School Council Election & Voting using Metaverse

Software Design Specification

2021.11.21.

Introduction to Software Engineering 41

TEAM 4

Team Leader	Gwanjong Park
Team Member	Seungji Lee
Team Member	Daeun Lim
Team Member	Soyoung Park
Team Member	Hojin Jeon
Team Member	Jiho Jang
Team Member	Jungin Lee

CONTENTS

1. Preface **11**

1.1. Readership 11
1.2. Scope 11
1.3. Objective 11
1.4. Document Structure 11

2. Introduction **12**

2.1. Objectives 12
2.2. Applied Diagrams 12
2.2.1. UML 13
2.2.2. Use Case Diagram 13
2.2.3. Class Diagram 13
2.2.4. Sequence Diagram 13
2.2.5. Context Diagram 13
2.2.6. Entity Relationship Diagram 14
2.3. Applied tools 14
2.3.1. StarUML 14
2.3.2. diagrams.net 14
2.4. Project Scope 14

3. System Architecture - Overall **14**

3.1. Objectives 14
3.2 System Organization 15
3.2.1 Context Diagram 15
3.2.2 Sequence Diagram 16
3.2.3 Use Case Diagram 17

4. System Architecture - Frontend **18**

4.1. Objectives 18
4.2. Subcomponents 18
4.2.1 White Board 18
4.2.1.1. Attribute 18
4.2.1.2. Methods 18
4.2.1.3. Class Diagram 19
4.2.1.4 Sequence Diagram 19

4.2.2. Timer 20
  4.2.2.1. Attribute 20
  4.2.2.2. Methods 20
  4.2.2.3. Class Diagram 21
  4.2.2.4 Sequence Diagram 22
4.2.3. Request to speak 22
  4.2.3.1. Attribute 22
  4.2.3.2. Methods 23
  4.2.3.3. Class Diagram 23
  4.2.3.4 Sequence Diagram 24
4.2.4 Visual content 24
  4.2.4.1 Attributes 24
  4.2.4.2 Methods 24
  4.2.4.3 Class Diagram 25
  4.2.4.4 Sequence Diagram 26
4.2.4 Notice Board 26
  4.2.4.1 Attributes 26
  4.2.4.2. Methods 26
  4.2.4.3. Class Diagram 27
  4.2.4.4. Sequence Diagram 28
4.2.5 Q&A Board 29
  4.2.5.1 Attributes 29
  4.2.5.2. Methods 29
  4.2.5.3. Class Diagram 29
  4.2.5.4. Sequence Diagram 31
4.2.6 Free Board 32
  4.2.6.1 Attributes 32
  4.2.6.2. Methods 32
  4.2.6.3. Class Diagram 32
  4.2.6.4. Sequence Diagram 34
4.2.7. Voting process management 35
  4.2.7.1. Attributes 35
  4.2.7.2. Methods 35
  4.2.7.3. Class Diagram 36
  4.2.7.4. Sequence Diagram 37
4.2.8. Vote 37
  4.2.8.1. Attributes 37
  4.2.8.2. Methods 38

4.2.8.3. Class Diagram ... 38
4.2.8.4. Sequence Diagram ... 39

5. System Architecture - Backend ... 39

5.1. Objectives ... 39
5.2. Overall Architecture ... 40
5.3. Subcomponents ... 41
5.3.1. Debating Function ... 41
5.3.1.1. Class diagram ... 41
5.3.1.2. Sequence Diagram ... 41
5.3.2. Visual Content Function ... 42
5.3.2.1. Class diagram ... 42
5.3.2.2. Sequence diagram ... 42
5.3.3. 2D Content Function ... 43
5.3.3.1. Class diagram ... 43
5.3.3.2. Sequence diagram ... 43
5.3.4. 3D Content Function ... 44
5.3.4.1. Class diagram ... 44
5.3.4.2. Sequence diagram ... 44
5.3.5. Board ... 45
5.3.5.1. Class Diagram ... 45
5.3.5.2. Sequence Diagram ... 45
5.3.6. Voting ... 46
5.3.6.1. Class diagram ... 46
5.3.6.2. Sequence diagram ... 46

6. Protocol Design ... 47

6.1. Objectives ... 47
6.2. 2D Object Files ... 47
6.3. 3D Object Files ... 47
6.4. Video Files ... 47
6.5. White Board ... 48
6.5.1. Input agenda ... 48
6.5.2. Input candidate ... 48
6.5.3. Get debate info ... 49
6.5.4. Next stage ... 50
6.5.5. Show info ... 51
6.6. Timer ... 52

6.6.1. Set time 52
6.6.2. Reset 53
6.6.3. Stop 54
6.6.4. Start 55
6.6.5. Show time 55
6.7. Request to speak 56
6.7.1. Insert request 56
6.7.2. Delete request 57
6.7.3. Show request 58
6.8. Video object 59
6.8.1. Video object request 59
6.8.2. 2D object request 60
6.8.3. 3D object request 61
6.9. Board 61
6.9.1. Read post request 61
6.9.2. Verification before write request 62
6.9.3. Write post request 63
6.9.4. Update Post List 64
6.10. Voting 65
6.10.1. Create voting process request 65
6.10.2. Read voting process request 66
6.10.3. Submit the response request 67

7. Database Design 68

7.1. Objectives 68
7.2. ER Diagram 68
7.2.1 Entities 69
7.2.1.1. World Instance 69
7.2.1.2. Bulletin Board 69
7.2.1.3. Message 70
7.2.1.4. Debate 70
7.2.1.5. Survey 71
7.2.1.6. Vote 71
7.2.1.7. Candidate 72
7.2.1.8. Video Content 72

8. Testing Plan 73

8.1. objectives 73

8.2. Testing policy 73
8.2.1. development testing 73
8.2.1.1. Performance 73
8.2.1.2. Reliability 74
8.2.1.3. Security 74
8.2.2. Release Testing 74
8.2.3. User testing 75
8.2.4. Testing Case 75

9. Development Plan 75

9.1. Objectives 75
9.2. Frontend Environment 76
9.2.1. Unity 76
9.2.2. Adobe photoshop 76
9.3. Backend Environment 76
9.3.1. Udon 76
9.4. Constraints 76
9.5. Assumptions and Dependencies 77

10. Supporting Information 79

10.1. Software Requirement Specification 79
10.2. Document History 79

LIST OF FIGURES

[Figure 1] Overall Context Diagram........15
[Figure 2] Overall Sequence Diagram........16
[Figure 3] Use Case Diagram........17
[Figure 4] Class diagram – White Board........19
[Figure 5] Sequence Diagram – White Board........20
[Figure 6] Class diagram – White Board........21
[Figure 7] Sequence Diagram – White Board........22
[Figure 8] Class diagram – White Board........23
[Figure 9] Sequence Diagram – White Board........24
[Figure 10] Class Diagram – Visual content........25
[Figure 11] Sequence Diagram – Visual content........26
[Figure 12] Class Diagram – Notice Board........27
[Figure 13] Sequence Diagram – Notice Board........28
[Figure 14] Class Diagram – Q&A Board........30
[Figure 15] Sequence Diagram – Q&A Board........31
[Figure 16] Class Diagram – Free Board........33
[Figure 17] Sequence Diagram – Free Board........34
[Figure 18] Class Diagram – Voting Process Management........36
[Figure 19] Sequence Diagram – Voting Process Management........37
[Figure 20] Class Diagram – Vote........38
[Figure 21] Sequence Diagram – Vote........39
[Figure 22] Overall Architecture........40
[Figure 23] Class Diagram – Debating Function........41
[Figure 24] Sequence Diagram – Debating Function........41
[Figure 25] Class Diagram – Video content........42
[Figure 26] Sequence Diagram – Video content........42
[Figure 27] Class Diagram – 2D content........43
[Figure 28] Sequence Diagram – 2D content........43
[Figure 29] Class Diagram – 3D content........44
[Figure 30] Sequence Diagram – 3D content........44
[Figure 31] Class Diagram – Board Function........45
[Figure 32] Sequence Diagram – Board Function........45
[Figure 33] Class Diagram – Voting........46
[Figure 34] Sequence Diagram – Voting........46

[Figure 35] ER-Diagram 68
[Figure 36] ER diagram, Entity, World Instance 69
[Figure 37] ER diagram, Entity, Bulletin Board 69
[Figure 38] ER diagram, Entity, Message 70
[Figure 39] ER diagram, Entity, Debate 70
[Figure 40] ER diagram, Entity, Survey 71
[Figure 41] ER diagram, Entity, Vote 71
[Figure 42] ER diagram, Entity, Candidate 72
[Figure 43] ER diagram, Entity, Video Content 72
[Figure 44] Unity logo 76

LIST OF TABLES

[Table 1] Table of input agenda request of white board 48
[Table 2] Table of input agenda response of white board 48
[Table 3] Table of input candidate request of white board 49
[Table 4] Table of input candidate response of white board 49
[Table 5] Table of get debate info request of white board 49
[Table 6] Table of get debate info response of white board 50
[Table 7] Table of next stage request of white board 50
[Table 8] Table of next stage response of white board 51
[Table 9] Table of show info request of white board 51
[Table 10] Table of show info response of white board 52
[Table 11] Table of set time request of timer 52
[Table 12] Table of set time response of timer 53
[Table 13] Table of reset request of timer 53
[Table 14] Table of reset response of timer 53
[Table 15] Table of stop request of timer 54
[Table 16] Table of stop response of timer 54
[Table 17] Table of start request of timer 55
[Table 18] Table of start response of timer 55
[Table 19] Table of show time request of timer 56
[Table 20] Table of show time response of timer 56
[Table 21] Table of insert request of request to speak 56
[Table 22] Table of insert response of request to speak 57
[Table 23] Table of delete request of request to speak 57
[Table 24] Table of delete response of request to speak 58
[Table 25] Table of show request of request to speak 58
[Table 26] Table of show response of request to speak 59
[Table 27] Table of video object request 59
[Table 28] Table of video object response 59
[Table 29] Table of 2D object request 60
[Table 30] Table of 2D object response 60
[Table 31] Table of 3D object request 61
[Table 32] Table of 3D object response 61
[Table 33] Table of read post request 62
[Table 34] Table of read post response 62

[Table 35] Table of verification before write request 62
[Table 36] Table of verification before write response 63
[Table 37] Table of write post request 63
[Table 38] Table of write post response 64
[Table 39] Table of post update request 64
[Table 40] Table of post update response 64
[Table 41] Table of create voting process request 65
[Table 42] Table of create voting process response 65
[Table 43] Table of read voting process request 66
[Table 44] Table of read voting process response 66
[Table 45] Table of read voting process request 67
[Table 46] Table of read voting process response 67
[Table 47] Document History 79

1. Preface

This chapter contains information about the readership, scope, objective, and structure for the Software Design Document that would be used for School Council Election & Voting using Metaverse project.

1.1. Readership

This document composed 10 sections, each with its subsections. The structure for this document is found in Section 1.4. SE-2021-2-Team4 is the main reader for this document, but students, professors, TAs of the Introduction to Software Engineering course, and any other people who are interested in metaverse can also be one.

1.2. Scope

This document is used to provide descriptions of the designs that would be used to implement the School Council Election & Voting using Metaverse via VRChat. There are four major submodules to design, Visual Content Exhibition Module, Vote & Survey Module, Debating Module, Board Module.

1.3. Objective

The main objective of this Software Design Specification document is to describe the design aspects for the School Council Election & Voting using Metaverse. This document describes the software architecture and design decisions. Also, it depicts the structure and design of functions also referred to in the Software Requirement Specification document with some use cases and diagrams.

1.4. Document Structure

1. Preface: The chapter being read right now; It provides information about the readership, scope, objective, and structure for this document.

2. Introduction: It provides definitions and descriptions of diagrams and tools used for this project; It also provides the scope and references that may be necessary to understand this document

3. System Architecture - Overall: It provides information on how the system is organized

4. System Architecture - Frontend: It provides the conceptual model that defines the structure, behavior, and views of the frontend of a system

5. System Architecture - Backend: It provides the conceptual model that defines the structure, behavior, and views of the backend of a system

6. Protocol Design: It provides descriptions of the structure for the protocol used for the interaction of subsystems and the protocol for defining interfaces.

7. Database Design: It provides descriptions of the system data structures and how said data structures are to be represented in a database.

8. Testing Plan: It provides plans for development testing, release testing, and user testing.

9. Development Plan: It provides information on the development environment and tools used during development.

10. Supporting Information: It provides document history.

2. Introduction

The project is to design a metaverse system used to provide flexibility in voting and election process to non-face-to-face society caused by COVID-19 by implementing four core functions using the VRChat program. The system should allow students to perform the necessary steps of the student council election including campaign and vote. The designs that would be used when implementing the system would be provided in this document and would follow the requirements specified in the Software Requirements Specifications document provided before.

2.1. Objectives

This chapter provides the descriptions and definitions of the tools and diagrams used during the design phase. The program used are online tools called diagrams.net and visual-paradigm.com.

2.2. Applied Diagrams

This section defines the term UML and the types of UML diagrams used in this project.

2.2.1. UML

UML is short for "Unified Modeling Language" and its purpose is to visually represent a design of a system in a standard way. This project will use UML to visualize the design of the system and help readers better understand the system.

2.2.2. Use Case Diagram

Use case diagrams are a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. It is the primary form of system/software requirements for a new software program underdeveloped. It serves as a simplified guide for what the system should do, and does not denote the order in which steps are performed, and only summarizes some of the relationships between use cases, actors, and systems.

2.2.3. Class Diagram

Class diagrams are a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationship among objects. They are the cornerstone of object-oriented modeling as they provide the static structure of classifiers in a system through diagrams that consist of a set of classes and a set of relationships between classes, which help developers and other team members when working on the system.

2.2.4. Sequence Diagram

Sequence diagrams are interaction diagrams that detail how operations are carried out that capture the interaction between the objects in the context of a collaboration. They depict high-level interactions that occur between sub-systems, systems, or the system and its user that either visualize a use case or an operation.

2.2.5. Context Diagram

Context diagrams are used to show the flow of information between the system and external components. They are also the highest or most basic level of a Data Flow Diagram, which is why context diagrams are sometimes referred to as Level 0 Data Flow Diagrams, which are diagrams that tries to document how information flows within a system as a whole by specifying the processes that are involved with transferring data.

2.2.6. Entity Relationship Diagram

Entity Relation Diagram is a way to visually represent how various entities within a system relate to each other. Entities can be any type of business objects used in the system, such as people, roles, events, places, products, and logs.

2.3. Applied tools

2.3.1. StarUML

As a unified modeling language(UML) modeling tool, it can easily and quickly create various diagrams such as Use Case Diagram, Class Diagram, and Sequence Diagram using easy UI. StarUML is a “sophisticated software modeler for agile and concise modeling”

2.3.2. diagrams.net

diagrams.net is a free open-source cross-platform graph drawing software developed with HTML5 and JavaScript. The interface can be used to create diagrams such as flowcharts, wire frames, UML diagrams, organizational charts, and network diagrams.

2.4. Project Scope

This service was made to accommodate online school council election & voting processes using metaverse due to the current pandemic. This “School Council Election & Voting using Metaverse” project is entirely based on the VRChat platform, and the content to be implemented does not have the nature of cross-platform and relies on that specific VRChat platform.

