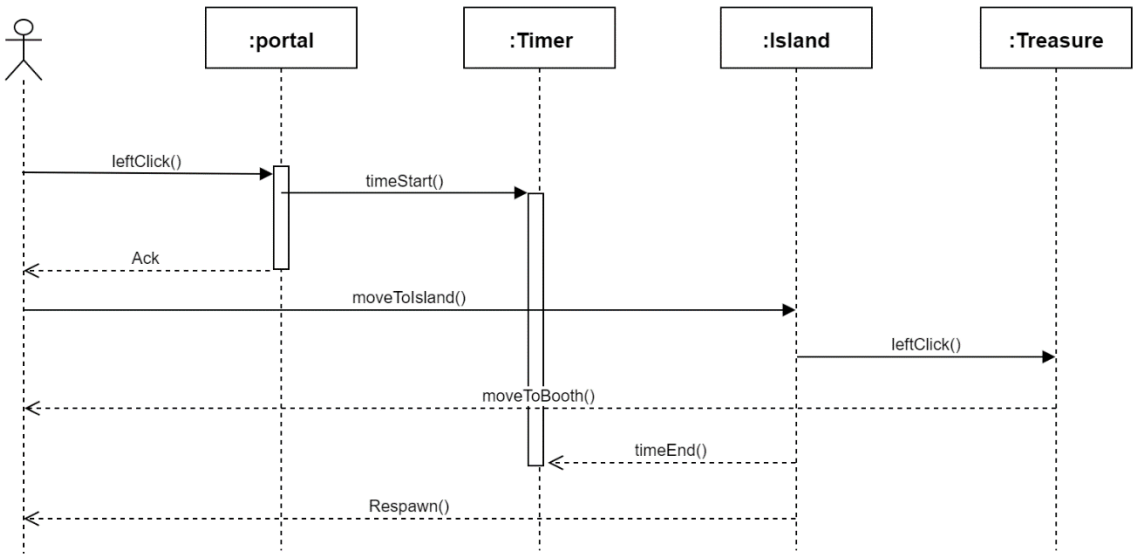
- leftClick()
- moveToIsland()
- timeStart()
- timeEnd()
- timeReset()
- moveToBooth()

4.2.8.3. Class Diagram



[Figure 25] Class diagram – Treasure Hunt booth

4.2.8.4. Sequence Diagram



[Figure 26] Sequence diagram – Treasure Hunt

4.2.9. Baseball booth

This booth is to enjoy pitching a baseball. The users pick up the baseball and throw to the target. The target is a hole. If the baseball goes into the hole, the count increases. After throwing all baseballs the scoreboard will be updated.

4.2.9.1. Attributes

These are the attributes that preference object has.

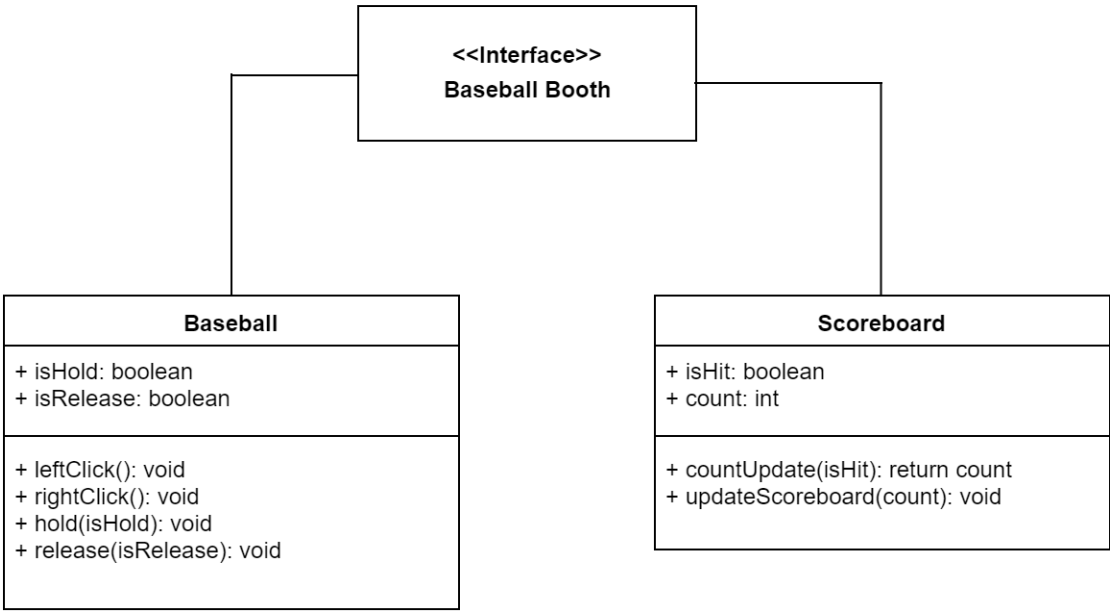
- **Baseball booth**: This booth is to enjoy pitching baseball. The user has to throw a baseball into the target point, the hole.
- **Baseballs**: There are many baseballs. The size of the baseball is the same.
- **Target**: The target is the hole. If the baseball hit the target, the count increases.
- **Score board**: The score is counted how many baseballs are in the hole.

