The Campus Hub

Software Design Specification

2021.11.03.

Introduction to Software Engineering

2021fall_41_TEAM13

Team Leader NaHyeon-Oh
Team Member GangMin-Lee
SangMin-Han
JongWon-Heo
SoHee-Yun
Ngzhiwei
JeongAh-Lee

CONTENTS

1. Preface

1.1. Objective

In this chapter, we define the expected readership of this document and describe its structure and version history. Document history shows a rationale for the creation of a new version and a summary of the changes made in each version done by each team member.

1.2. Readership

Team 13 designs and implements the functions of the Campus hub service according to this specification. Then, team members of Team 13, professors, and TAs in the Introduction to Software Engineering class are expected be the main readers. Furthermore, students or staff related to Sungkyunkwan University who have access to this system can also become readers.

1.2.1. User

The expected reader of user requirements is users of the system. Therefore, from the user's point of view, we explain the requirements in an easy-to-understand manner, refraining from using technical terms and rather using visual materials such as diagrams with the description in natural language..

1.2.2. System

System requirements are a structured language expression of system functions and constraints. It shall be written systematically so that it can be used as a reference material during the development process for developers or used in contracts with customers. Therefore, this is mainly for developers, Team 13 members.

1.3. Document Structure

Before starting each chapter describing our system, we introduce what each chapter covers in this chapter.

1.3.1. Introduction

This chapter mainly includes the instructions of several diagrams and tools applied to design our project. Additionally, references we have been used for this design specification regarding our system are also described in this chapter.

1.3.2. System Architecture

This chapter suggests how the architectures of our system are produced in three divided description parts. You can get overall insight about the campus hub by context diagram, sequence diagram and use case diagram of our system. The system is subdivided by several classes. For each class, we introduce attributes and methods and show class diagram and sequence diagram for frontend perspective and backend perspective.

1.3.3. Protocol Design

We describe design of protocols which used for communication of entities. The contents are focused on the classes in subsystem where communication is needed. For each class, we describe request and response protocols by tabular type.

1.3.4. Database Design

In this chapter, we describe how structures are organized and the relationships are constructed for the system data. data structure of system and process

1.3.5. Testing Plan

In this chapter, the testing plan of our system is illustrated. The test will be progressed by three phases which are development, user and release test. You can check how our system is tested for performance, reliability, usability, maintainability and security. Also, we define testing policy and test case.

1.3.6. Development Plan

In this chapter, the tools and technologies in frontend environments and backend environments for our system is introduced and described how those are utilized. At the last section in this chapter, you can see the schedule to develop our system.

1.3.7. Index

In this chapter, you can check all tables and figures used to describe the product at a glance.

1.4. Document History

[Table 1] Document History

Date	Version	Description	Writer
2021/11/03	0.1	Style and overview	NaHyeon-Oh
2021/11/11	1.0	Add contents of 5.3.2 – 5.3.5	Ngzhiwei
2021/11/12	1.1	Add contents of 6.3.5 – 6.3.6	JeongAh-Lee
2021/11/12	1.2	Add contents of 6 – 6.3.4	NaHyeon-Oh
2021/11/12	1.2.1	Assemble and organize	NaHyeon-Oh
2021/11/12	1.3	Add contents of 4.2.10–4.2.18	Sangmin-Han
2021/11/12	1.3	Add contents of 4.2.19–4.2.25	Sohee-Yoon
2021/11/12	1.3	Add contents of 4.2.1–4.2.9	Jongwon-Heo
2021/11/12	1.3	Add contents of 5 – 5.3.1	Gangmin-Lee
2021/11/12	1.3	Reflect feedback of 6 – 6.1.3	NaHyeon-Oh
2021/11/15	1.4	Assemble and organize	Sangmin-Han
2021/11/16	1.5	Add contents of 9 – 9.2.4	Gangmin-Lee
2021/11/18	1.6	Add contents of 1.3.2 – 1.3.6	Sangmin-Han
2021/11/18	1.7	Add contents of 7.2.3 – 7.2.3.19	NaHyeon-Oh
2021/11/19	1.8	Add contents of 7.1 – 7.2.2	Ngzhiwei
2021/11/19	1.8	Add contents of 8.2.1.4 – 8.2.4	Jongwon-Heo
2021/11/19	1.8	Add contents of 9.3 – 9.4	JeongAh-Lee
2021/11/19	1.8	Add contents of 8.1 – 8.2.3	Sohee-Yoon
2021/11/19	1.8.1	Assemble and organize	Sangmin-Han

2. Introduction

2.1. Objective

This document is a Software Design Specification (SDS) for providing The Campus Hub. This system is designed and implemented by Team 13 of the Introduction to Software Engineering at Sungkyunkwan University. The requirements for this are summarized, analyzed, and the system is designed and implemented based on the contents described in this document.

2.2. Applied Diagrams

2.2.1. Unified Modeling Language (UML)

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques. UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems. UML can be applied to diverse application domains (e.g., banking, finance, internet, aerospace, healthcare, etc.) It can be used with all major object and component software development methods and for various implementation platforms (e.g., J2EE, .NET). It is based on diagrammatic representations of software components. As the old proverb says: "a picture is worth a thousand words". By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

2.2.2. Use case Diagram

A cornerstone part of the system is the functional requirements that the system fulfills. Use Case diagrams are used to analyze the system's high-level requirements. These requirements are expressed through different use cases. We notice three main components of this UML diagram: Functional requirements – represented as use cases; a verb describing an action. Actors – they interact with the system; an actor can be a human being, an organization or an internal or external application. Relationships between actors and use cases –

represented using straight arrows.

2.2.3. Sequence Diagram

Sequence diagrams are probably the most important UML diagrams among not only the computer science community but also as design-level models for business application development. Lately, they have become popular in depicting business processes, because of their visually self-explanatory nature. As the name suggests, sequence diagrams describe the sequence of messages and interactions that happen between actors and objects. Actors or objects can be active only when needed or when another object wants to communicate with them. All communication is represented in a chronological manner.

2.2.4. Class Diagram

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. Since classes are the building block of objects, class diagrams are the building blocks of UML. The various components in a class diagram can represent the classes that will actually be programmed, the main objects, or the interactions between classes and objects. The class shape itself consists of a rectangle with three rows. The top row contains the name of the class, the middle row contains the attributes of the class, and the bottom section expresses the methods or operations that the class may use. Classes and subclasses are grouped together to show the static relationship between each object.

2.2.5. Context Diagram

The system context diagram (also known as a level 0 DFD) is the highest level in a data flow diagram and contains only one process, representing the entire system, which establishes the context and boundaries of the system to be modeled. It identifies the flows of information between the system and external entities (i.e. actors). A context diagram is typically included in a requirements document. It must be read by all project stakeholders and thus should be written in plain language, so the stakeholders can understand items. The objective of the system 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. A system context diagram is often used early in a project to determine the scope

under investigation. Thus, within the document. A system context diagram represents all external entities that may interact with a system. The entire software system is shown as a single process. Such a diagram pictures the system at the center, with no details of its interior structure, surrounded by all its External entities, interacting systems, and environments.

2.2.6. Entity Relationship Diagram (ERD)

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases. Entity Relationship diagrams are used to sketch out the design of a database.

2.2.7. Package Diagram

Package diagram is a sort of structure diagram showing the arrangement and organization of model elements. Package diagram can show both structure and dependencies between sub-systems or modules showing different views of a system. Package diagram can be used to simplify complex class diagrams, it can group classes into packages. A package is a collection of logically related UML element.