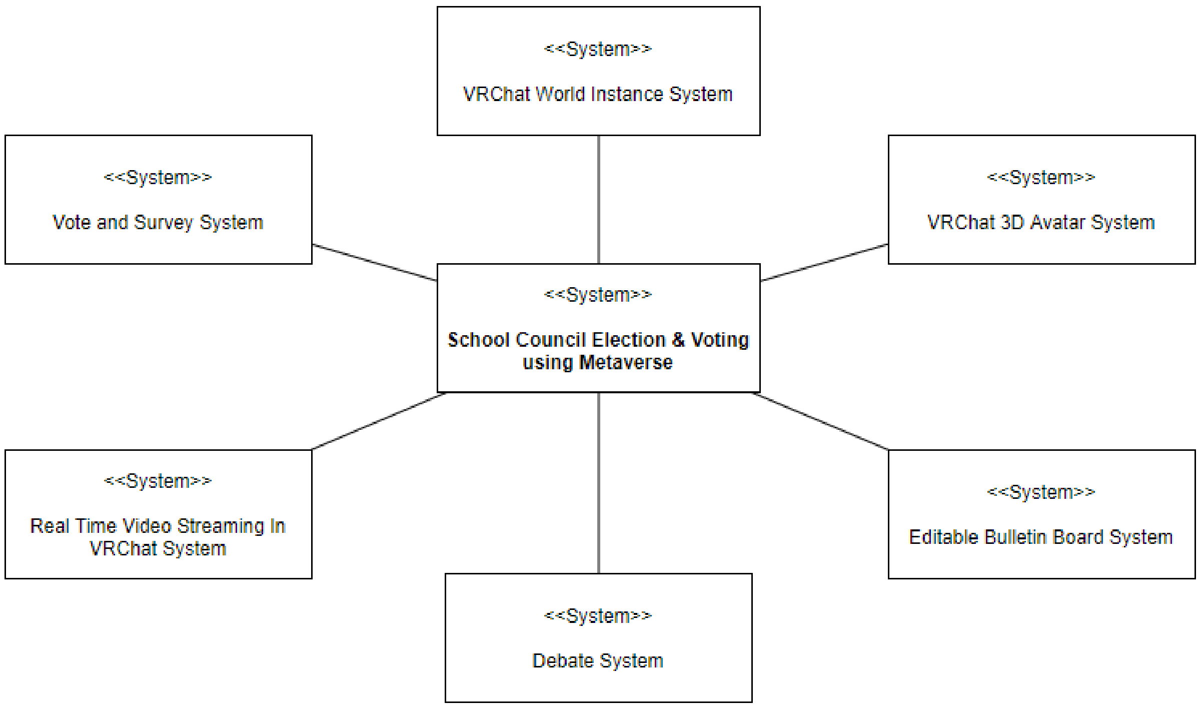
3. System Architecture - Overall

3.1. Objectives

The objective of this chapter is to provide information on how the system is organized

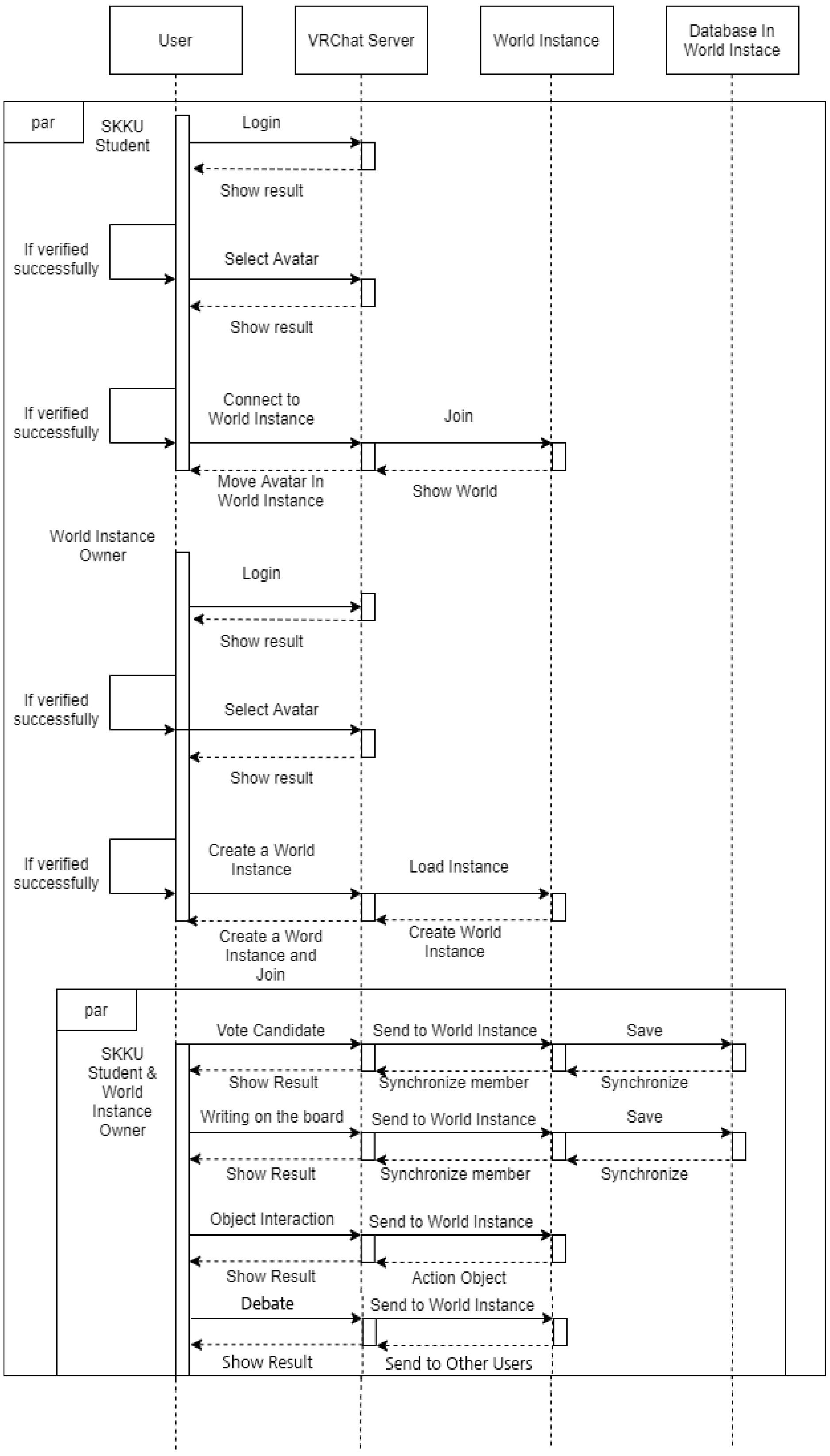
3.2 System Organization

3.2.1 Context Diagram


<<System>>
VRChat World Instance System
<<System>>
Vote and Survey System
<<System>>
VRChat 3D Avatar System
<<System>>
School Council Election & Voting using Metaverse
<<System>>
Real Time Video Streaming In VRChat System
<<System>>
Editable Bulletin Board System
<<System>>
Debate System

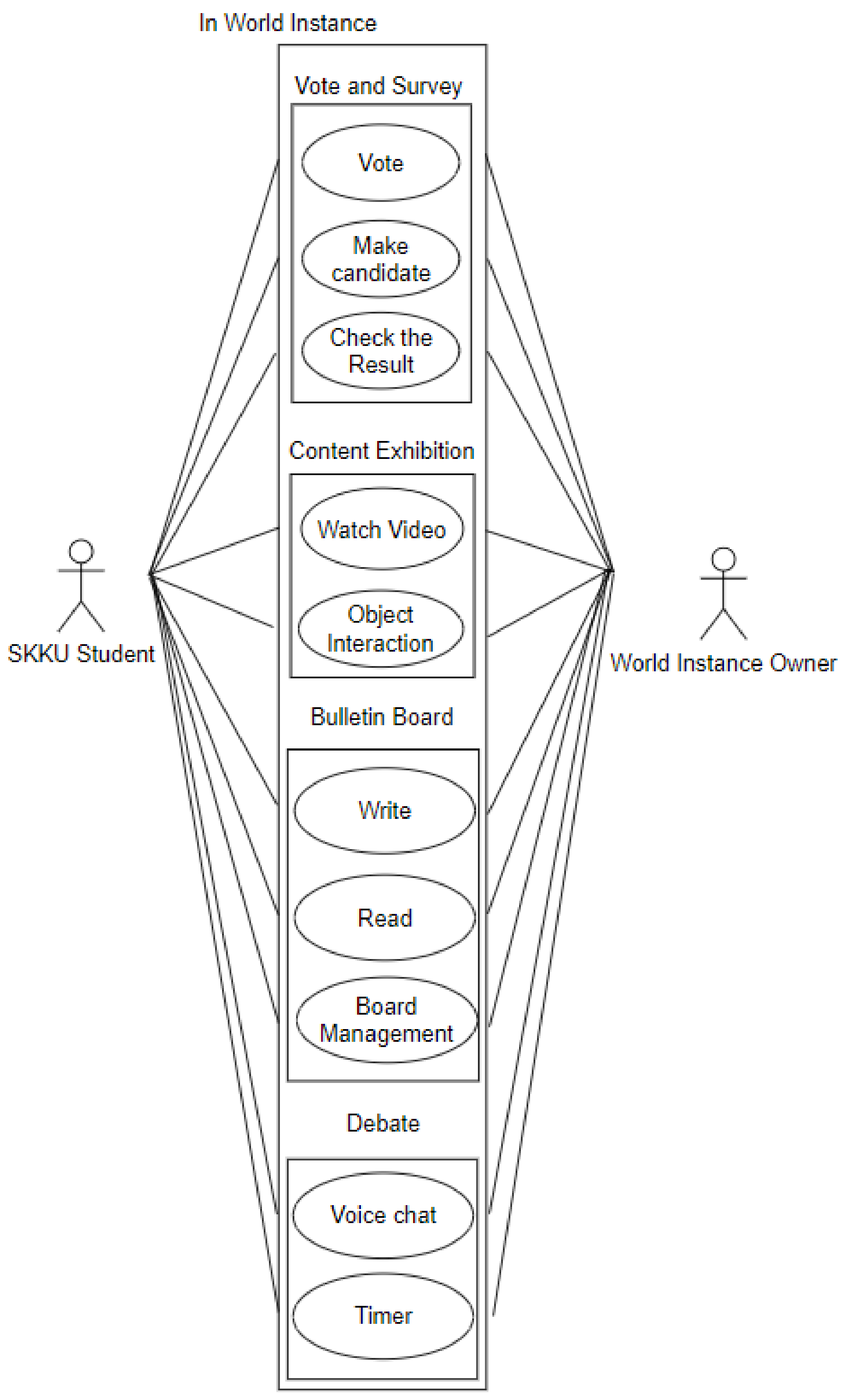
[Figure 1] Overall Context Diagram

3.2.2 Sequence Diagram



[Figure 2] Overall Sequence Diagram

3.2.3 Use Case Diagram



[Figure 3] Use Case Diagram

4. System Architecture - Frontend

4.1. Objectives

This chapter describes the conceptual model that defines the structure, behavior, and more views of the frontend of a system. It consists of system components and the sub-systems developed, that will work together to implement the overall system.

4.2. Subcomponents

4.2.1 White Board

Whiteboard class is a class that displays information such as ongoing agenda and current debate stage to participants in the debate forum. It is visible to all participants, but only the chairperson should be able to manipulate the whiteboard.

4.2.1.1. Attribute

These are the attributes that the white board class has.

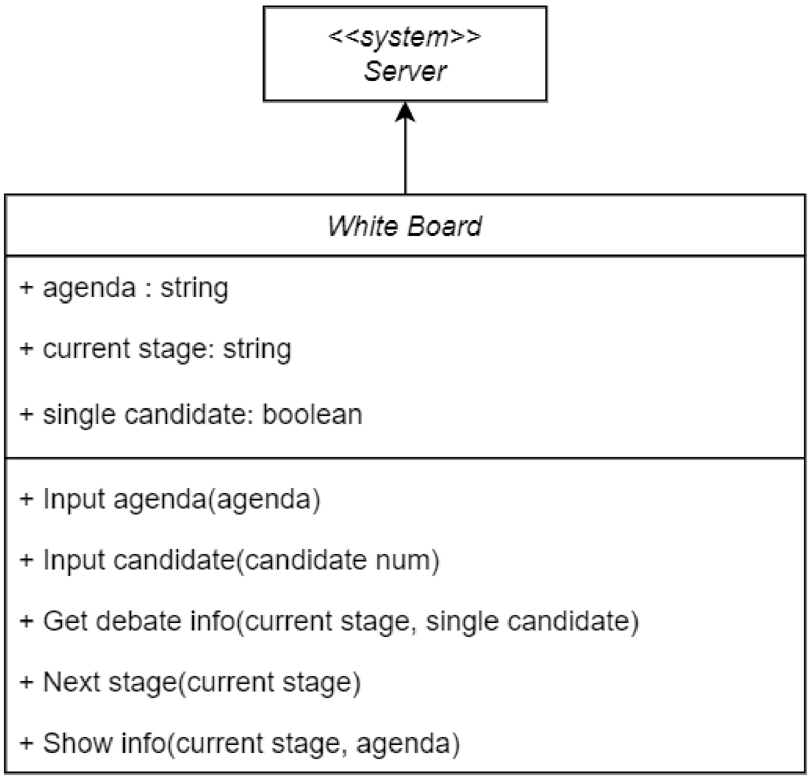
- Agenda: Agenda to be discussed at this stage
- Current stage: Current stage of discussion
- Single candidate: Indicates whether or not candidate is single team.

4.2.1.2. Methods

These are the methods that the white board class has.

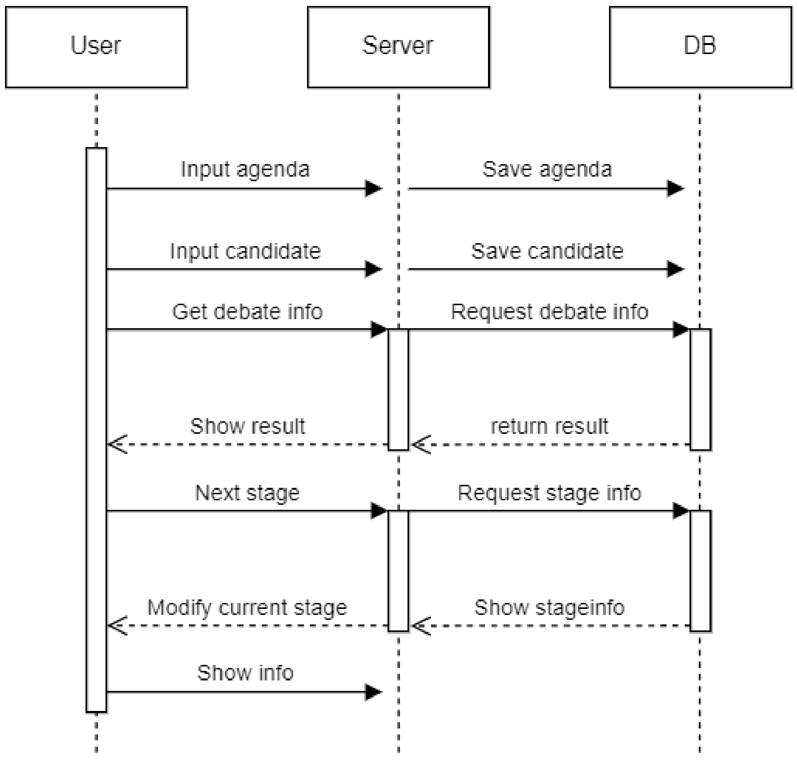
- Input agenda()
- Input candidate()
- Get debate info()
- Next stage()
- Show info()

4.2.1.3. Class Diagram


<<system>>
Server
White Board
+ agenda : string
+ current stage: string
+ single candidate: boolean
+ Input agenda(agenda)
+ Input candidate(candidate num)
+ Get debate info(current stage, single candidate)
+ Next stage(current stage)
+ Show info(current stage, agenda)

[Figure 4] Class diagram – White Board

4.2.1.4 Sequence Diagram



[Figure 5] Sequence Diagram – White Board

4.2.2. Timer

Timer class is a class to adjust the speaking time of speakers for smooth progress of discussion. It displays the remaining speaking time. When the speaking time ends, the speaker will no longer be able to speak. Just like the whiteboard class, only the chairperson can control it.