4.2.9.2. Methods

There are six methods in the baseball booth.

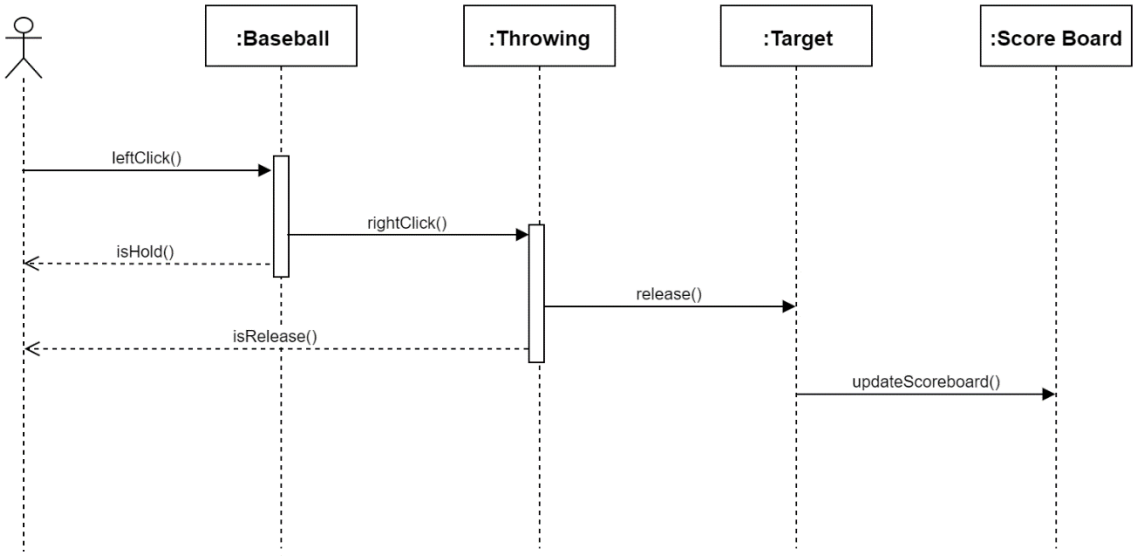
- leftClick()
- rightClick()
- hold()
- release()
- countUpdate()
- updateScoreboard()

4.2.9.3. Class Diagram



[Figure 27] Class diagram – Baseball booth

4.2.9.4. Sequence Diagram



[Figure 28] Sequence diagram – Baseball booth

5. Testing Plan

5.1. Objectives

This chapter describes plans for tests with three main subgroups: development testing, release testing, and user testing. These tests help to detect potential errors and defects in the application and provide stable functionality to customers.

5.2. Testing Policy

5.2.1. Development Testing

Development testing is an approach in software development that aims to bring the development and testing phases closer together. By performing development testing, we can get higher code quality at any given time because new code is continually being tested and we can shortened time to market for new features. This stage is important because it enables higher levels of efficiency in the software development life cycle, it helps to reduce the effect of software errors, and it speed the delivery of new features and bug fixes to customers.

Specific tests include static code analysis, data flow analysis, peer code review, and unit tests. In this stage, software may be unstable because the software is in the process of being tested.

Through this process, we can check whether we meet the following three non-functional requirements: Performance, reliability, security.

5.2.1.1. Performance

In software quality assurance, performance testing is in general a testing practice performed to determine how a system performs in terms of responsiveness and stability under a particular workload.

In our system, when more than a certain number of people gather in the world, problems such as lagging and deteriorating performance are the most important problems. Therefore, we will test the performance degradation according to the number of visitors in various ways and find an appropriate trade-off. Through the test, the number of people entering will be strictly limited according to the set standards, and through this, we will improve it so that it can be used without any problems with the server.