2.2.8. Deployment Diagram

Deployment Diagram is a sort of structure diagram used in modeling the physical aspects of an object-oriented system. Deployment diagram shows the configuration of tun time processing nodes and the components that inhibit on them. They are usually used to model the static deployment of a system. They are crucial for visualizing, specifying, and documenting embedded, client/server, and distributed systems and also for managing executable systems through forward and reverse engineering. The diagram is a collection of vertices and arcs.

2.3. Applied Tools

2.3.1. Microsoft PowerPoint

This is a tool that supports drawing text and figures. It is convenient to draw diagrams using various shapes. In addition, it is easy to edit because it works as full word-processor formatting (which is a tool for working with documents), graphic shapes with attached text for drawing diagrams and tables.

2.3.2. ERD Plus

This is a web-based database modeling tool that lets you quickly and easily create Entity Relationship Diagrams (ERDs), Relational Schemas (Relational Diagrams), and Star Schemas (Dimensional Models).

2.3.3. Draw.io

This is a free online diagramming software application for building diagramming applications. It also produces web-based diagramming technology and integrates with Google Drive and Dropbox. They produce draw.io, a web-based diagramming application built on mxGraph. It enables you to create flowcharts, UML, entity relation, network diagrams, mockups and more. In this paper, every diagram except ERDs is written by draw.io.

2.4. Project Scope

The purpose of this document is to outline and publish the requirement specification for our smart campus in Metaverse, the Campus Hub. Unlike many other existing virtual smart campuses out in the current market, the Campus Hub is at a whole new level when it comes to creating a unique and seemingly-real campus life for our students and professors. Especially during this period where schools, students, and teaching and non-teaching staff struggle to the changes in the post-pandemic approaches, the Campus Hub compensates and provides an alternative solution to the issues. Academically, the Campus Hub has virtual classrooms which will not only improve learning but more importantly, regain the level of comfort and convenience for teaching, learning, and interactions between students and professors. Apart from the academic side, non-academically, the Campus Hub has an exhibition hall, library, and hub in which each offers a new spectrum of learning, expanding non-academically horizons, and having fun.

The remainder of this software requirement specification document includes three chapters and indexes. The 3rd chapter provides a bird's eye view of the Campus Hub, including its system interfaces, user interfaces, hardware interfaces, and more. Also, it includes brief descriptions of the Campus Hub's functionalities. Furthermore, the chapter ends with some constraints which need to be adhered to and some assumptions and dependencies which need its equal considerations. In the 4th chapter, this document dives deeper into looking at the specific requirements, hardware and software interfaces and communication interfaces. Also, to provide a better understanding on the functional requirements, use case diagram, entity relationship diagram, and data flow diagram were put together to aid in the understanding of the Campus Hub from a diagrammatic perspective. Nonfunctional requirements, not described in the 3rd chapter, are described in this chapter. Last but not least, the 5th and final chapter talks about the system evolution. It provides a brief yet much-considered overview on the limitations and assumptions the Campus Hub may potentially face, and also, provides information on the evolution of hardware and change of user requirements.

All members contributed equally to the production of this document and we sincerely hope that you, the reader, can garner a better understanding of The Campus Hub through this document.

2.5. References

- Team 1. "Software Design Document". SKKU, Last Modified: May. 24, 2020. https://github.com/skkuse/2020spring_41class_team1/blob/master/docs/SDD_TEAM1.pdf
- Appleton, Brad. A Software Design Specification Template. N.d.
- P. Burke, K. Martin, D. Longtin. Software Design Specification for a One Runway Airport/Air Traffic Controller Simulation <https://www.academia.edu/29811493/SOFTWARE_DESIGN_SPECIFICATION_FOR_A_ONE_RUNWAY_AIRPORT_AIR_TRAFFIC_CONTROLLER_SIMULATION>.

- https://www.crunchbase.com/organization/draw-io

3. System Architecture - Overall

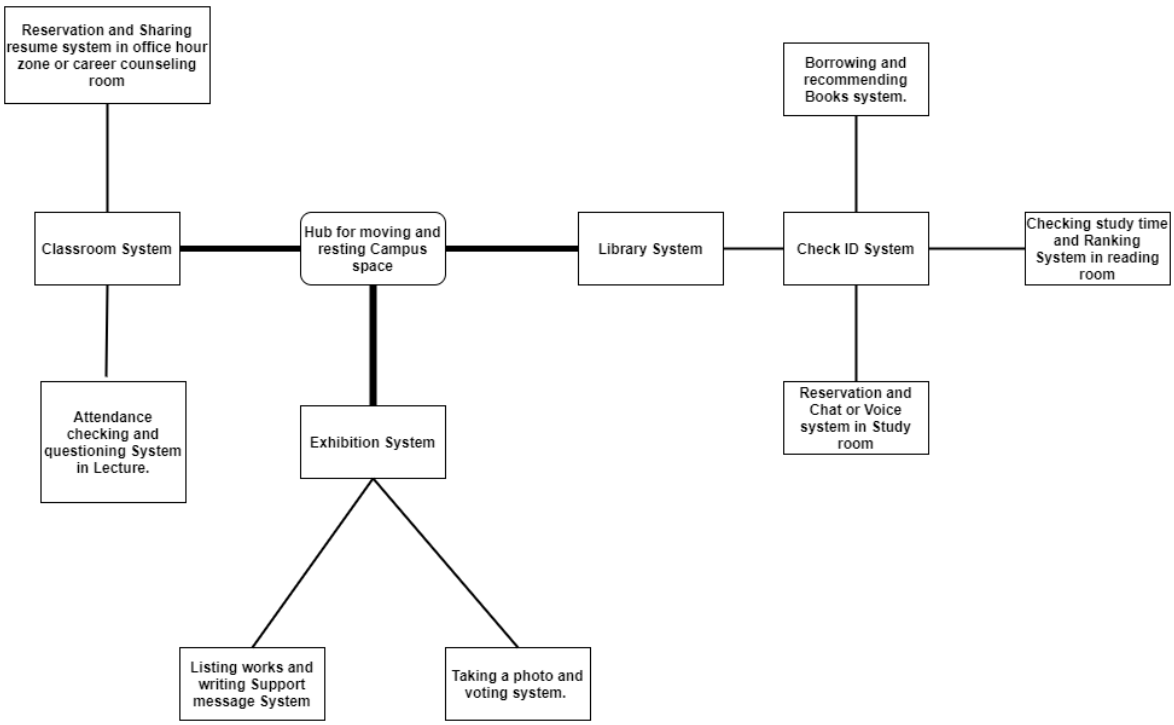
3.1. Objectives

In this chapter, we describe and show the organization of the system, ranging from the frontend design to the backend design of the application for the project.