4.2.2.1. Attribute

These are the attributes that the timer class has.

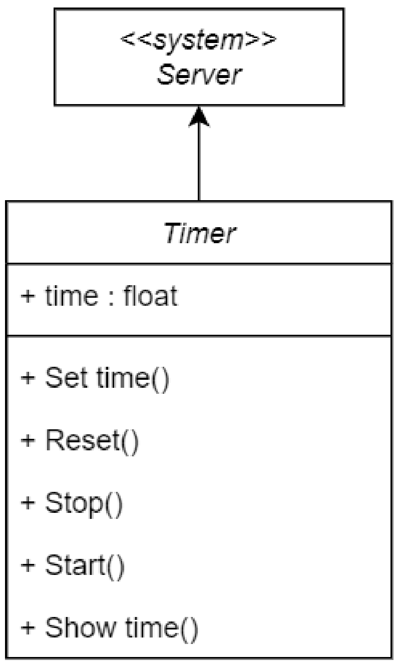
- time: means speaking time.

4.2.2.2. Methods

These are the methods that the timer class has.

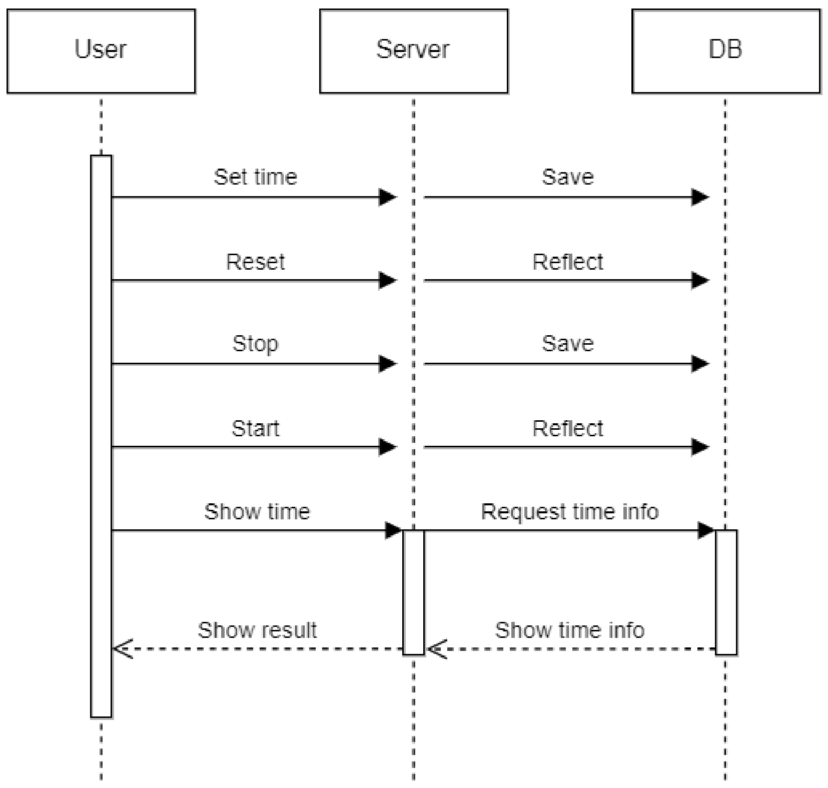
- Set time()
- Reset()
- Stop()
- Start()
- Show time()

4.2.2.3. Class Diagram



[Figure 6] Class diagram – White Board

4.2.2.4 Sequence Diagram



[Figure 7] Sequence Diagram – White Board

4.2.3. Request to speak

The Request to speak class is used for Q&A session between the candidate and the audience. Any audience who wants to ask a question to the candidate will uses this class to indicate to the chairperson that they want to speak. Among them, the chairperson will select the speaker. Since the press question time and the student question time are separated, only the press can speak during the press time and only the student can speak during the student time.

4.2.3.1. Attribute

These are the attributes that the Request to speak class has.

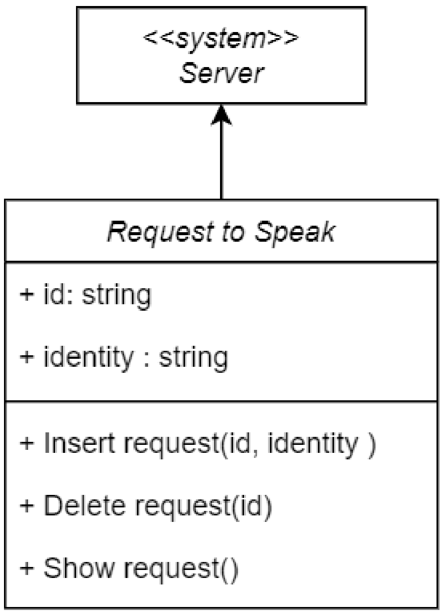
- Id: the id of the person who expressed his/her intention to speak
- Identity: Identifies whether the person who expressed the intention to speak is a press or a student

4.2.3.2. Methods

These are the methods that the Request to speak class has.

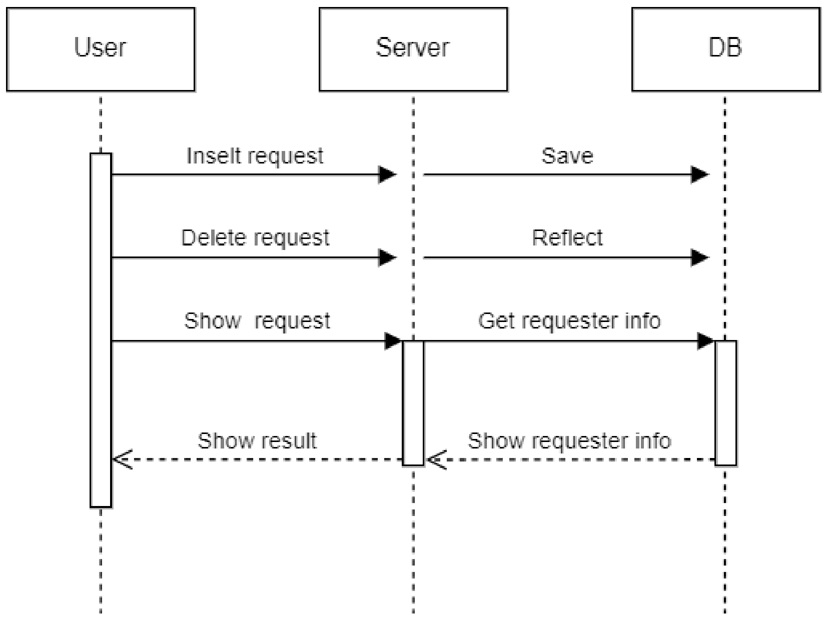
- Insert request()
- Delete request()
- Show request()

4.2.3.3. Class Diagram



[Figure 8] Class diagram – White Board

4.2.3.4 Sequence Diagram



[Figure 9] Sequence Diagram – White Board

4.2.4 Visual content

Voters can watch videos, 2D contents and 3D contents. Those contents are loaded from database that already uploaded.

4.2.4.1 Attributes

These are the attributs that the visual content has.

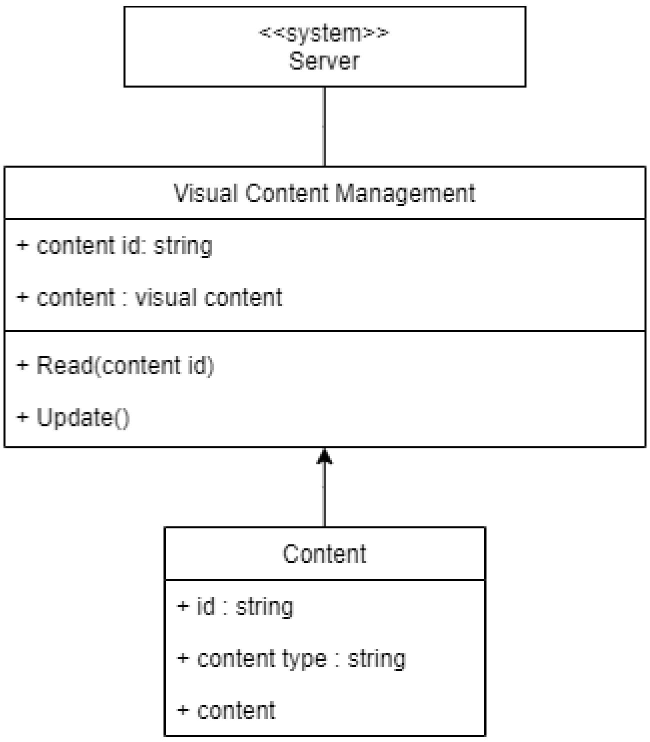
- Content ID : ID of each contents
- Content : content of the visual content

4.2.4.2 Methods

These are the methods that the visual content has.

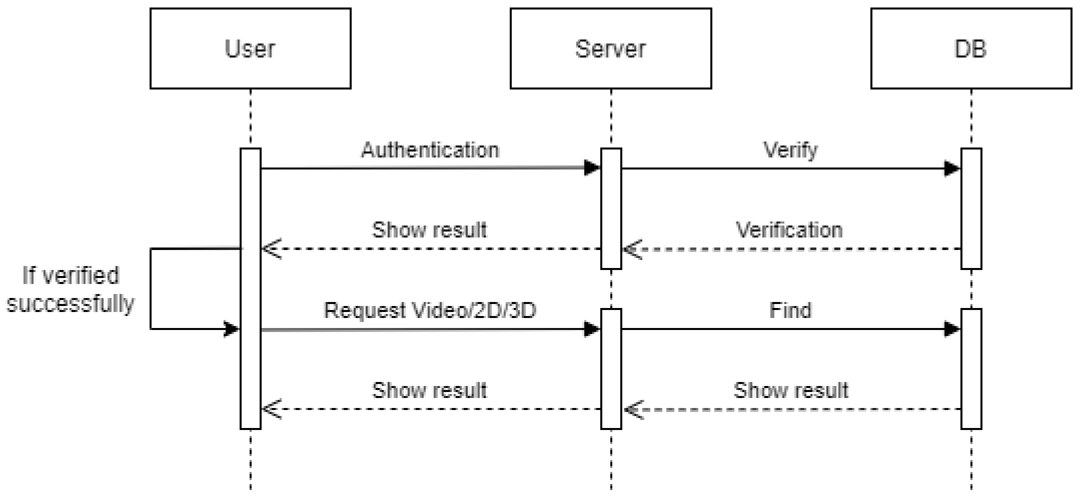
- Read()
- Update()

4.2.4.3 Class Diagram



[Figure 10] Class Diagram – Visual content

4.2.4.4 Sequence Diagram



[Figure 11] Sequence Diagram – Visual content

4.2.4 Notice Board

Chairperson can write a notice on the board. Notices may contain any information related to voting or elections. Submitted notices are freely viewable by all.

4.2.4.1 Attributes

These are the attributes that the notice board class has.

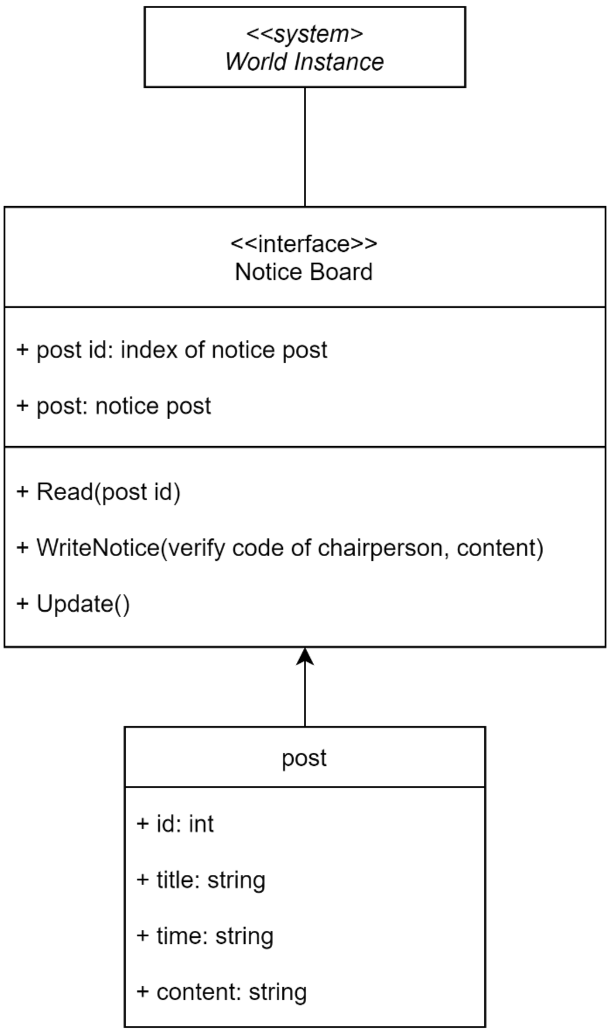
- Post id: index of the notice post
- Title: title of the notice post
- Time: time the notice post was written
- Content: content of notice post

4.2.4.2. Methods

These are the methods that the notice board class has.