5.2.1.2. Reliability

Reliability tests, which are often carried out at the subsystem level, include tests to estimate failure frequencies for one-shot devices, mean time to failure for continuously operating, non-repairable devices, mean time between failures for repairable systems, and probabilities of mission success as a function of reliability performance for all systems. Therefore, we will continue to conduct development tests from the unit development stage and repeatedly check for failures while each unit is integrated into the system.

5.2.1.3. Security

In these days, as almost every company is digitally transformed into a technology company, our cumulative exposure to risk has grown exponentially. So we have to build it on a foundation of robust security.

Security testing can be described as a type of software testing that’s deployed to identify vulnerabilities that could potentially allow a malicious attack. By engaging in this activity, we can uncover all loopholes in the system to prevent the loss of information, revenue, and a negative impact on brand value. The primary objective here is to detect all possible risks before

the software is integrated into enterprise infrastructure. This approach also provides developers with ample time to fix these problems before it becomes a significant security incident.

5.2.2. Release Testing

Release testing refers to coding practices and test strategies that give teams confidence that a software release candidate is ready for users. Release testing aims to find and eliminate errors and bugs from a software release so that it can be released to users. Let's dive in and explore several methods used to perform release testing.

Release testing starts with having a software release candidate. This represents a snapshot of the code baseline containing all new software features and bug fixes selected for this release. This software candidate has been packaged and labeled as "production-ready." It's this software release candidate that we want to test to ensure that it's ready for our users.

In other words, the objective of release testing is to build confidence into a release candidate. Release testing is a testing approach or strategy rather than one single grand testing method.

Like any other testing approach, release testing aims at breaking the system running the software release candidate in a controlled environment. Development teams focus their quality assurance efforts on this one release candidate, trying to break it so it can be fixed before it gets to actual users.

5.2.3. User Testing

User testing is an irreplaceable usability practice that provides direct input to how real users use the system. These tests are of paramount importance to the success of the final product because applications of functions that cause confusion among users do not last long.

We will pay more attention to the intuition of the product and test it on users who have not been exposed in advance.

5.2.4. Testing Case

The test case will include boot participation and the availability of stage and backstage, which are the most basic functions, and will also include additional performance and security related to the number of people. We will conduct more than three tests for each aspect and receive evaluation from users who participated in the test.

6. Development Plan

6.1. Objectives

This chapter explains the technology and environment for application development.

6.2. Frontend Environment

6.2.1. Unix Hub

Unity is the world’ s most popular development platform for creating 2D and 3D multiplatform games and interactive experiences. Unity is a game engine developed by Unity Technologies in August 2004, mainly suitable for the development of low-end/small-scale games. The initial Unity engine was one of the multi-platforms, and the browser was also mainly supported, but it turned to a game engine as developers began to use it to create games based on the fact that it was relatively simple and easy to use. The Unity engine has the advantage that the tool's GUI is very intuitive and easy to build. The WYSIWYG type tool allows users to intuitively change the positions of internal assets or import the assets very easily. These were the biggest secrets to Unity's explosive popularity in small/indie games. It also has the advantage of being able to develop simple games with low-end targets even on low-end PCs due to its low-demand specifications.

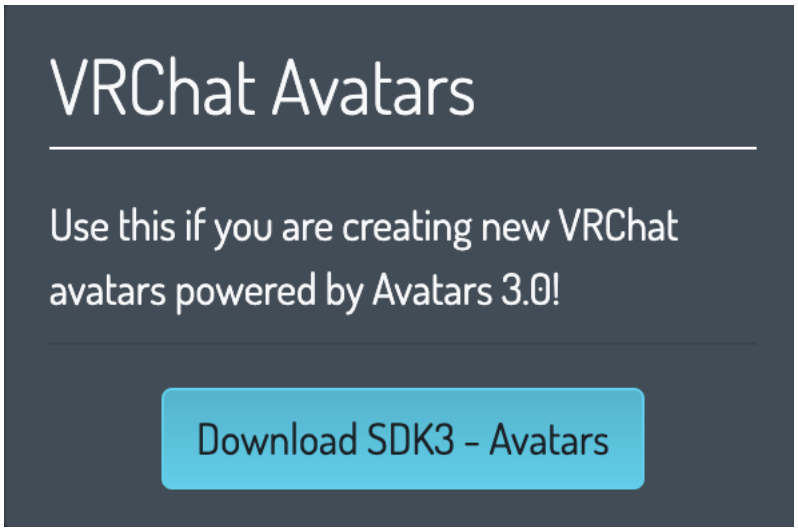
[Figure 29] Unity

6.2.2. VRChat SDK3

VRChat SDK3 is the latest version of VRChat SDK. It is divided into two versions, one for avatars, and the other for worlds production. VRChat SDK3-Avatars, Worlds can be used simultaneously in one project.