3.2. System Organization

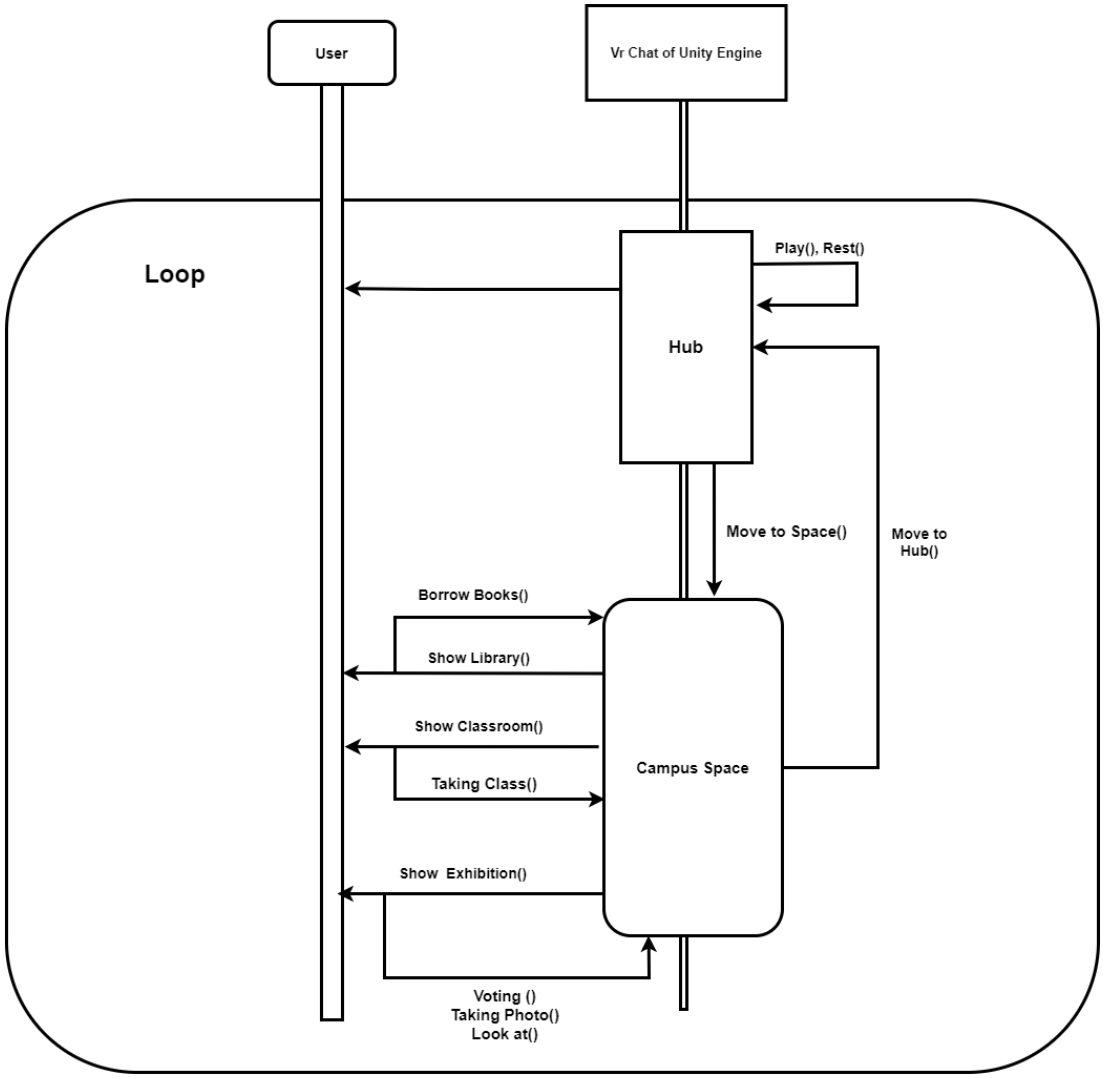
3.2.1. Context Diagram

[Diagram 1] Context model



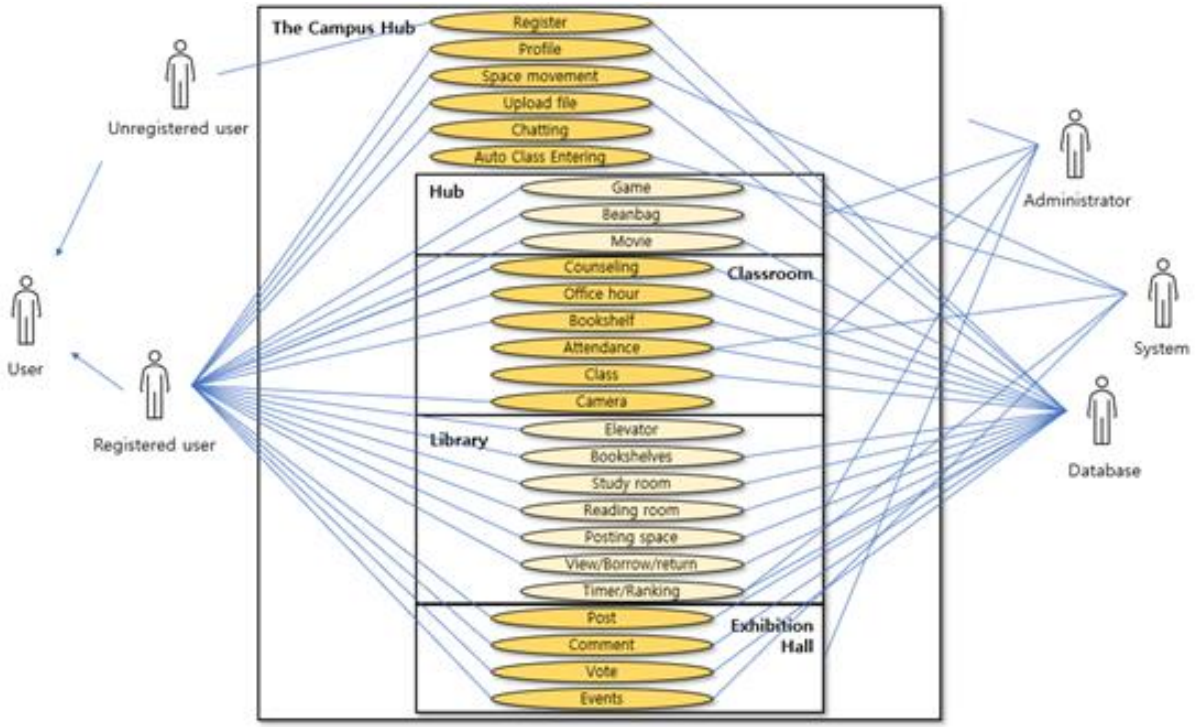
3.2.2. Sequence Diagram

[Diagram 2] Sequence diagram



3.2.3. Use Case Diagram

[Diagram 3] Use case diagram


The Campus Hub
Register
Profile
Space movement
Upload file
Chatting
Auto Class Entering
Hub
Game
Beanbag
Movie
Classroom
Counseling
Office hour
Bookshelf
Attendance
Class
Camera
Library
Elevator
Bookshelves
Study room
Reading room
Posting space
View/Borrow/return
Timer/Ranking
Exhibition Hall
Post
Comment
Vote
Events
Unregistered user
User
Registered user
Administrator
System
Database

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. Entering the Campus Hub

4.2.1.1. Attributes

These are the attributes that entering the campus hub object has.

- **User_id** : id of the user (email address)
- **Student_id** : student id of the user
- **Password**

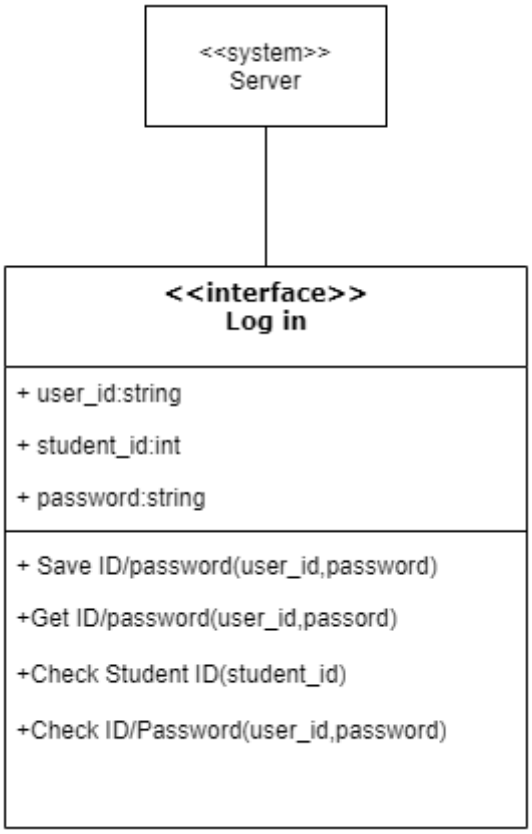
4.2.1.2. Methods

These are the methods that entering the campus hub class has.