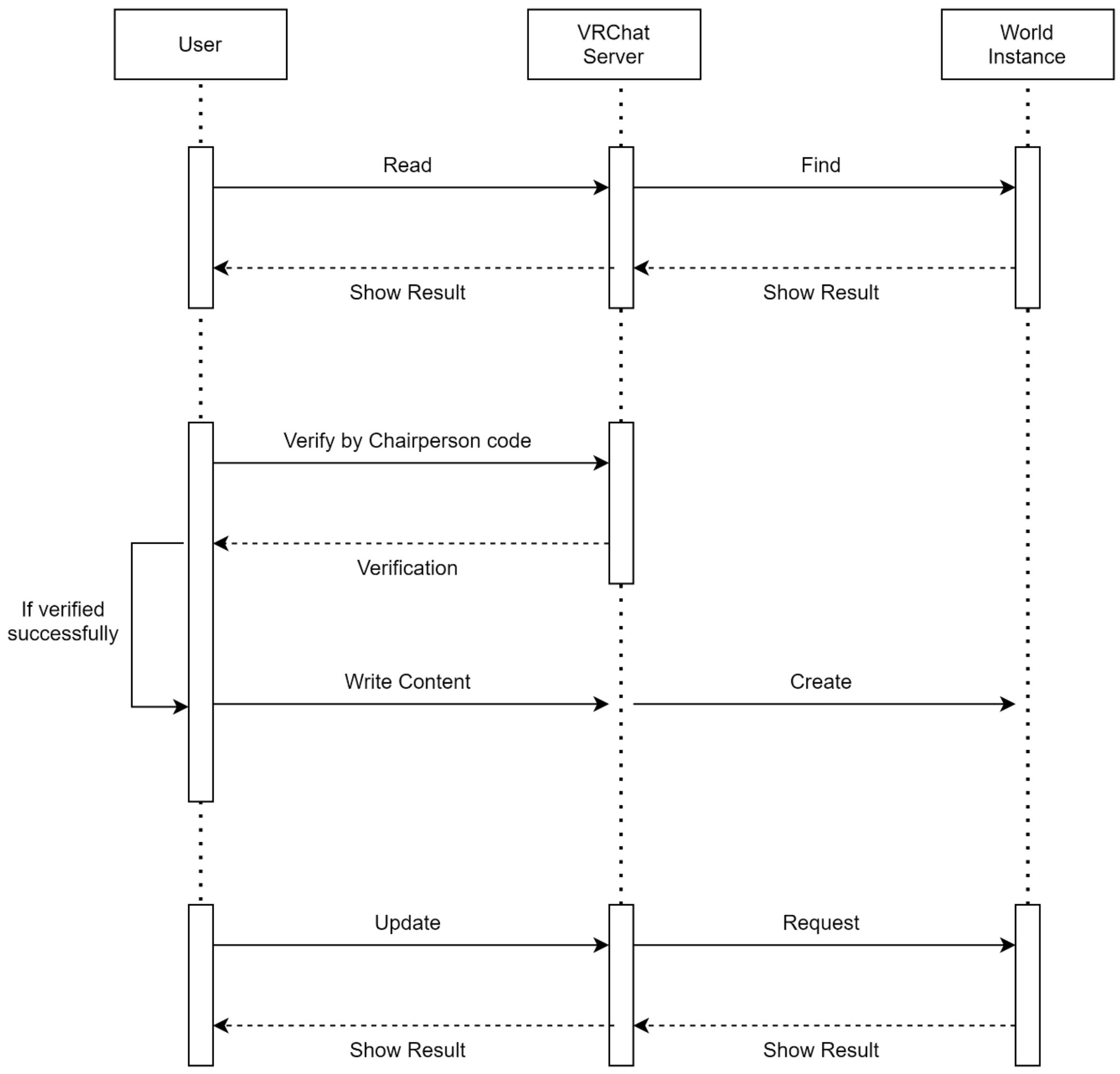
- Read()
- WriteNotice()
- Update()

4.2.4.3. Class Diagram



[Figure 12] Class Diagram – Notice Board

4.2.4.4. Sequence Diagram



[Figure 13] Sequence Diagram – Notice Board

4.2.5 Q&A Board

Voters can write questions to the Q&A board. Questions may contain any information related to voting or elections. Submitted questions are freely viewable by all. Also, Candidates can view the questions written on the Q&A board and write the corresponding answers. Submitted answers are freely viewable by all.

4.2.5.1 Attributes

These are the attributes that the Q&A board class has.

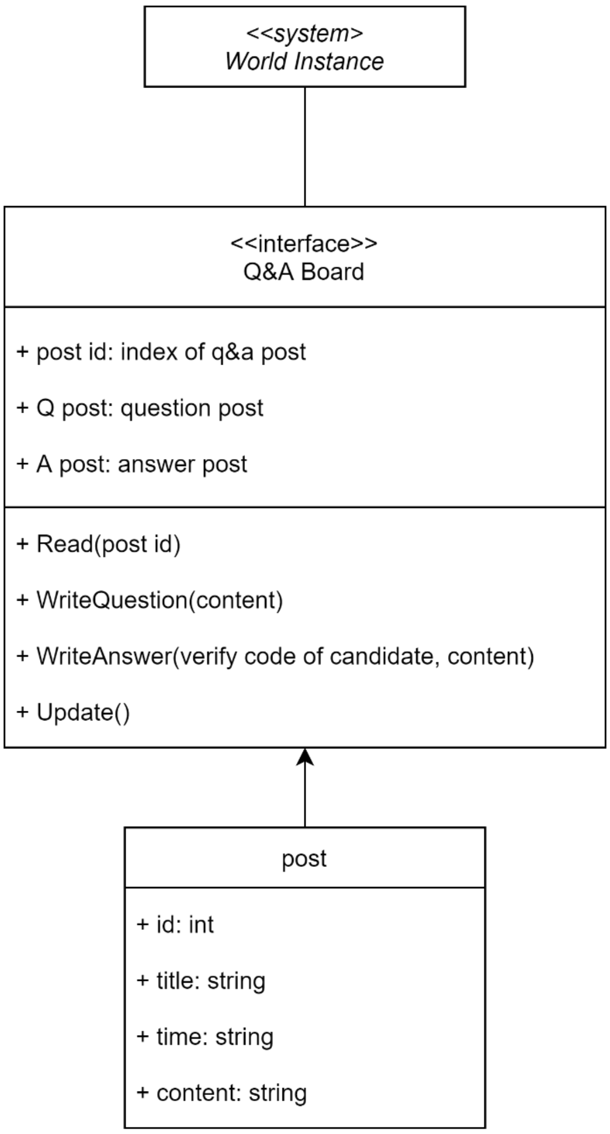
- Post id: index of the question post
- Title: title of the question post
- Time: time the question post was written
- Content: content of question post
- Post id: index of the answer post
- Title: title of the answer post
- Time: time the answer post was written
- Content: content of answer post

4.2.5.2. Methods

These are the methods that the Q&A class has.

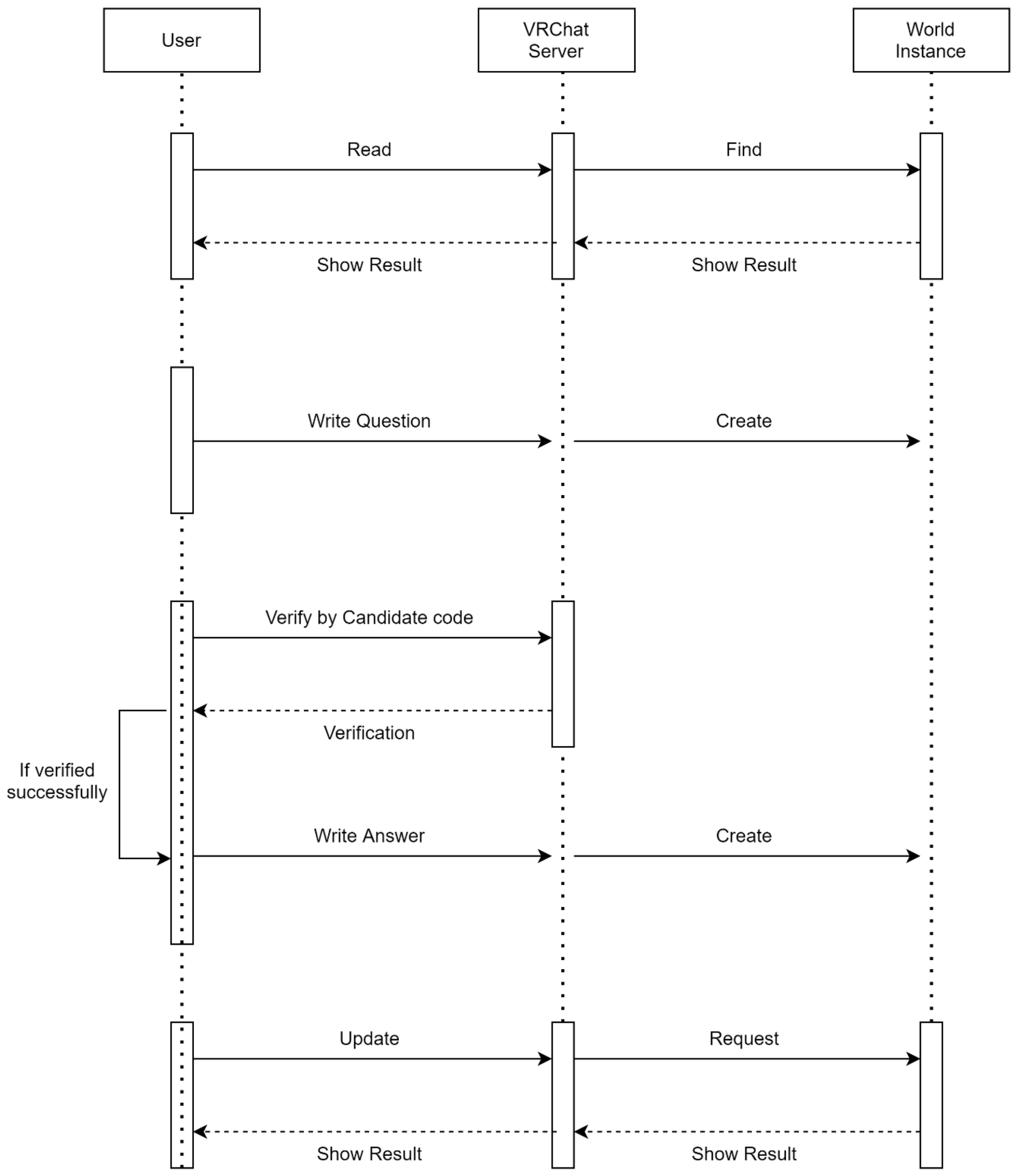
- Read()
- WriteQuestion()
- WriteAnswer()
- Update()

4.2.5.3. Class Diagram



[Figure 14] Class Diagram – Q&A Board

4.2.5.4. Sequence Diagram



[Figure 15] Sequence Diagram – Q&A Board

4.2.6 Free Board

Anyone can write questions to the free board. Submitted posts are freely viewable by all.

4.2.6.1 Attributes

These are the attributes that the free board class has.

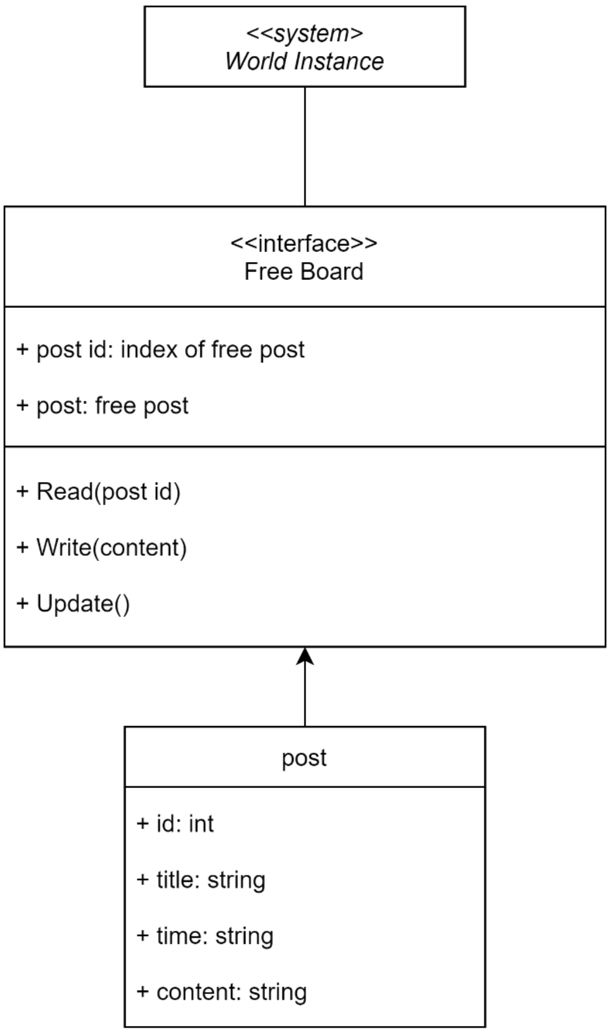
- Post id: index of the free post
- Title: title of the free post
- Time: time the free post was written
- Content: content of free post

4.2.6.2. Methods

These are the methods that the Q&A class has.

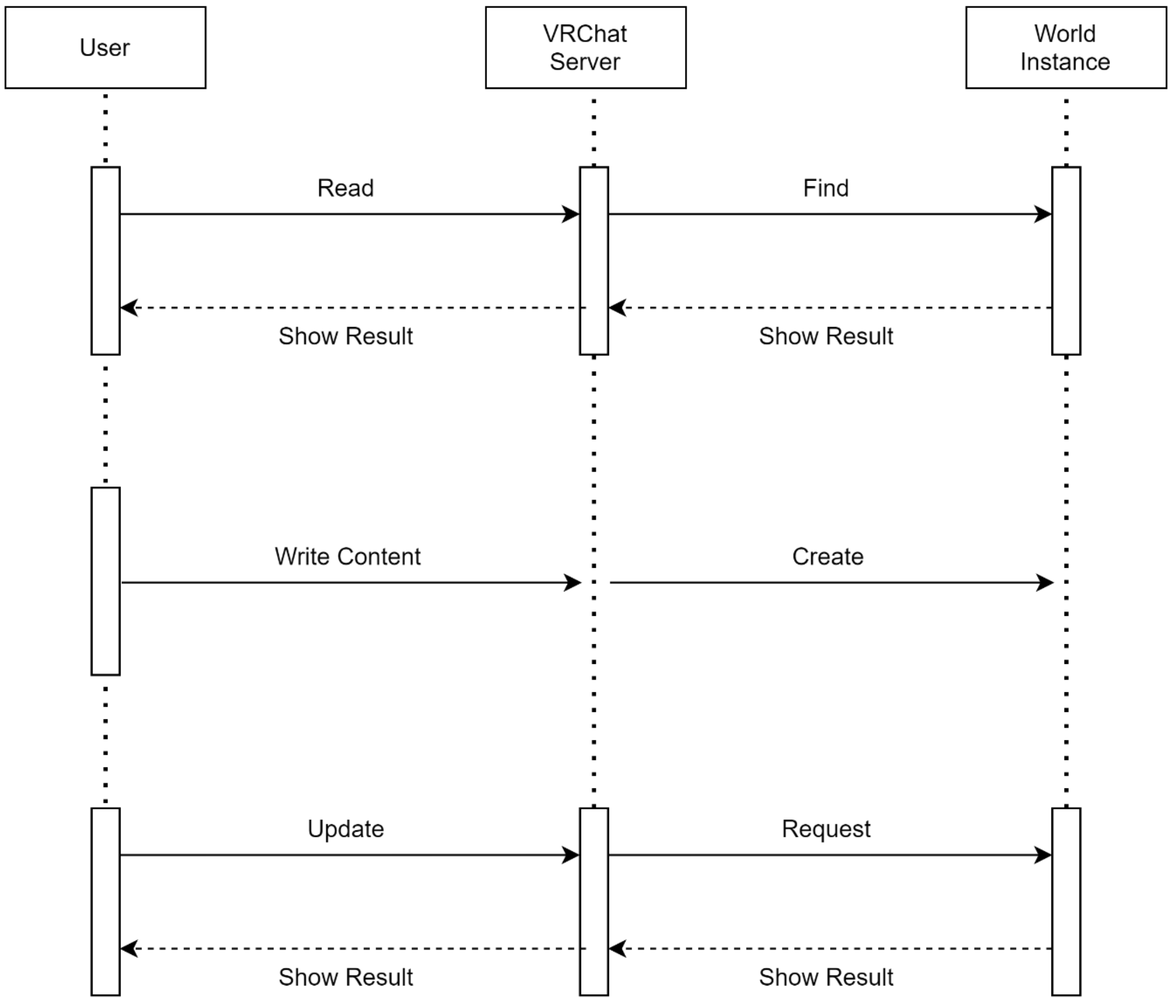
- Read()
- Write()
- Update()

4.2.6.3. Class Diagram



[Figure 16] Class Diagram – Free Board

4.2.6.4. Sequence Diagram



[Figure 17] Sequence Diagram – Free Board

4.2.7. Voting process management

Voting process management deals with organizations. This class allows the voting process to be created only for certified organizations. After receiving the details of the voting process, the voting process is created only if the number of allowed processes remains in the process validity period.