6.2.2.1. VRChat SDK3 - Avatars

The VRChat SDK3-Avatars package comes with Avatars 3.0, the latest version of the framework for avatar production. Avatars 3.0 is a huge collection of features for avatars in VRChat. It is a new version of VRChat avatar feature framework. AV3's features are focused on improving expression, performance, and the abilities of avatars in VRChat. Moreover, we've learned a lot from years of watching users find ways to do cool stuff with Avatars 2.0. A lot of those methods are considered "hacky", and it is hard for us to support those. We want to formalize the process so you can do the things you want, access them more easily, and use them in a system that is officially supported.



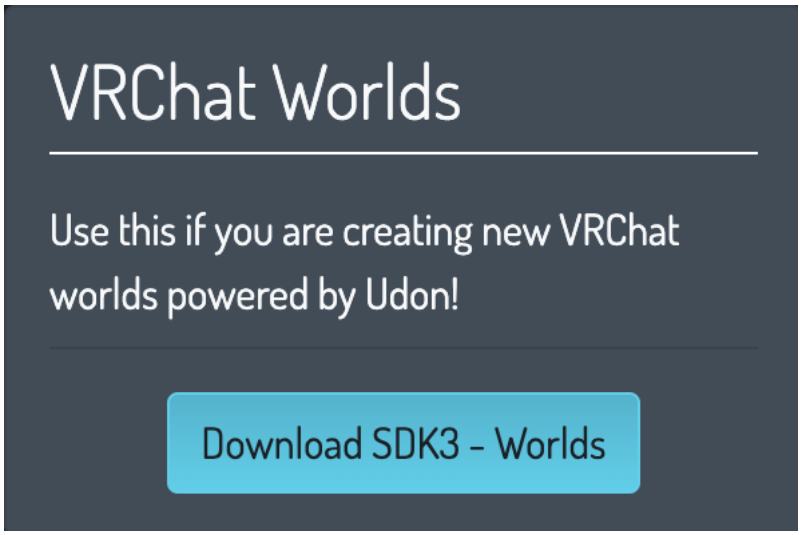
[Figure 30] VRChat SDK3 - Avatars

6.2.2.2. VRChat SDK3 - Worlds

The VRChat SDK3-Worlds package comes with a VRChat Udon that allows you to program actions. VRChat Udon is a programming language built completely in-house by the VRChat Development Team. It is designed to be secure, performant, and easy to use via the VRChat Udon Node Graph, a built-in visual programming interface that uses nodes and wires (we call

them “noodles”) to connect flow, inputs, and outputs. You can build complex behaviors with Udon-- far more complex and easier to understand than unwieldy chains of Triggers and Actions. Not only can you replicate the full behavior of Triggers and Actions with VRChat Udon, but you can create your own behaviors, sync variables with others, interact with scenes, interact with players, and more.

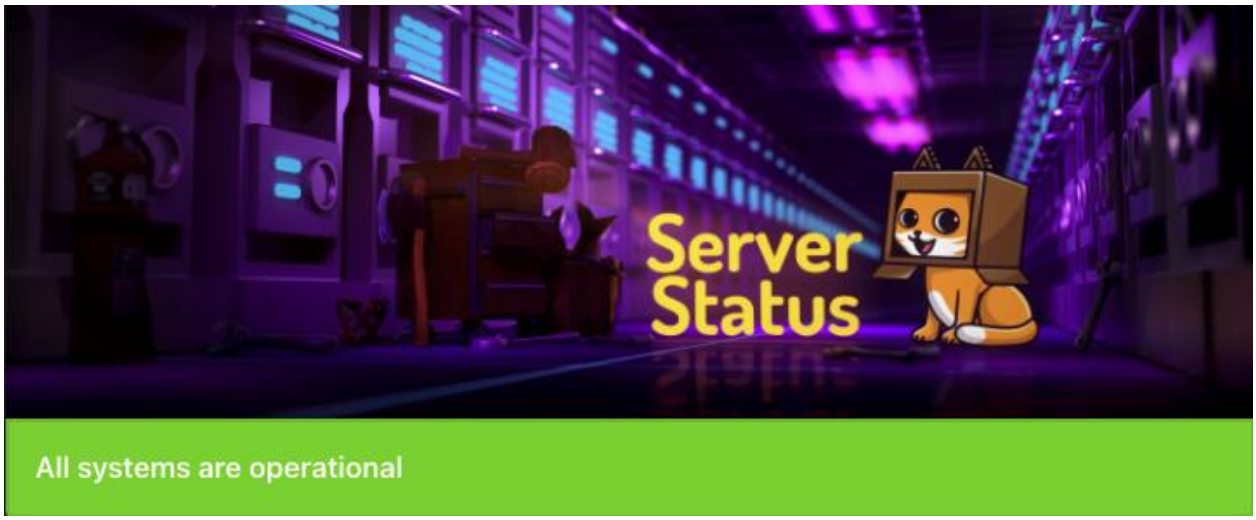
In addition, Udon runs in both the VRChat client and the Unity Editor, allowing you to test and debug your creations with ease. For the more technically inclined: VRChat Udon is a VM running bytecode compiled from Udon Assembly. You can generate Udon Assembly using the built-in VRChat Udon Node Graph UI, writing your own Udon Assembly, or even by writing your own compiler to generate Udon Assembly or bytecode programs directly.