- Save ID/Password()
- Get ID/Password()
- Check Student ID()
- Check ID/Password()

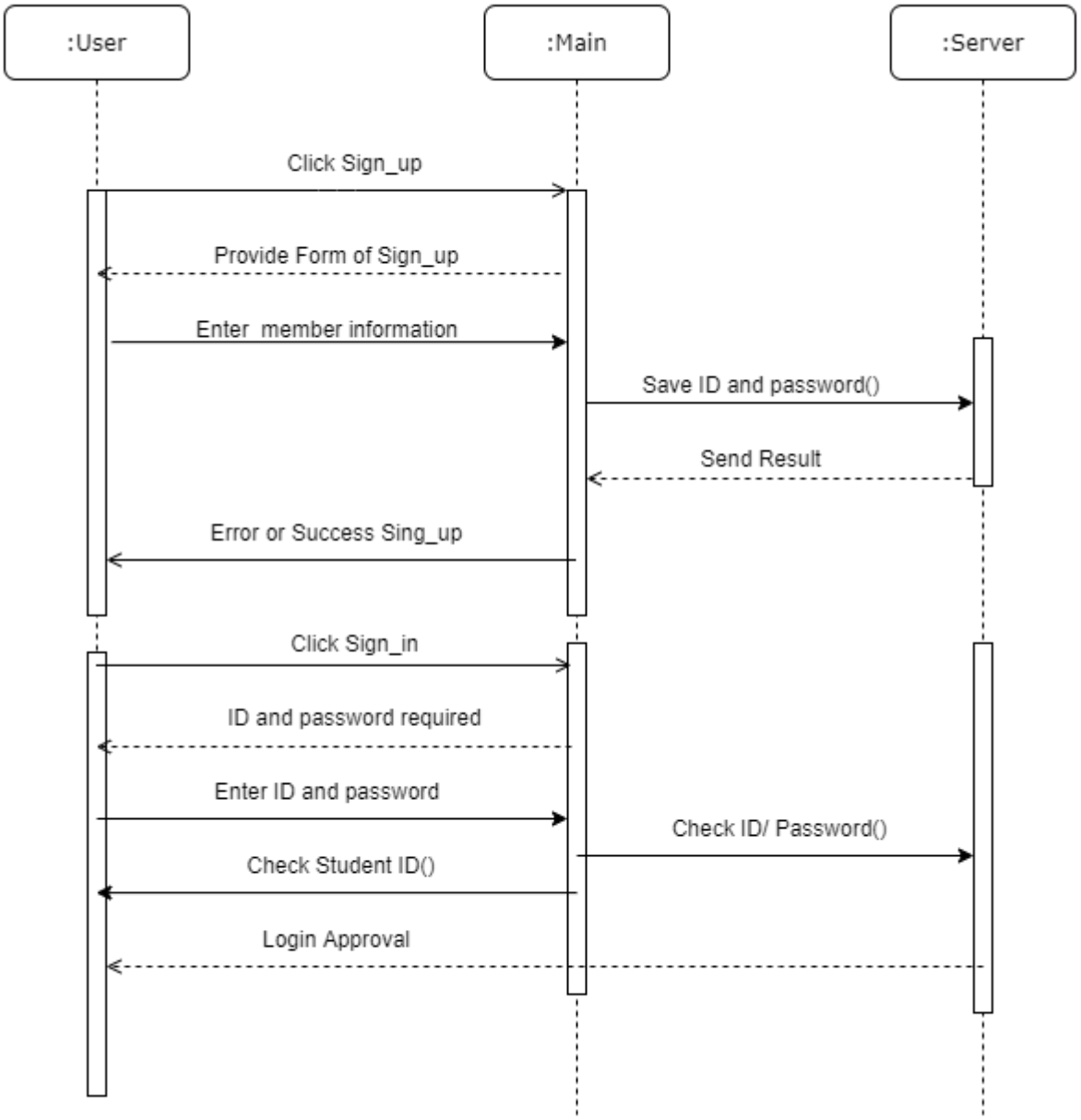
4.2.1.3. Class Diagram

[Diagram 4] Class diagram – Entering the campus hub



4.2.1.4. Sequence Diagram

[Diagram 5] Sequence diagram – Entering the campus hub



4.2.2. Profile

4.2.2.1. Attributes

These are the attributes that profile object has.

- **User_id** : id of the user (student ID)
- **Profile**
- **Avatar**

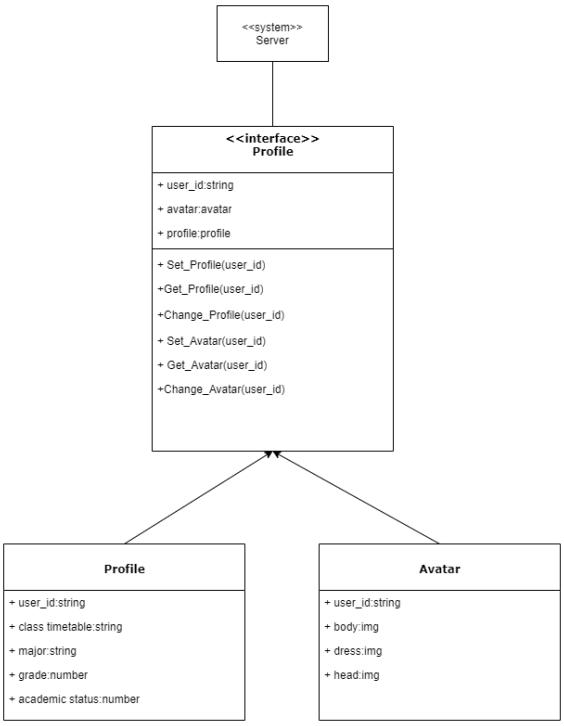
4.2.2.2. Methods

These are the methods that profile class has.

- Set_profile()
- Get_profile()
- Change Profile()
- Set_Avatar()
- Get_Avatar()
- Change_Avatar()