4.2.7.1. Attributes

These are the attributes that the voting process management class has.

- Organization Id: Id of the organization represented by the user

- Purpose: purpose of the voting process

- Target: target voters (participants)

- Period: validity voting period of the process

- Voting items: registered candidates

4.2.7.2. Methods

These are the methods that the voting process management class has.

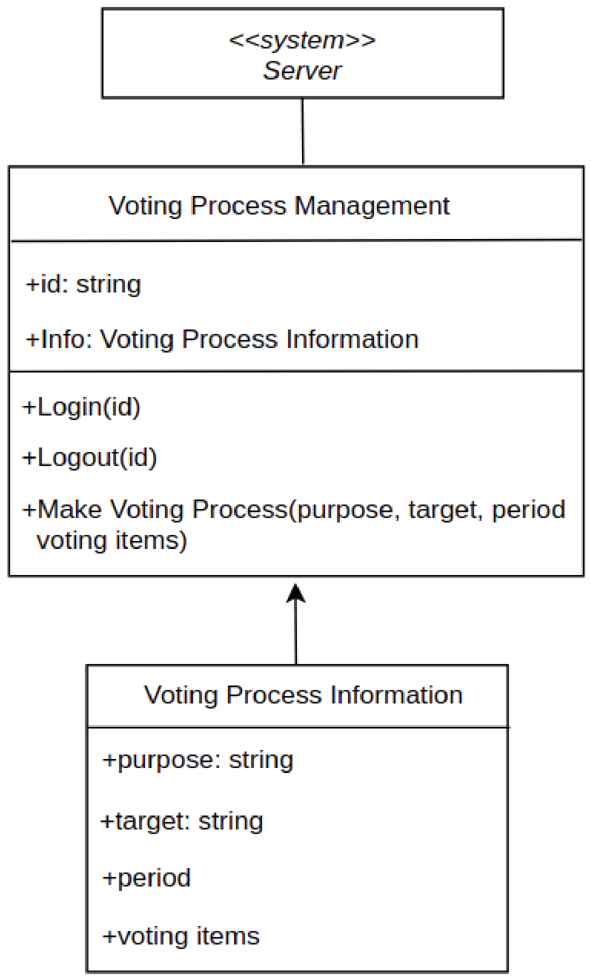
- Login()

- Logout()

- Make Voting Process()

- Register()

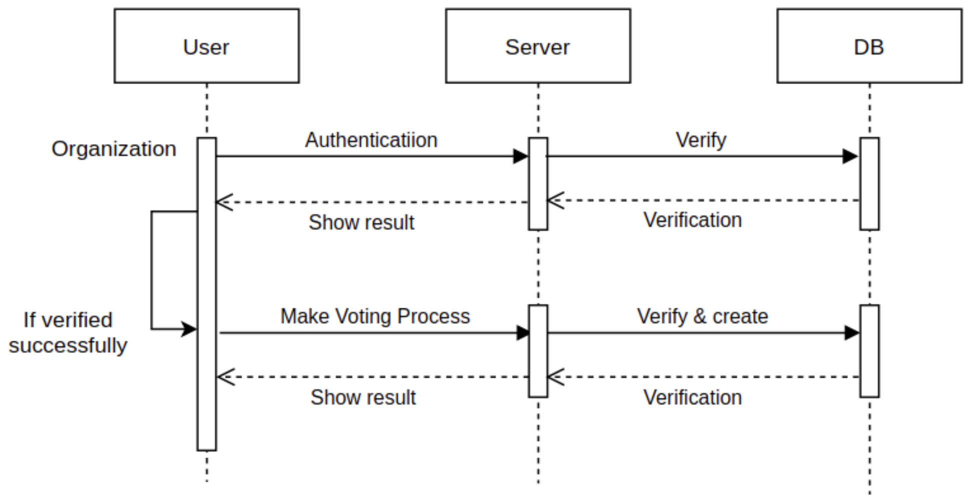
4.2.7.3. Class Diagram



<<system>>
Server
Voting Process Management
+id: string
+Info: Voting Process Information
+Login(id)
+Logout(id)
+Make Voting Process(purpose, target, period voting items)
Voting Process Information
+purpose: string
+target: string
+period
+voting items

[Figure 18] Class Diagram – Voting Process Management

4.2.7.4. Sequence Diagram



[Figure 19] Sequence Diagram – Voting Process Management

4.2.8. Vote

Vote class deals with users who are voting. This class gets the process id and SSN and checks whether the user is the target voter of the process. Users enter the room, vote, and submit the response.

4.2.8.1. Attributes

These are the attributes that the vote class has.

- Process id: Voting process that the user wants to participate in

- SSN: Student school number

- Contents: response to questions

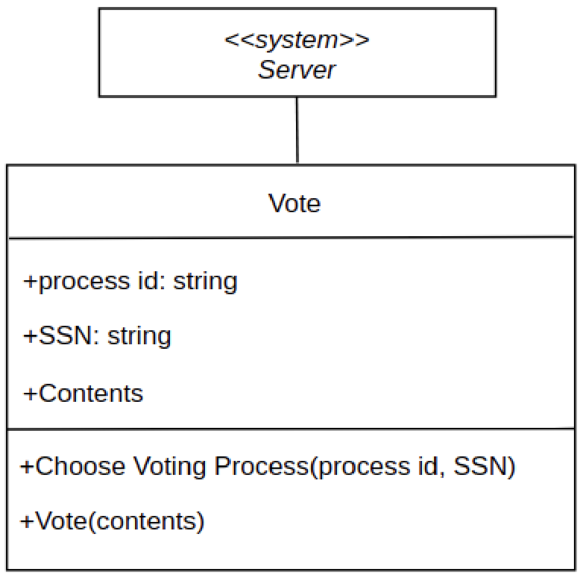
4.2.8.2. Methods

These are the methods that the vote class has.

- Choose Voting Process()

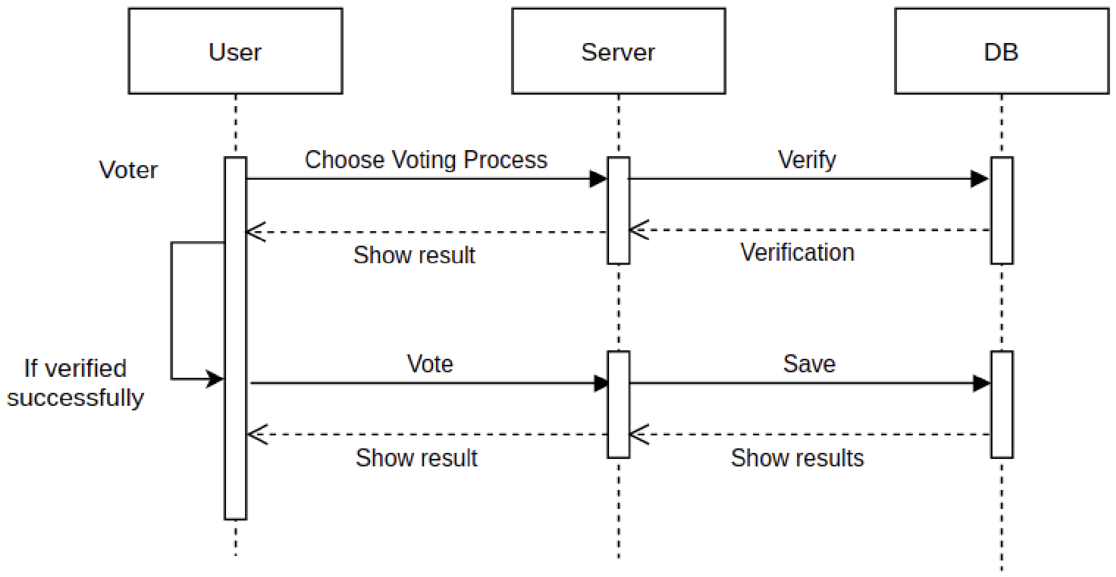
- Vote()

4.2.8.3. Class Diagram



[Figure 20] Class Diagram – Vote

4.2.8.4. Sequence Diagram



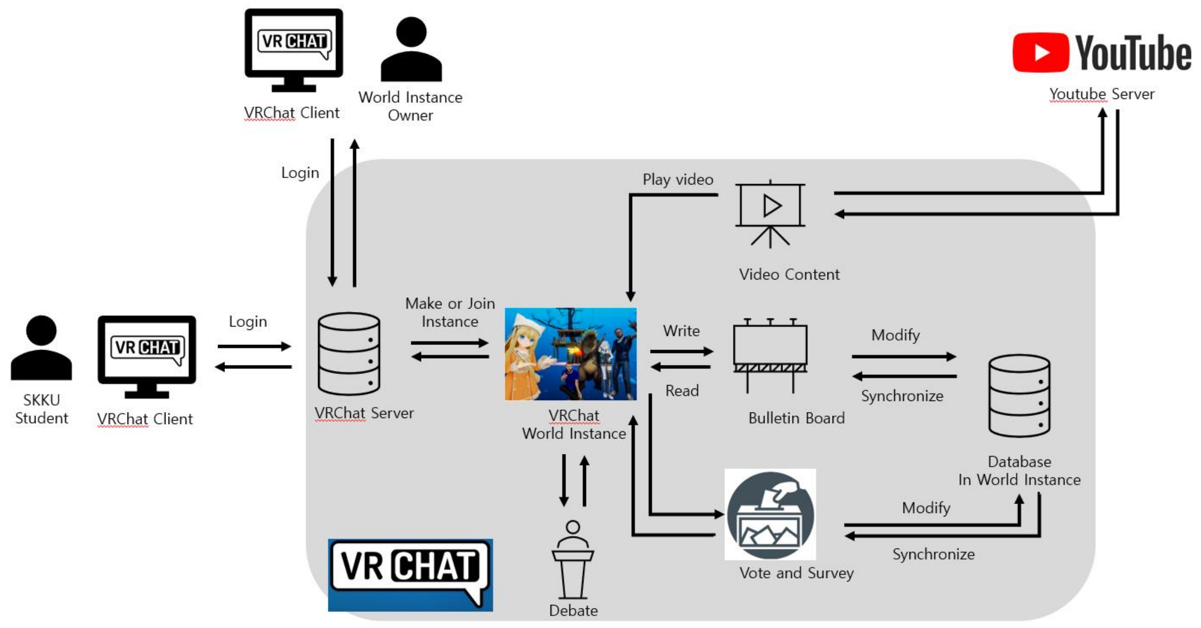
[Figure 21] Sequence Diagram – Vote

5. System Architecture - Backend

5.1. Objectives

This chapter describes the structure of the back-end system include DB in our project.