[Figure 31] VRChat SDK3 - Worlds

6.3. Backend Environment

6.3.1. Steam Server



[Figure 32] VRChat Server Status

We will use the server provided by VRChat, and you can check the current status of the servers in VRChat Status. Server statuses are live and confirmed each time this page is loaded, we do not cache data to avoid out of date information.

6.3.2. SteamDB

[Figure 33] SteamDB

VRChat uses SteamDB for database. SteamDB was created to give more insight into the Steam database. SteamDB track updates for both applications and packages, and keep a history of all changes made to both applications and packages. They also have a range of other tools such as the Calculator to give people insight into their Steam accounts that would otherwise be impossible.

SteamDB is a purely informational website, it does not provide any downloads, and does not solicit piracy.

6.4. Constraints

The software will be designed and implemented based on the contents mentioned in this document. Other details will be carried out in the direction preferred by the developer while developing, and they must comply with the following.

- The technology provided by VRChat is prioritized.
- Implement functions without overloading the server.
- Do not use technologies or software that require separate licenses or pay royalties.
- It is decided in the direction of improving the overall system performance.
- Convenience is prioritized in development.
  - ✓ Reuse the software already in use as much as possible.
  - ✓ Consider system development costs and maintenance costs.
- Optimize the source code so that system resources are not wasted.
- Follow the system requirements provided by VRChat.

[Table 1] VRChat requirement

System Requirement	
Section	**Minimum Specification**
OS	Windows 7 Windows 8.1 Windows 10
Processor	Intel® i5-4590 ~ AMD FX 8350 ~
RAM	4GB
Graphic card	NVIDIA GeForce® GTX 970 ~ AMD Radeon™ R9 290 ~ Intel UHD Graphics 610 ~
DirectX	Version 11
Storage	1GB

6.5. Assumptions and Dependencies

All systems in this document are designed and implemented based on Unix devices and VRChat SDK3 and are created under the assumption that they run on Windows operating systems. Therefore, all content is based on a minimum Windows 7 operating system and may not apply to other operating systems or versions.

7. Supporting Information

7.1. Software Design Specification

This software design specification was written in accordance with the IEEE Recommendation (IEEE Recommended Practice for Software Design Description, IEEE-Std-1016).

7.2. Document History

[Table 2] Document History

Date	Version	Description	Writer
2021/11/19	1.0	Style and Overview	Hyejin Yoon
2021/11/19	1.1	Addition of 4.2.7, 4.2.8	Gyeongun Kang
2021/11/19	1.2	Addition of 5, 6	Soo Namgoong
2021/11/20	1.3	Addition of 1, 2	Jeongbok An
2021/11/20	1.4	Addition of 4.2.1~4.2.4	Yuyeong Seo
2021/11/20	1.5	Addition of 4.2.5 ~ 4.2.7	Seunggu Kang
2021/11/20	1.6	Addition of 3	Hyejin Yoon
2021/11/21	1.6.1	Revision of Style	Hyejin Yoon
2021/11/21	1.6.2	Revision of 4.2.7, 4.2.8	Gyeongun Kang
2021/11/21	1.6.3	Revision of 4.2.1~4.2.4	Yuyeong Seo
2021/11/21	1.7	Addition of 7.1 and Revision of style	Jeongbok An
2021/11/21	1.7.1	Addition of contents	Hyejin Yoon
2021/11/21	vf review	Review	All