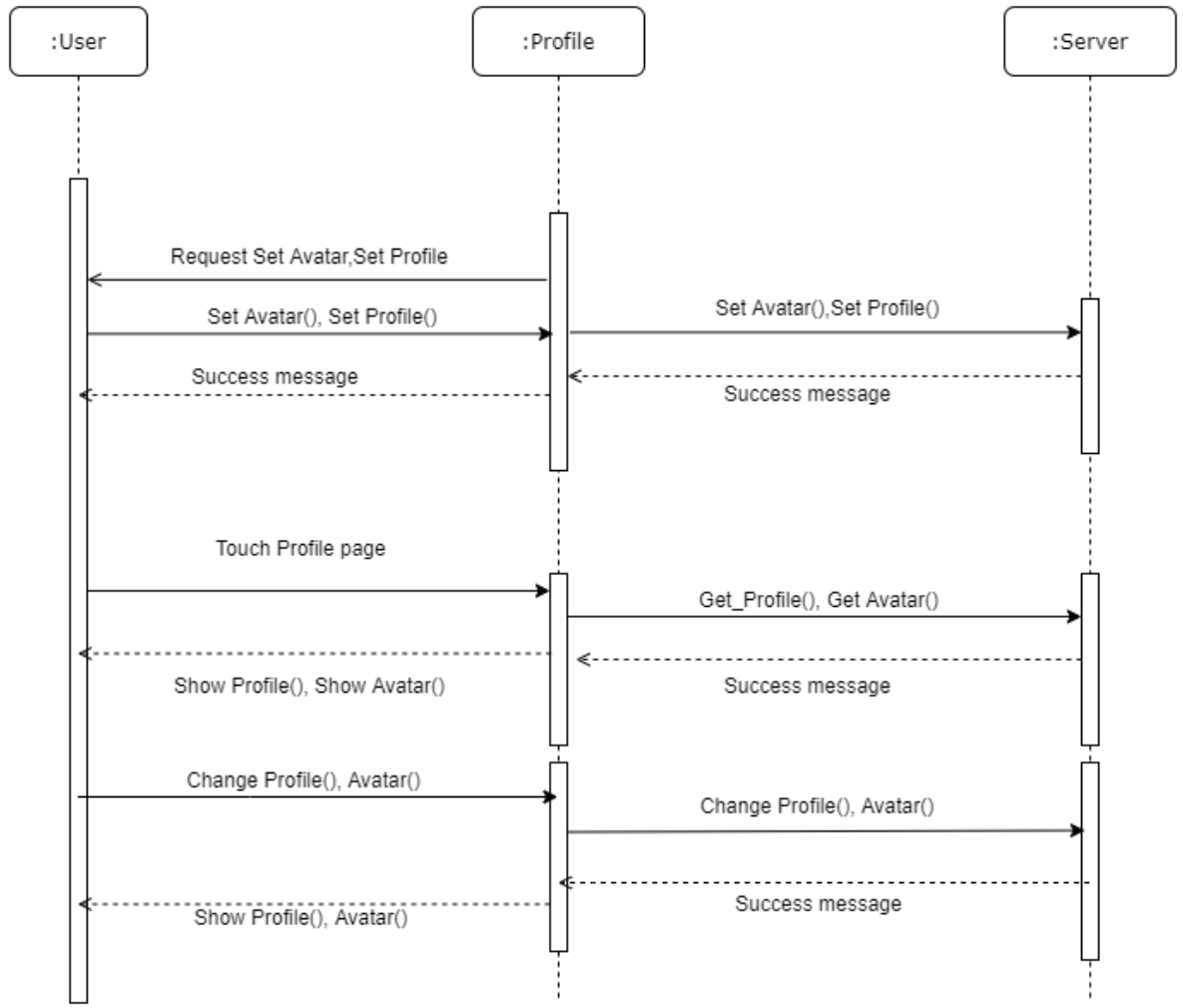
4.2.2.3. Class Diagram

[Diagram 6] Class diagram – Profile



4.2.2.4. Sequence Diagram

[Diagram 7] Sequence diagram – Profile



4.2.3. Space movement

4.2.3.1. Attributes

These are the attributes that space movement object has.

- **User_id**: id of the user (student ID)
- **Space**: one of the spaces (hub, classroom, library, exhibition)

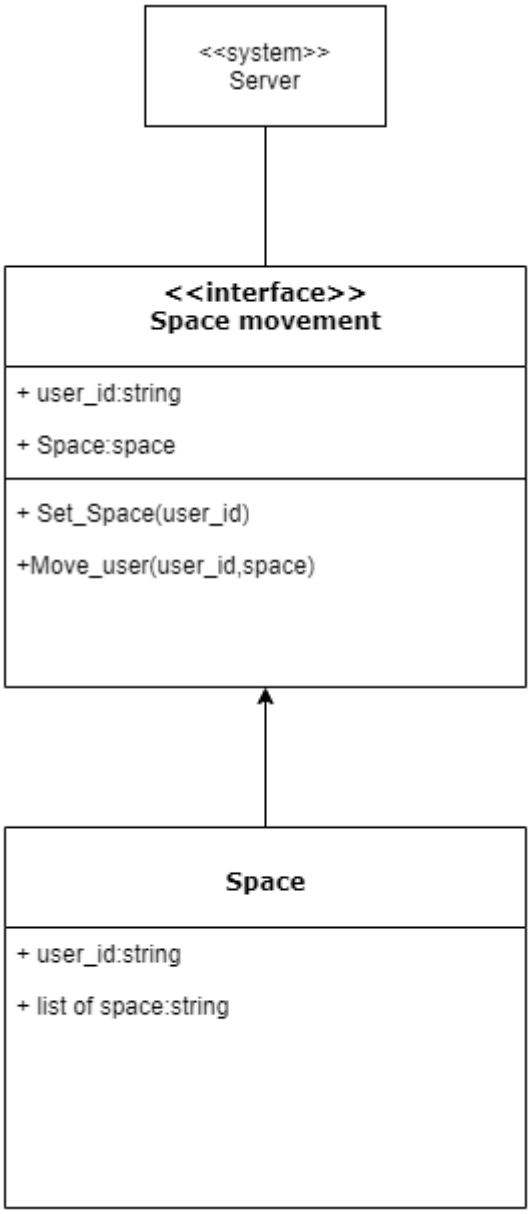
4.2.3.2. Methods

These are the methods that space movement class has.

- Set_Space()
- Move_user()

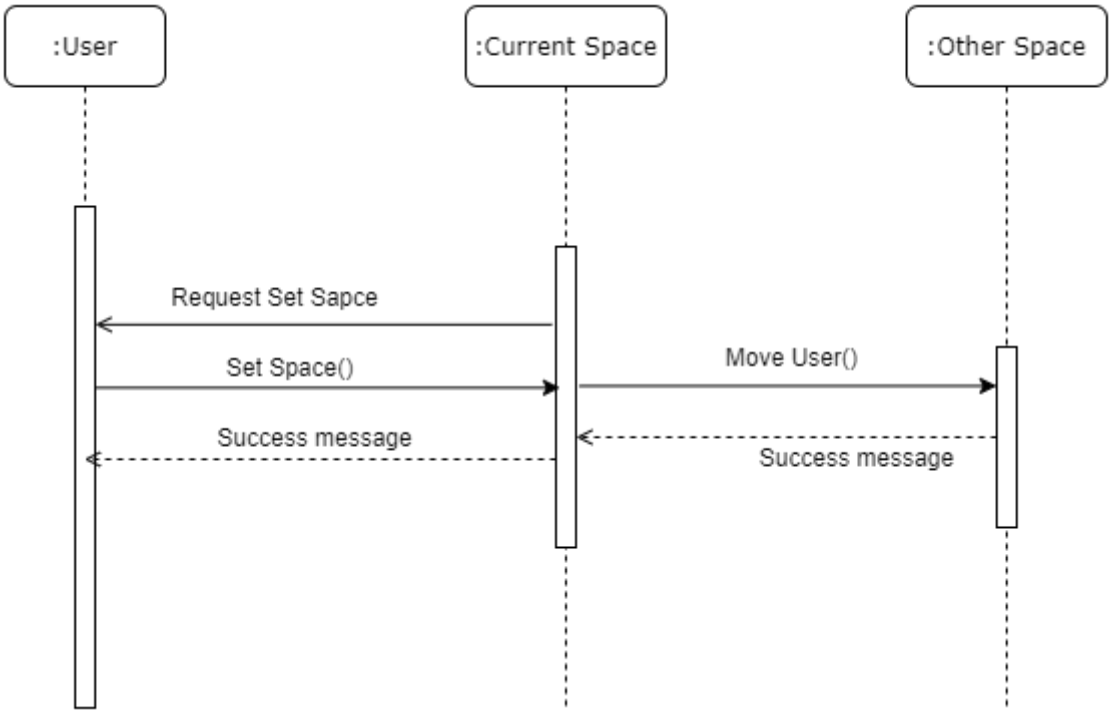
4.2.3.3. Class Diagram

[Diagram 8] Class diagram – Space movement



4.2.3.4. Sequence Diagram

[Diagram 9] Sequence diagram – Space movement



4.2.4. Upload file

4.2.4.1. Attributes

These are the attributes that upload file object has.