5.2. Overall Architecture

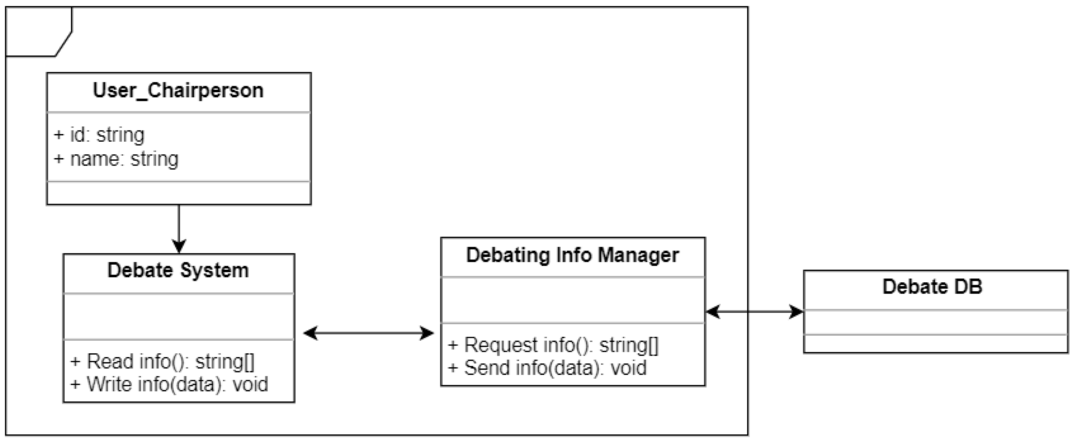


[Figure 22] Overall Architecture

5.3. Subcomponents

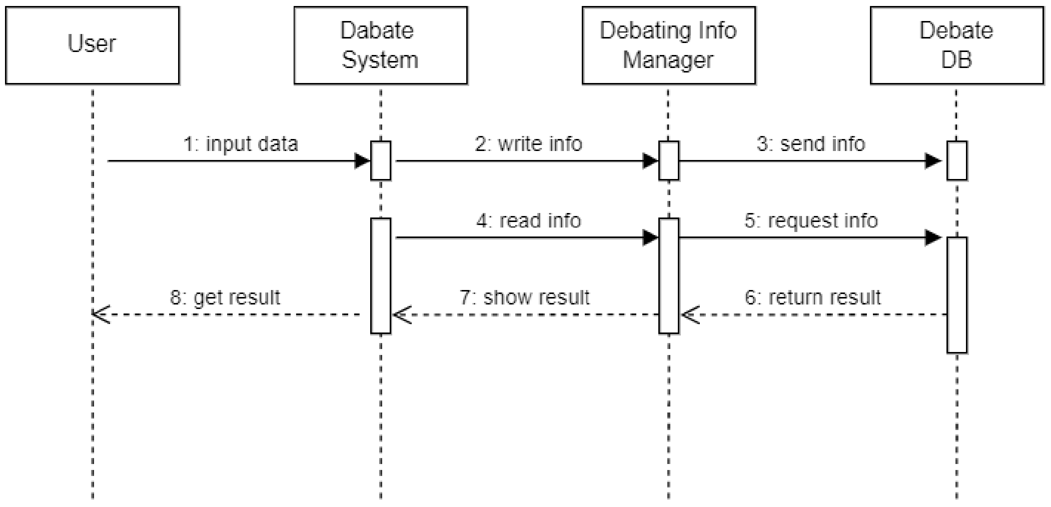
5.3.1. Debating Function

5.3.1.1. Class diagram



[Figure 23] Class Diagram – Debating Function

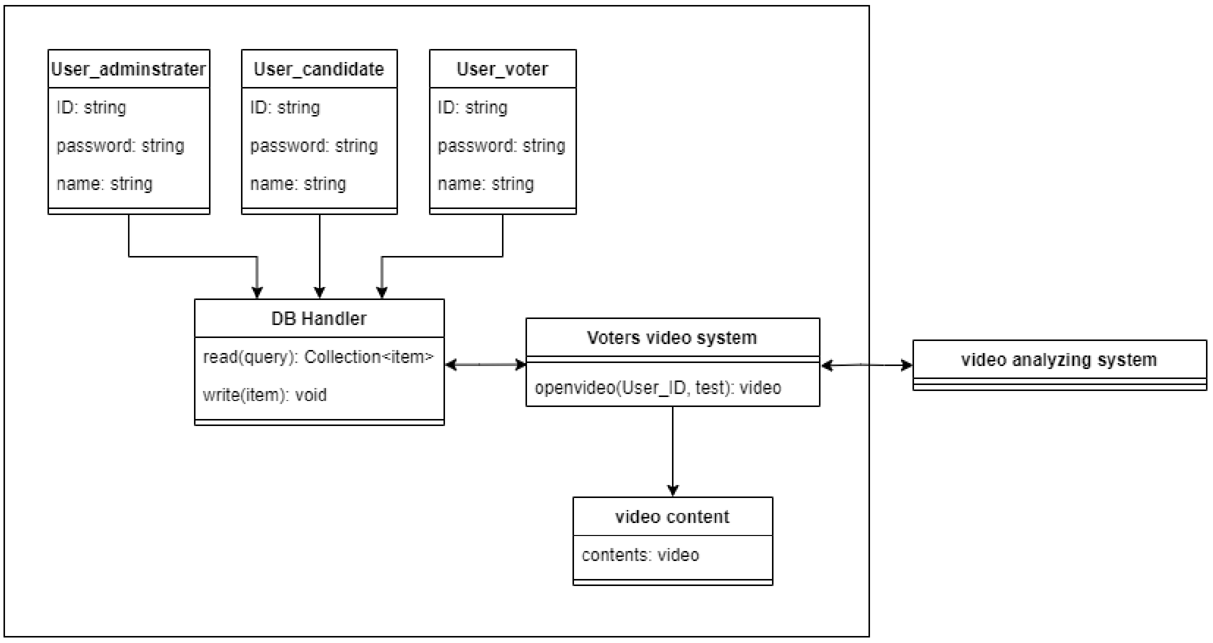
5.3.1.2. Sequence Diagram



[Figure 24] Sequence Diagram – Debating Function

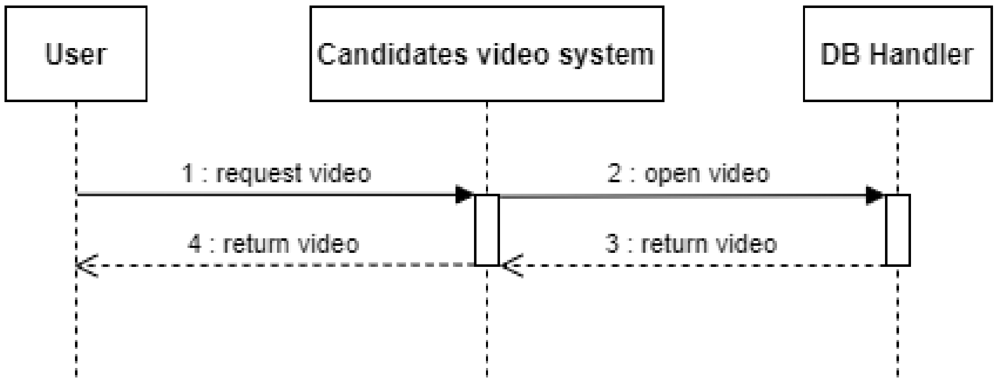
5.3.2. Visual Content Function

5.3.2.1. Class diagram



[Figure 25] Class Diagram – Video content

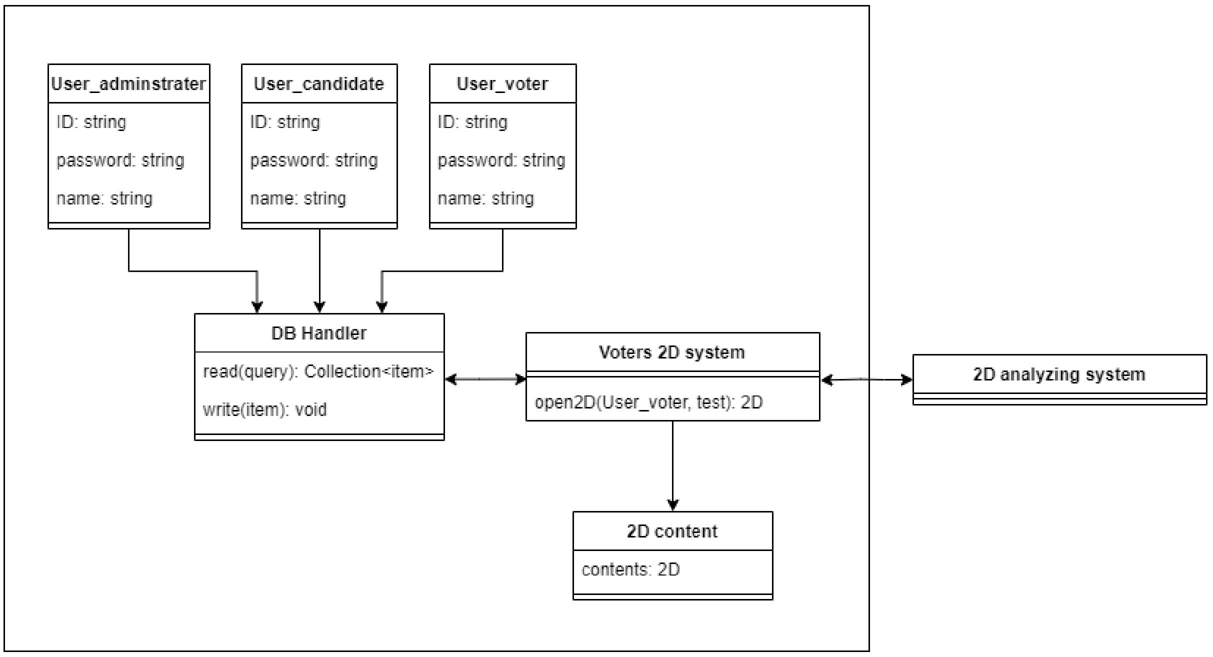
5.3.2.2. Sequence diagram



[Figure 26] Sequence Diagram – Video content

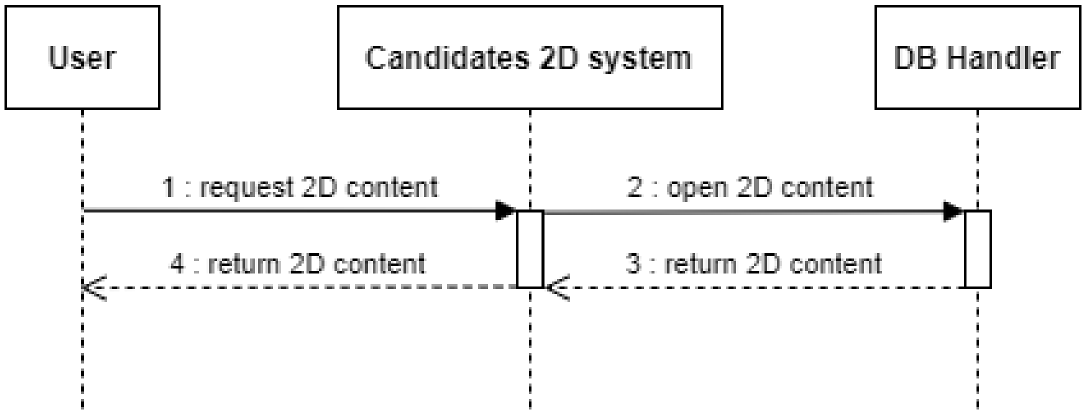
5.3.3. 2D Content Function

5.3.3.1. Class diagram



[Figure 27] Class Diagram – 2D content

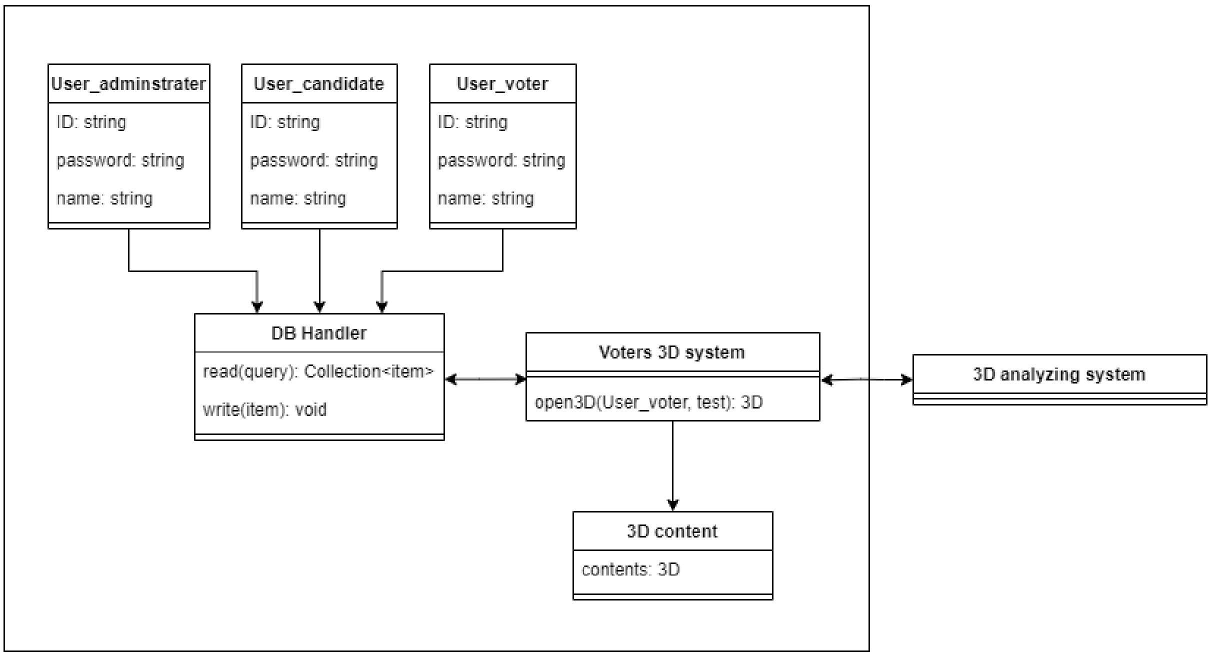
5.3.3.2. Sequence diagram



[Figure 28] Sequence Diagram – 2D content

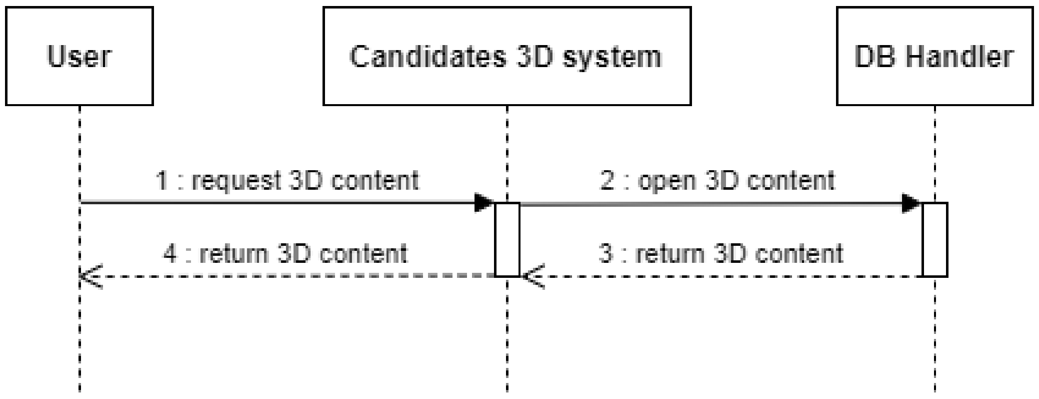
5.3.4. 3D Content Function

5.3.4.1. Class diagram



[Figure 29] Class Diagram – 3D content

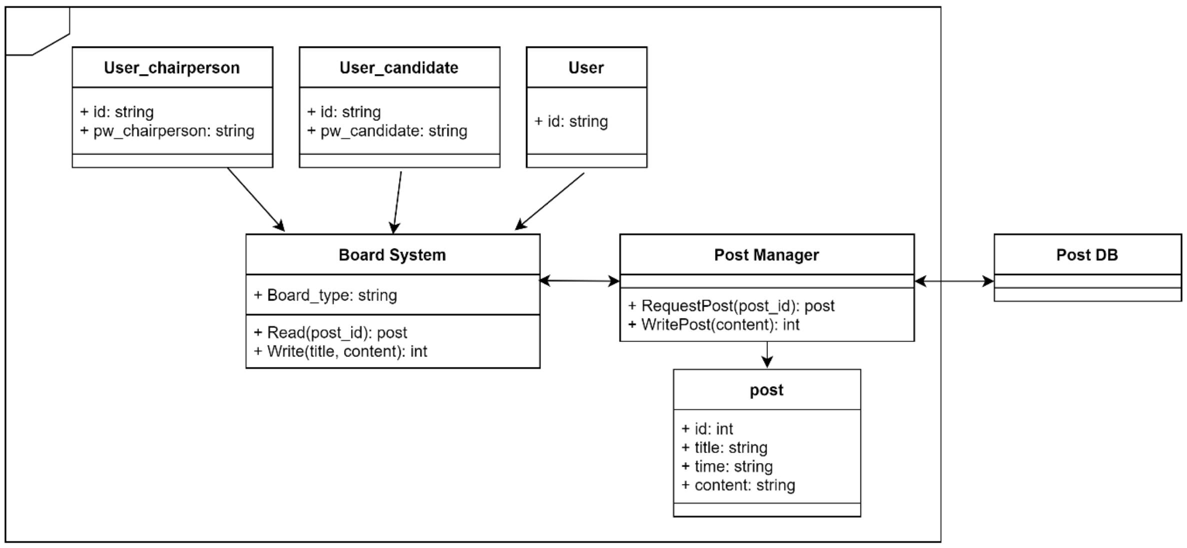
5.3.4.2. Sequence diagram



[Figure 30] Sequence Diagram – 3D content

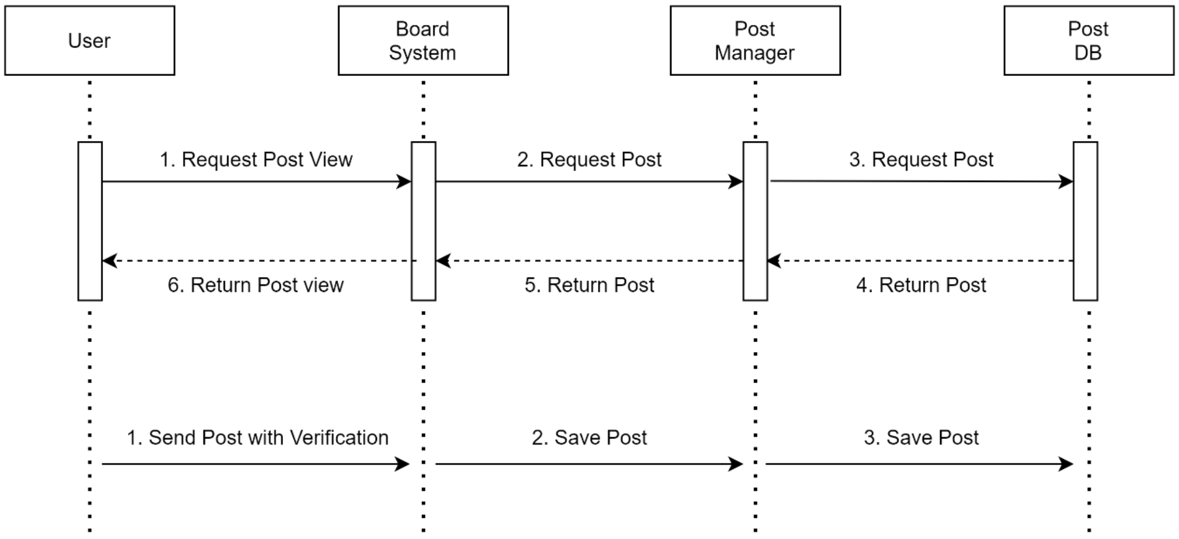
5.3.5. Board

5.3.5.1. Class Diagram



[Figure 31] Class Diagram – Board Function

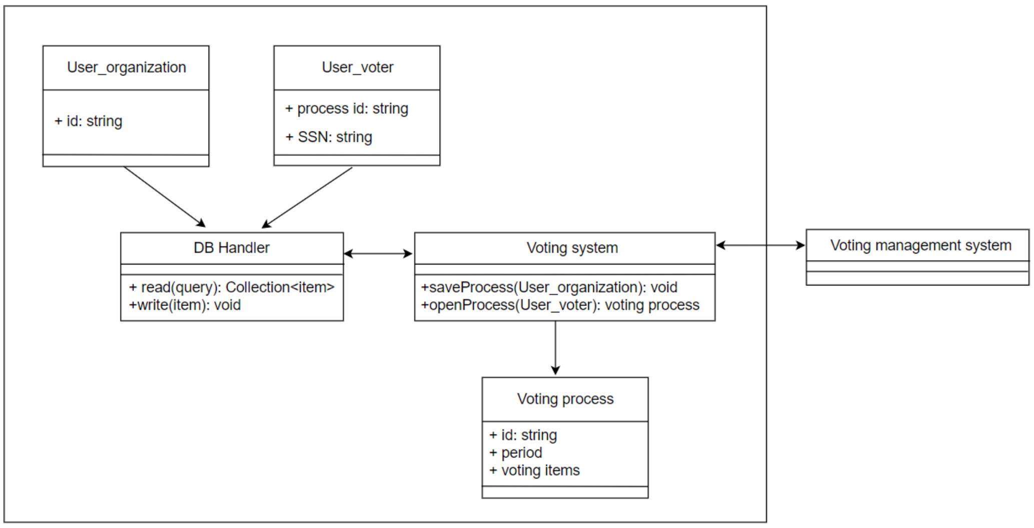
5.3.5.2. Sequence Diagram



[Figure 32] Sequence Diagram – Board Function

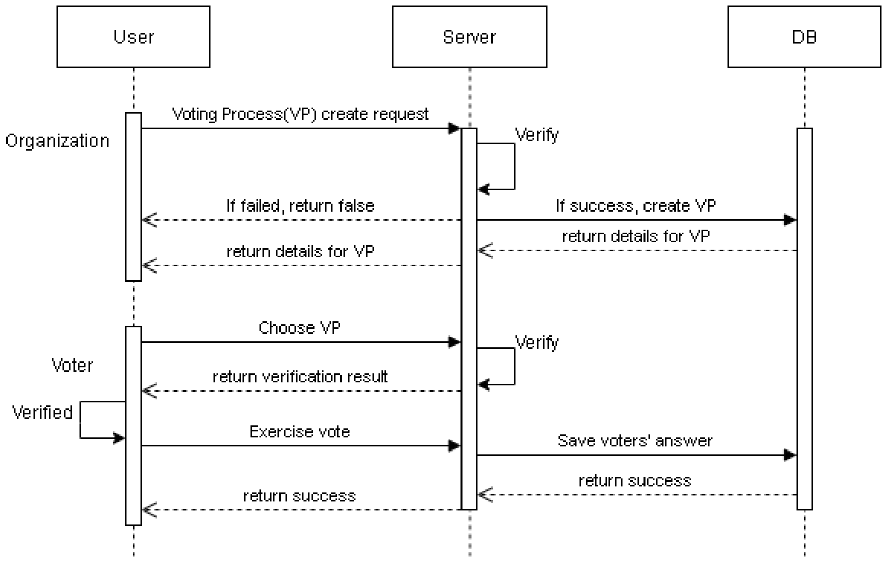
5.3.6. Voting

5.3.6.1. Class diagram



[Figure 33] Class Diagram – Voting

5.3.6.2. Sequence diagram



[Figure 34] Sequence Diagram – Voting

6. Protocol Design

6.1. Objectives

In this chapter, we cover the structure for the protocol used for the interaction of subsystems such as web pages, servers, DBs, and applications. This chapter also talks about how to define interfaces.

6.2. 2D Object Files

2D objects that includes images and texts are composed of standard image files such as jpg, png, gif, etc. All such components are supported by Unity engine. These files are familiar to voters. Voters can see candidates' election promises or their history. 2D objects could be images, text, etc. Our goal is to give candidates administrators which can upload their objects right away.

6.3. 3D Object Files

3D objects are the strengths for this metaverse system. Users can go inside the VRChat and see their promises three-dimensionally. For instance, if candidates promise to redesign some rooms or spaces, they can give view maps to voters. Voters can directly see the before/after and imagine the real scene. 3D objects can be uploaded by .fbx or .obj files. It can also be edited by using MeshRenderer. All the components can be supported by Unity Engine.

6.4. Video Files

Videos are a good medium for candidates to convince voters. It can give additional and certain explanations to voters. There will be added videos and live streaming service for voters. Our goal is to get videos from urls such as Youtube. Candidates will upload their videos on Youtube. Then, we will use Udon Video Sync Player and add URLs. It will play continuously. Also, for videos that have been already made before the metaverse opens, we will use prefabs provided by VRChat SDK. It can reduce the burden from VRChat

6.5. White Board

6.5.1. Input agenda

- Request

[Table 1] Table of input agenda request of white board

Attribute	Detail	
Method	Input agenda	
URL	/admin/debate	
Parameter	agenda	Agenda name string

- Response

[Table 2] Table of input agenda response of white board

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	White board object	Reflect parameter agenda name string
Failure Response Body	Message	Fail message

6.5.2. Input candidate

- Request

[Table 3] Table of input candidate request of white board

Attribute	Detail	
Method	Input candidate	
URL	/admin/debate	
Parameter	Candidate	Candidate name list

- Response

[Table 4] Table of input candidate response of white board

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	White board object	Reflect parameter candidate name list
Failure Response Body	Message	Fail message

6.5.3. Get debate info

- Request

[Table 5] Table of get debate info request of white board

Attribute	Detail
Method	Get debate info

URL	/admin/debate	
Parameter	Query	The query for getting information of debating session
	Current stage	value of current stage string
	Single candidate	Boolean option whether Single candidate is or not

- Response

[Table 6] Table of get debate info response of white board

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Debate info	The context according to current stage with debating
Failure Response Body	Message	Fail message

6.5.4. Next stage

- Request

[Table 7] Table of next stage request of white board

Attribute	Detail
Method	Next stage

URL	/admin/debate	
Parameter	Current stage	Value of current stage string

- Response

[Table 8] Table of next stage response of white board

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Next stage object	Change current stage into next stage and Return context of following stage
Failure Response Body	Message	Fail message

6.5.5. Show info

- Request

[Table 9] Table of show info request of white board

Attribute	Detail	
Method	Show info	
URL	/admin/debate	
Parameter	Query	The query for requesting information about agenda

	Current stage	value of current stage string
	agenda	Value of agenda name string

- Response

[Table 10] Table of show info response of white board

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Agenda information object	The context object according to agenda information
Failure Response Body	Message	Fail message

6.6. Timer

6.6.1. Set time

- Request

[Table 11] Table of set time request of timer

Attribute	Detail
Method	Set time
URL	/admin/debate
Parameter	-

- Response

[Table 12] Table of set time response of timer

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.6.2. Reset

- Request

[Table 13] Table of reset request of timer

Attribute	Detail
Method	Reset
URL	/admin/debate
Parameter	-

- Response

[Table 14] Table of reset response of timer

Attribute	Detail

Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Timer object	Reflected in time attribute
Failure Response Body	Message	Fail message

6.6.3. Stop

- Request

[Table 15] Table of stop request of timer

Attribute	Detail
Method	Stop
URL	/admin/debate