- **User_id**: id of the user (student ID)
- **URL**: url where the file saved at

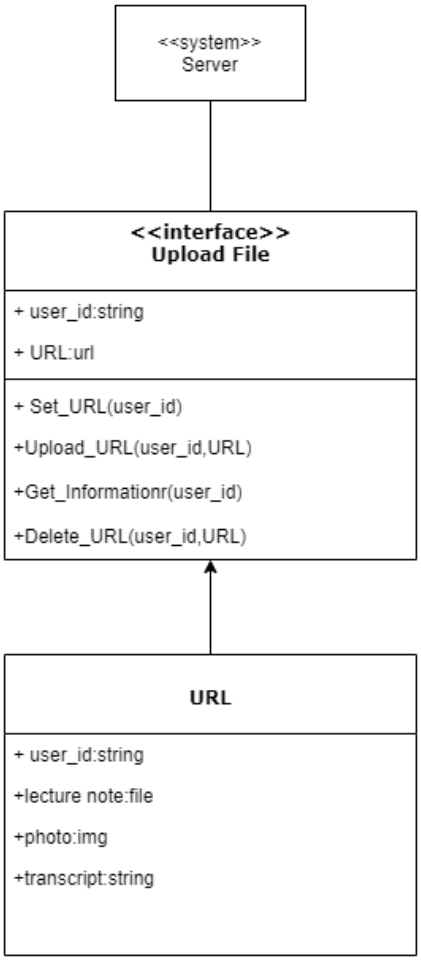
4.2.4.2. Methods

These are the methods that upload file class has.

- Set_URL()
- Upload_URL()
- Get_Information()
- Delete_URL()

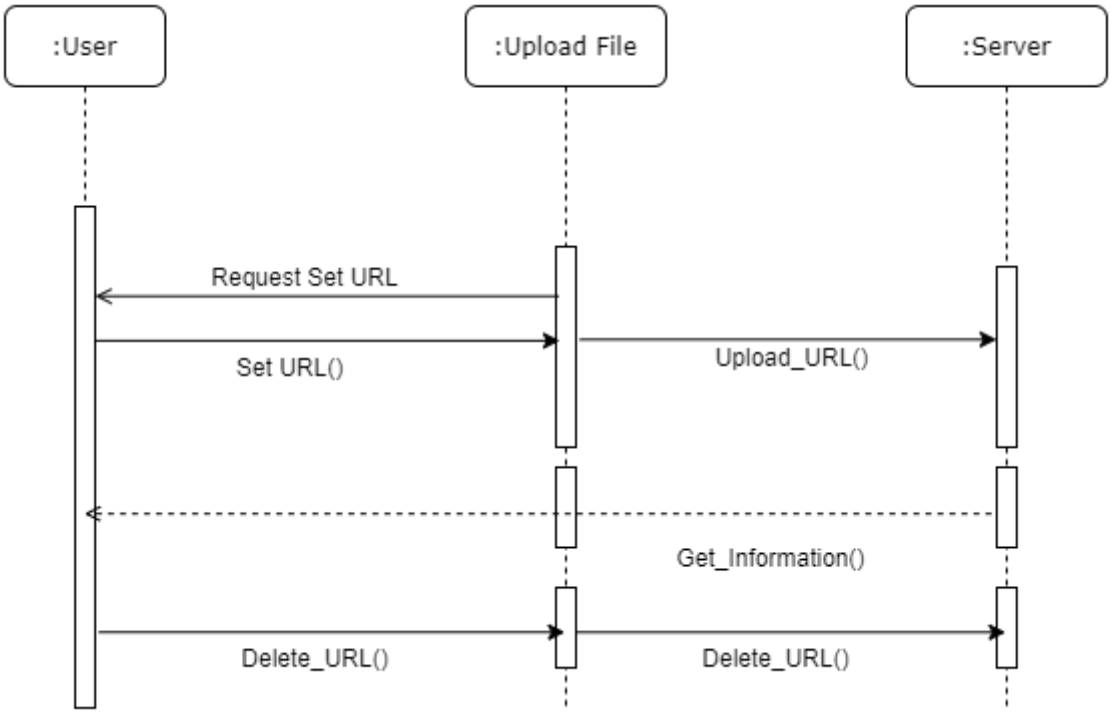
4.2.4.3. Class Diagram

[Diagram 10] Class diagram – Upload file



4.2.4.4. Sequence Diagram

[Diagram 11] Sequence diagram – Upload file



4.2.5. Chat

4.2.5.1. Attributes

These are the attributes that chat object has.

- **User_id**: id of the user (student ID)
- **Message**: means of communication as text or voice

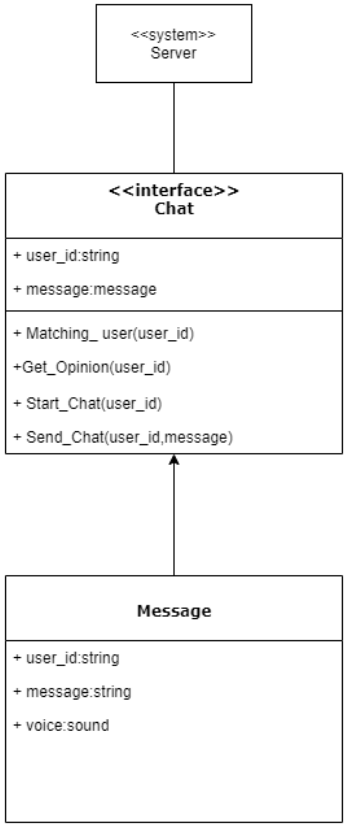
4.2.5.2. Methods

These are the methods that chat class has.

- Matching_user()
- Get_Opinion()
- Start_Chat()
- Send_Chat()

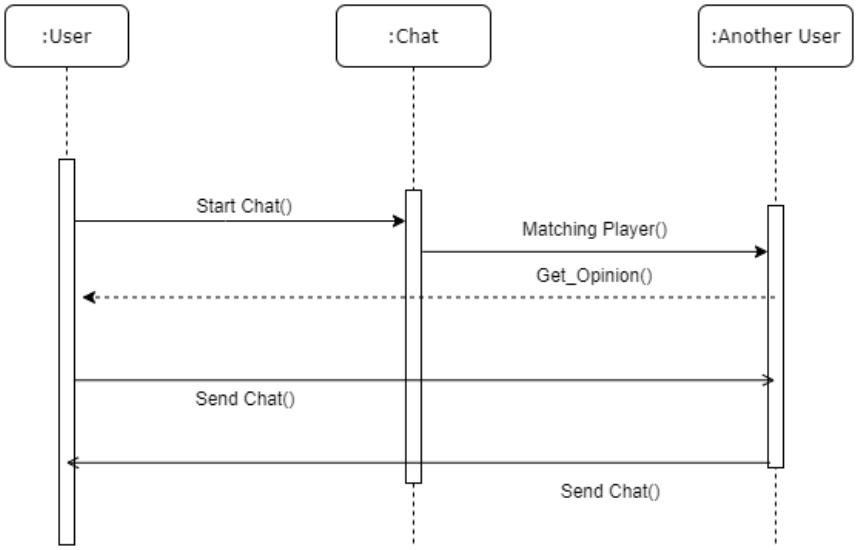
4.2.5.3. Class Diagram

[Diagram 12] Class diagram – Chat



4.2.5.4. Sequence Diagram

[Diagram 13] Sequence diagram – Chat



4.2.6. Hub

4.2.6.1. Attributes

These are the attributes that hub object has.

- **User id**: id of the user (student ID)
- **Space_key**

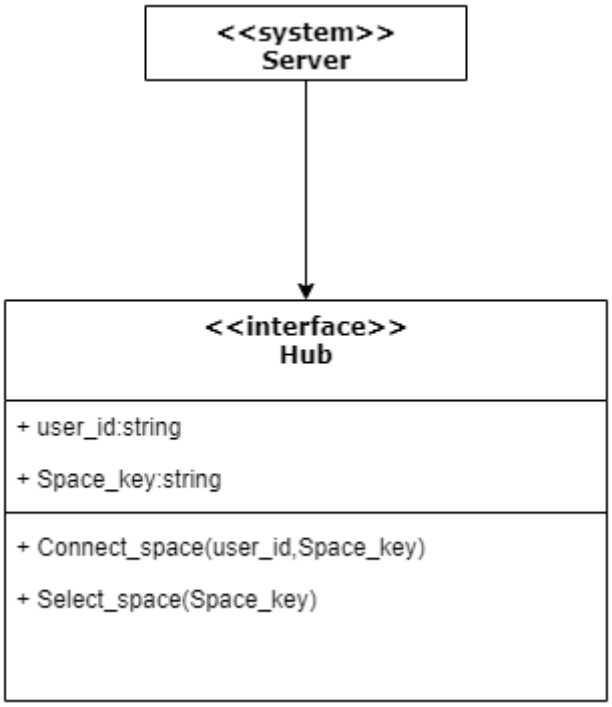
4.2.6.2. Methods

These are the methods that hub class has.

- Connect_space()
- Select_space()

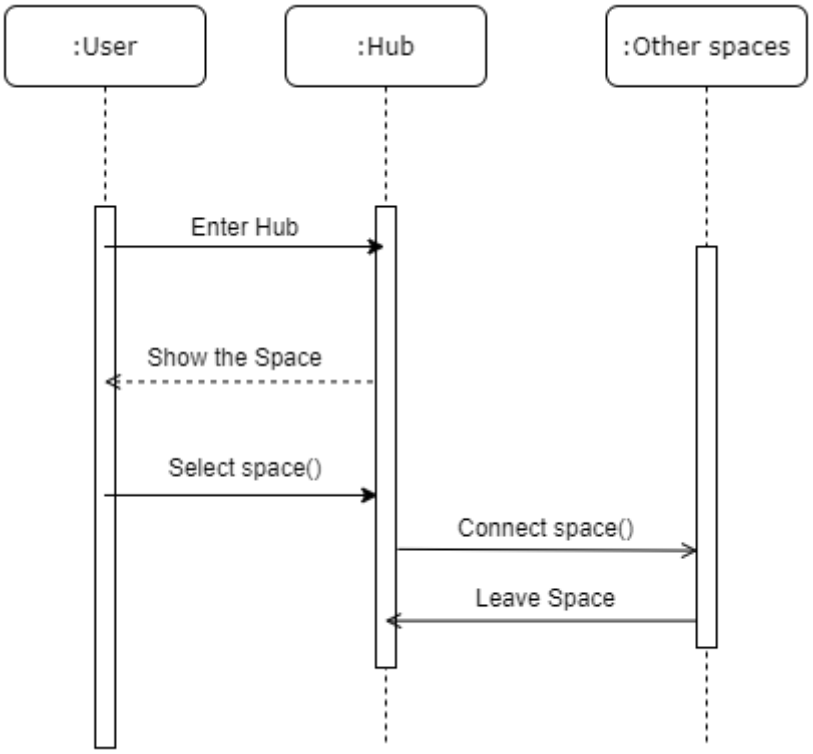
4.2.6.3. Class Diagram

[Diagram 14] Class diagram – Hub



4.2.6.4. Sequence Diagram

[Diagram 15] Sequence diagram – Hub



4.2.7. Game

4.2.7.1. Attributes

These are the attributes that game object has.

- **User id**: for matching user
- **Game key code**: for checking what game user select and show game for user

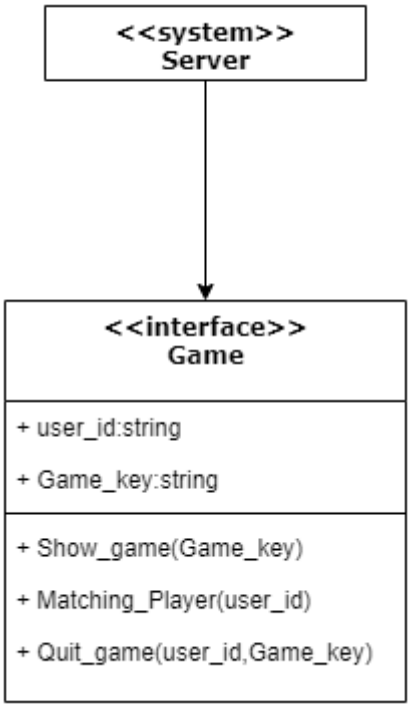
4.2.7.2. Methods

These are the methods that game class has.

- Show game()
- Matching Player()
- Quit Game()

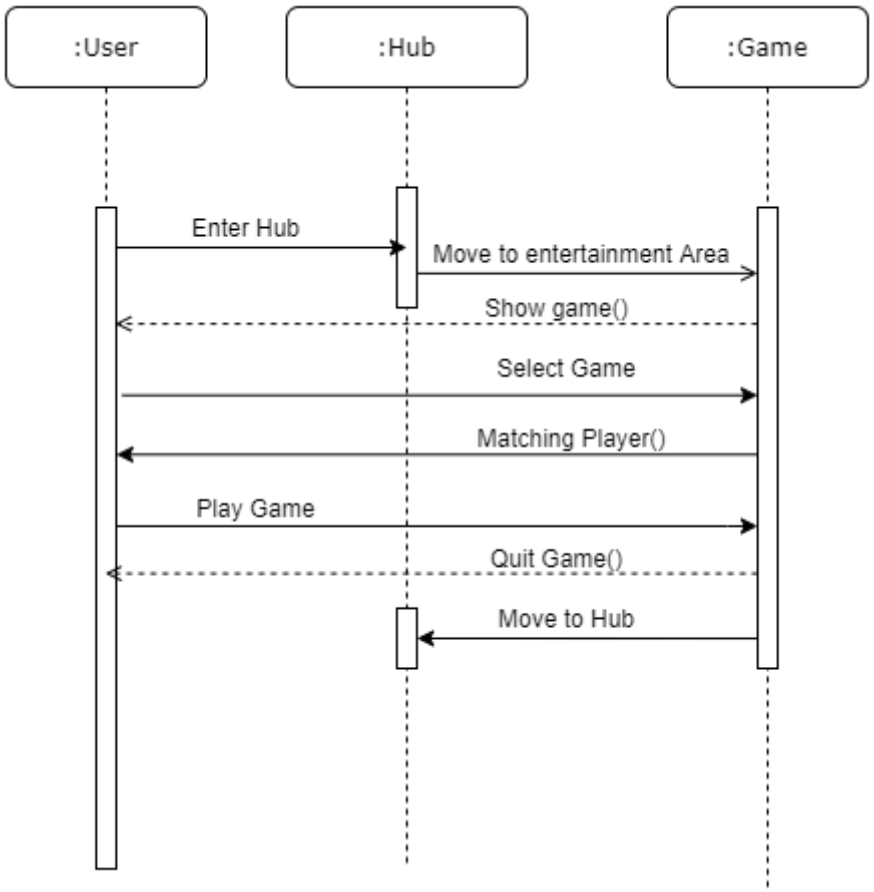
4.2.7.3. Class Diagram

[Diagram 16] Class diagram – Game



4.2.7.4. Sequence Diagram

[Diagram 17] Sequence diagram – Game



4.2.8. Movie

4.2.8.1. Attributes

These are the attributes that movie object has.