Parameter	-

- Response

[Table 16] Table of stop response of timer

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Timer object	Object state change
	Message	Message informing time stop

Failure Response Body	Message	Fail message

6.6.4. Start

- Request

[Table 17] Table of start request of timer

Attribute	Detail
Method	Start
URL	/admin/debate
Parameter	-

- Response

[Table 18] Table of start response of timer

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Timer object	Object state change
	Message	Message informing time start
Failure Response Body	Message	Fail message

6.6.5. Show time

- Request

[Table 19] Table of show time request of timer

Attribute	Detail	
Method	Show time	
URL	/admin/debate	
Parameter	Query	The query for getting information of time

- Response

[Table 20] Table of show time response of timer

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	White board object	showing time attribute
Failure Response Body	Message	Fail message

6.7. Request to speak

6.7.1. Insert request

- Request

[Table 21] Table of insert request of request to speak

Attribute	Detail

Method	Insert request	
URL	/admin/debate	
Parameter	Id	Identification of audience
	Identity	Audience's belong identity

- Response

[Table 22] Table of insert response of request to speak

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Request to speak object	New requester registration notification
Failure Response Body	Message	Fail message

6.7.2. Delete request

- Request

[Table 23] Table of delete request of request to speak

Attribute	Detail
Method	Delete request
URL	/admin/debate

Parameter	Id	Identification of audience

- Response

[Table 24] Table of delete response of request to speak

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.7.3. Show request

- Request

[Table 25] Table of show request of request to speak

Attribute	Detail	
Method	Show request	
URL	/admin/debate	
Parameter	Query	The query for getting requesters list

- Response

[Table 26] Table of show response of request to speak

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Request to speak object	Showing requester list
Failure Response Body	Message	Fail message

6.8. Video object

6.8.1. Video object request

- Request

[Table 27] Table of video object request

Attribute	Detail
Method	POST
URL	/video/link
Parameter	-

- Response

[Table 28] Table of video object response

Attribute	Detail
Success Code	200 OK

Failure Code	HTTP error code = 400	
Success Response Body	Video Object	Showing Video Object
Failure Response Body	Message	Fail message

6.8.2. 2D object request

- Request

[Table 29] Table of 2D object request

Attribute	Detail
Method	POST
URL	/object/2D
Parameter	-

- Response

[Table 30] Table of 2D object response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	2D Object	Showing 2D Object
Failure Response Body	Message	Fail message

6.8.3. 3D object request

- Request

[Table 31] Table of 3D object request

Attribute	Detail
Method	POST
URL	/object/3D
Parameter	-

- Response

[Table 32] Table of 3D object response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	3D Object	Showing 3D Object
Failure Response Body	Message	Fail message

6.9. Board

6.9.1. Read post request

- Request

[Table 33] Table of read post request

Attribute	Detail	
Method	Read Post	
URL	/post/read	
Parameter	Board type	Type of the board to get the post from
	Post id	Index of post

- Response

[Table 34] Table of read post response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Post read view	Provides a view to read the post
Failure Response Body	Message	Fail message

6.9.2. Verification before write request

- Request

[Table 35] Table of verification before write request

Attribute	Detail
Method	Write

URL	/post/verification	
Parameter	Verification pw	Pw for verification

- Response

[Table 36] Table of verification before write response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Post write view	Provides a view to write the post
Failure Response Body	Message	Fail message

6.9.3. Write post request

- Request

[Table 37] Table of write post request

Attribute	Detail	
Method	Write Post	
URL	/post/write	
Parameter	Post title	Title of post
	Post content	Content of post

- Response

[Table 38] Table of write post response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.9.4. Update Post List

- Request

[Table 39] Table of post update request

Attribute	Detail	
Method	Update Post	
URL	/post/update	
Parameter	Board type	Type of the board to get the post from

- Response

[Table 40] Table of post update response

Attribute	Detail
Success Code	200 OK
Failure Code	HTTP error code = 400

Success Response Body	Update view	Provides updated view of post list
Failure Response Body	Message	Fail message

6.10. Voting

6.10.1. Create voting process request

- Request

[Table 41] Table of create voting process request

Attribute	Detail	
Method	saveProcess	
URL	/user/create	
Parameter	User id	id of organization
	Content	Content of voting

- Response

[Table 42] Table of create voting process response

Attribute	Detail
Success Code	200 OK

Failure Code	HTTP error code = 400	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

6.10.2. Read voting process request

- Request

[Table 43] Table of read voting process request

Attribute	Detail	
Method	openProcess	
URL	/user/open	
Parameter	Process id	Voting process that the user wants to participate in
	SSN	Student school number

- Response

[Table 44] Table of read voting process response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Voting process	Provides contents of the voting process

Failure Response Body	Message	Fail message

6.10.3. Submit the response request

- Request

[Table 45] Table of read voting process request

Attribute	Detail	
Method	Vote	
URL	/user/vote	
Parameter	contents	response to questions

- Response

[Table 46] Table of read voting process response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 400	
Success Response Body	Message	Success message
Failure Response Body	Message	Fail message

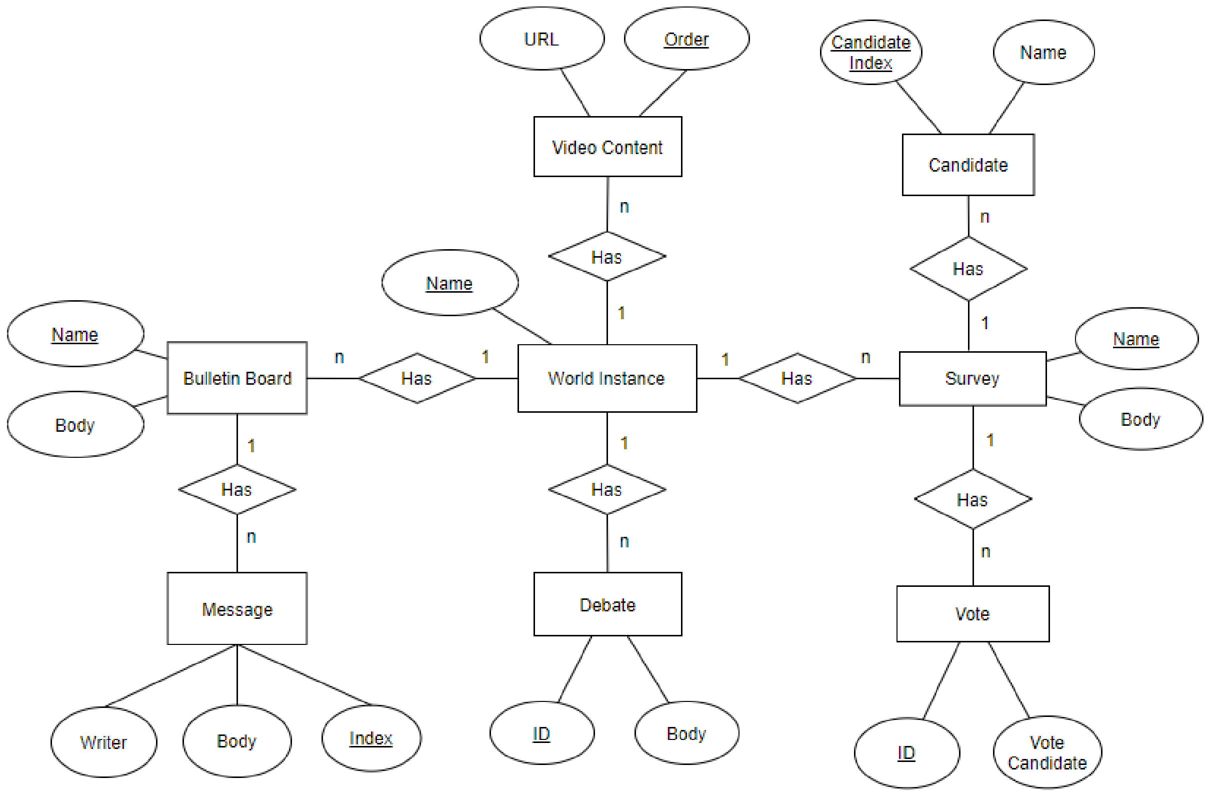
7. Database Design

7.1. Objectives

This section describes the system data structures and how these are to be represented in a database. This part describes entities first, and their relationship through ER-diagram (Entity Relationship diagram).

7.2. ER Diagram

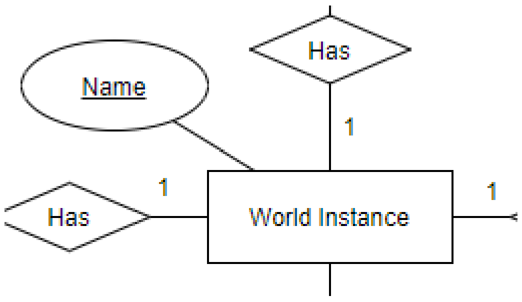
The system consists of eight entities: Bulletin Board, Message, World Instance, Debate, Video Content, Candidate, Survey, Vote. ER-diagram expresses each entity as rectangular and their relationship as a rhombus. The primary key(unique attribute) which uniquely identifies an entity is underlined. There are also entities with multiple primary keys.



[Figure 35] ER-Diagram

7.2.1 Entities

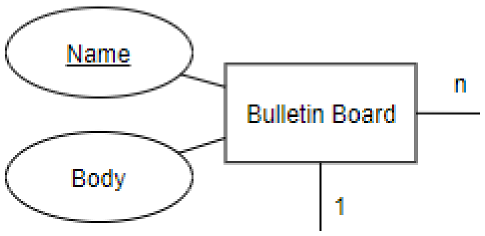
7.2.1.1. World Instance



[Figure 36] ER diagram, Entity, World Instance

Word instance entity represents the instance of the world in VRChat that can exist. It consists Name. Name is the primary key.

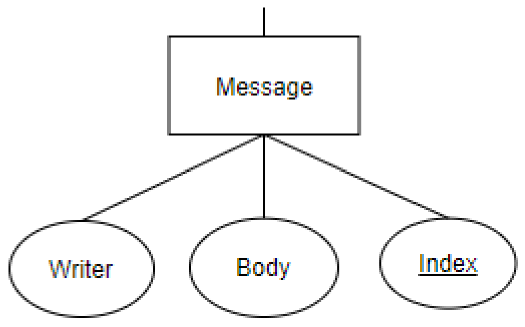
7.2.1.2. Bulletin Board



[Figure 37] ER diagram, Entity, Bulletin Board

The Bulletin Board entity represents the object that allows users to check various stored contents through a board. It consists Name and body attributes. Name is the primary key. The body contains the entire container information that can contain multiple messages.

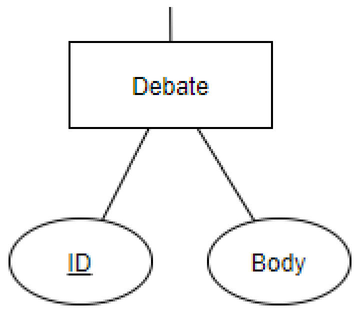
7.2.1.3. Message



[Figure 38] ER diagram, Entity, Message

The Message Board entity represents the message that reflects free opinions. It consists Writer, Body, Index attributes. Index is the primary key. Information on the person who wrote the message can be checked through the writer attribute, and the actual content of the message can be checked in the body.

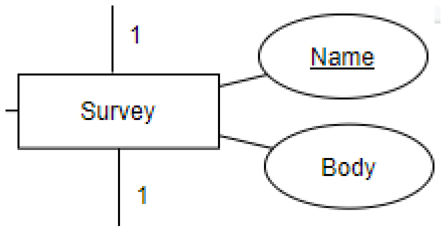
7.2.1.4. Debate



[Figure 39] ER diagram, Entity, Debate

The Debate entity represents the state that reflects available debating session. It consists ID, Body. ID is the primary key and it is used to differentiate other debating information.

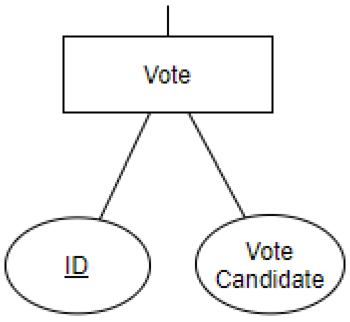
7.2.1.5. Survey



[Figure 40] ER diagram, Entity, Survey

The Survey entity represented the state that reflects available survey session. It consists Name, Body. Name is the primary key and it is used to differentiate other people who participated in the survey.

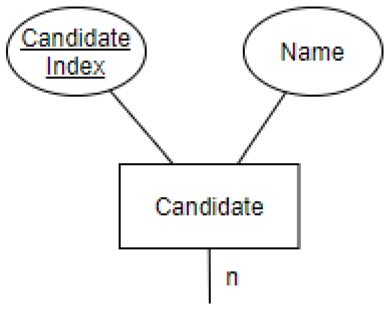
7.2.1.6. Vote



[Figure 41] ER diagram, Entity, Vote

The Vote entity represented the state that reflects available voting session. It consists ID, Vote candidate. ID is the primary key and it is used to differentiate other people who participated in the voting process.

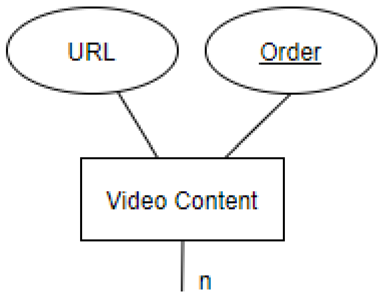
7.2.1.7. Candidate



[Figure 42] ER diagram, Entity, Candidate

The Candidate entity represented accessing to candidates registered in a particular survey. It consists Candidate Index and Name. Candidate Index is the primary key, each candidate has a unique index to distinguish through it.

7.2.1.8. Video Content



[Figure 43] ER diagram, Entity, Video Content

The Video content entity represented accessing to video content viewed by users in the world. It consists URL and Order. Order is the primary key and all screened and watched or requested contents are uniquely managed through order values.

8. Testing Plan

8.1. objectives

This phase will explain plans for development testing, release testing, and user testing. These tests play an important role in increasing program completeness and enabling stable deployment by detecting program errors and supplementing flaws. It also helps determine the effort required to verify the quality of the application being tested when testing our project. Therefore, following guides the way of thinking our project is planning.

8.2. Testing policy

8.2.1. development testing

Development testing is a test that takes place in the development phase, and the main goal is to stabilize the program. At this stage, it presents policies for the modules to test whether developed well or not satisfying the content of the design specification implemented in code. The program is stabilized through static code analysis, data flow analysis, peer code review, unit testing. In this process, we will focus on three main factors. First is to evaluate how efficiently the program achieves what it does in terms of performance. Secondly, we will evaluate how stable the program works in terms of Reliability. The third is to evaluate Security.

8.2.1.1. Performance

In particular, modules such as Vote & Survey, Debating are considered important because the entire process must proceed with many users connected at the same time. Also, our project's processes also need to consider the situation of performance decline due to rapid network traffic in VRChat. Even if we can't solve this phenomenon by looking at a completely deep level architecture, we would have to minimize the consumption of computing resources that are as unprovoked and no-reason as possible in implementing those modules. We would prepare test cases on various conditions and cases and evaluate the speed of the processing overall contexts and improve the flow of code regarding specific VRChat platform and communication with the server.

8.2.1.2. Reliability

Reliability is also one of the important factors that must be evaluated in our project. This is because a certain malfunction during the test causes a serious situation if the program is terminated or specific error prevents access to the test procedure in time. The sub-components and units comprising the system should operate and be connected correctly. In a module that conducts real-time voting and survey, a situation in which re-voting occurs due to a system problematic issue is a serious problem that should never occur, and it is a problem that users can break the reliability of this project at once. The same is true of board modules that bring together users' opinions, (never poor reliability) and in the end, reliability is an important factor that can't be unconditionally compromised. Therefore, we need to unit test first and iteratively integrate to the system checking failure iteratively and very carefully.

8.2.1.3. Security

In this project, where various participants' opinions are collected, reflected through voting, and a free forum for discussion, users' information is likely to be exposed. After checking the policies between worlds basically proposed by VRChat, additional security threats should be identified and evaluated in the process of implementing the project. If implementation proceeds as planned, it is judged that it will not deviate significantly from the scope of basic security provided by VRChat.

8.2.2. Release Testing

Release testing is the process of testing a particular release of a system that is intended for use outside of the development team. The primary goal of the release testing process is to convince the customer of the system that it is good enough for use. Testing is generally initiated from the first version when a basic implementation of the software is completed, and we will look at the responses of users. Users will compare the existing offline procedure method(offline voting, offline debating, etc.) and release the second version of them, fixing what we identify as our weaknesses and also strengthening what metaverse platform has.

8.2.3. User testing

We should set up possible scenarios and realistic situations that can proceed with necessary user tests. It plans by imitating the actual voting and discussion process by referring to the SKKU website or SNS posts of related organizations. To exclude process interference due to malicious users from the consideration first, we assume that the type of world is set to ‘friends’ or ‘invite’ to access at the invitation of an acquaintance or room manager other than designating public. After setting this situation, we would distribute the Beta version to them and collect user reviews while carrying out our use cases test.

8.2.4. Testing Case

Testing cases would be set according to 3 fundamental aspects of the Performance, Reliability, and security. Test cases that may occur as much as possible were considered to minimize coping with unexpected situations, and details are described in the test plan document.

9. Development Plan

9.1. Objectives

This chapter illustrates the development environment and tools we used during development. The limitations and assumptions considered during development will also be described.

9.2. Frontend Environment

9.2.1. Unity

[Figure 44] Unity logo

Unity is a 3D/2D game engine and a cross-platform IDE. In this project Unity will be used to create and import the necessary 3D elements that are required for this project.

9.2.2. Adobe photoshop

Adobe photoshop is a graphics editor. In this projtect Adobe photoshop will be used to create the necessary 2D assets required for designing the VRChat metaverse either as a 2D asset or a component to the 3D asset.

9.3. Backend Environment

9.3.1. Udon

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 to connect flow, inputs, and outputs. This will provide the project with the necessary behaviors and interactions within VRChat.

9.4. Constraints

The system will be designed and implemented based on the contents mentioned in this document. While other details are designed and implemented by selecting the direction preferred by the developer, the following constraints should be maintained:

- Constraints for developers:

- ✓ Use free/libre open-source software in place of proprietary software when possible
- ✓ Keep the system as dependable and secure as possible via testing
- ✓ Optimize the system while minimizing premature optimization
- ✓ Consider future scalability of the system where the system would be offered in other universities and institutions.
- ✓ Keep the "Don't Repeat Yourself" principle by using the same backend and mostly similar frontend for the system

- Constraints for users:
  - ✓ A 100Mbps or higher internet connection is required.
  - ✓ User must meet minimum system requirements to run VRChat, which includes but not limited to: Intel® i5-4590 / AMD FX 8350 equivalent or greater CPU, 4 GB RAM, NVIDIA GeForce® GTX 970 / AMD Radeon™ R9 290 equivalent or greater video card.

9.5. Assumptions and Dependencies

- The assumptions and dependencies for this system are the following:
  - ✓ Assumptions
    - ➮ It is assumed that the system is allowed to use the database of Sungkyunkwan University to authenticate the user.
    - ➮ About movement and communication through the world, Users must use the content they want.
    - ➮ About creating voting process, authentication status for each organization is checked through id, and the voting process can be created only for certified organizations.

- About user authentication and selection of the voting process, if users do not have the right to vote, it is decided to send a notification that they do not have.
- About user voting and counting voting status, it is assumed that responses to each question of those participating in the vote are received normally.
- About completion of voting process, it is not considered errors between result transmission.
- About multiple candidates debate, audiences and candidates must follow the direction of the chairperson for the debating to proceed.
- About single candidate debate, audiences and candidate must follow the direction of the chairperson for the debating to proceed.
- About writing a notice on the notice board, there are no errors in the process of writing the text.
- About writing a question and answers on the Q&A board, there are no errors in the process of writing the text
- About writing a free post on the free board, there are no errors in the process of writing the text

✓ Dependencies

- The user must have access to a PC that satisfies the minimum requirements to run VRChat
- The following are dependencies required for the project:
  - ★ Udon will be used to program the logic of the environment

10. Supporting Information

10.1. Software Requirement 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

[Table 47] Document History

Writer	Description
Gwanjong Park	✓ Dividing roles and duties, Drafting a document form ✓ Collecting documents written by 'Jungin Lee', 'Daeun Lim', 'Hojin Jeon', and advising & supplementing their works. ✓ Writing Introduction, Testing plan ✓ Meeting final document format and writing lists of contents
Seungji Lee	✓ Collecting documents written by 'Soyoung Park', 'Jiho Jang', and advising & supplementing their works. ✓ Writing Introduction, Developing plan ✓ Writing lists of figure, table, contents
Daeun Lim	✓ Chapter 4,5,6 related with 'Debating Module'
Soyoung Park	✓ Chapter 4,5,6 related with 'Vote & Survey Module'
Hojin Jeon	✓ Chapter 4,5,6 related with 'Visual Content Exhibition Module'
Jiho Jang	✓ Chapter 4,5,6 related with 'Board Module'
Jungin Lee	✓ Chapter 3.2, 5.2 related with diagram & architecture in terms of overall scope in this project ✓ Chapter 7 related with this project