- **User Id**: for checking user watching movie
- **Movie URL**: it is need to for user

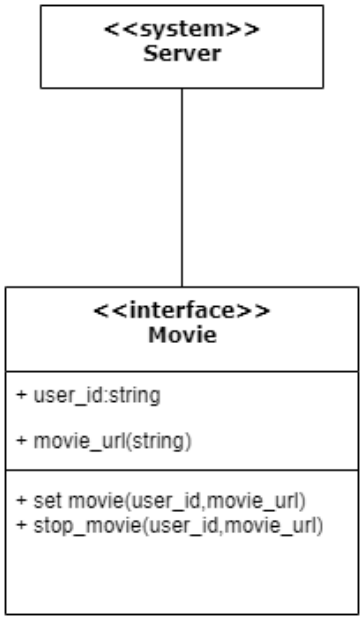
4.2.8.2. Methods

These are the methods that movie class has.

- Set_movie()
- Stop Movie()

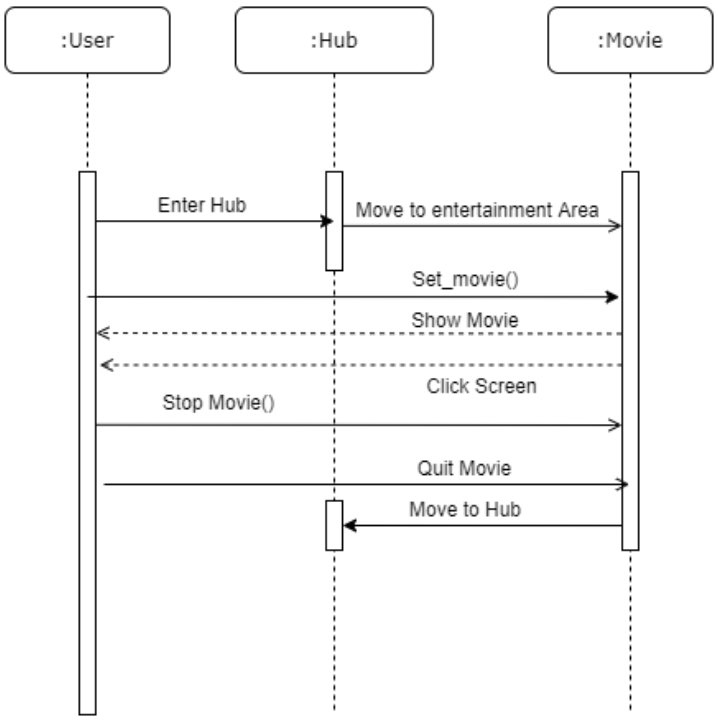
4.2.8.3. Class Diagram

[Diagram 18] Class diagram – Movie



4.2.8.4. Sequence Diagram

[Diagram 19] Sequence diagram – Movie



4.2.9. Beanbag

4.2.9.1. Attributes

These are the attributes that profile object has.

- **User id**: for checking user

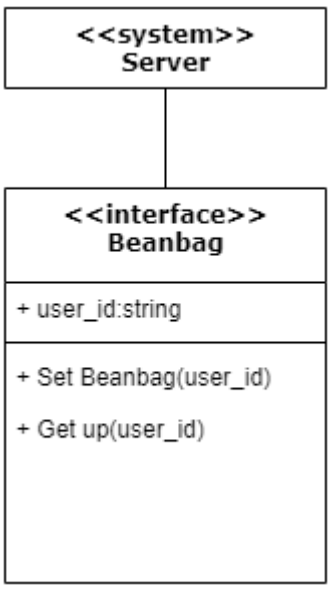
4.2.9.2. Methods

These are the methods that profile class has.

- Set Beanbag()
- Get up()

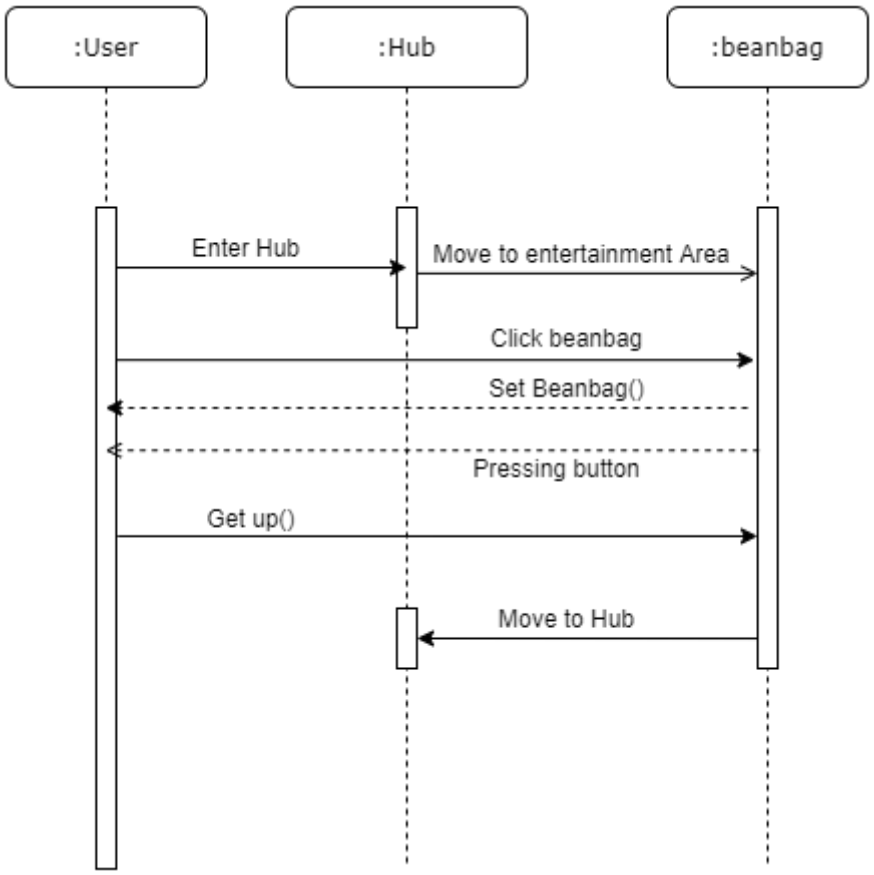
4.2.9.3. Class Diagram

[Diagram 20] Class diagram – Beanbag



4.2.9.4. Sequence Diagram

[Diagram 21] Sequence diagram – Beanbag



4.2.10. Classroom

The classroom class deals with a space "Classroom" in general where a user can give a lecture with uploading file on screen board or take a lecture on his/her seat or using bookshelf, move to Counseling and Office Hour.

4.2.10.1.Attributes

These are the attributes that classroom object has.

- **User**: id of the user
- **Seat**: whether take a seat on the chair
- **Bookshelf**: bookshelf
- **CounselingDoor**: mean of entering and reserving counseling
- **OfficehourDoor**: mean of entering and reserving officehour
- **Desk**: desk

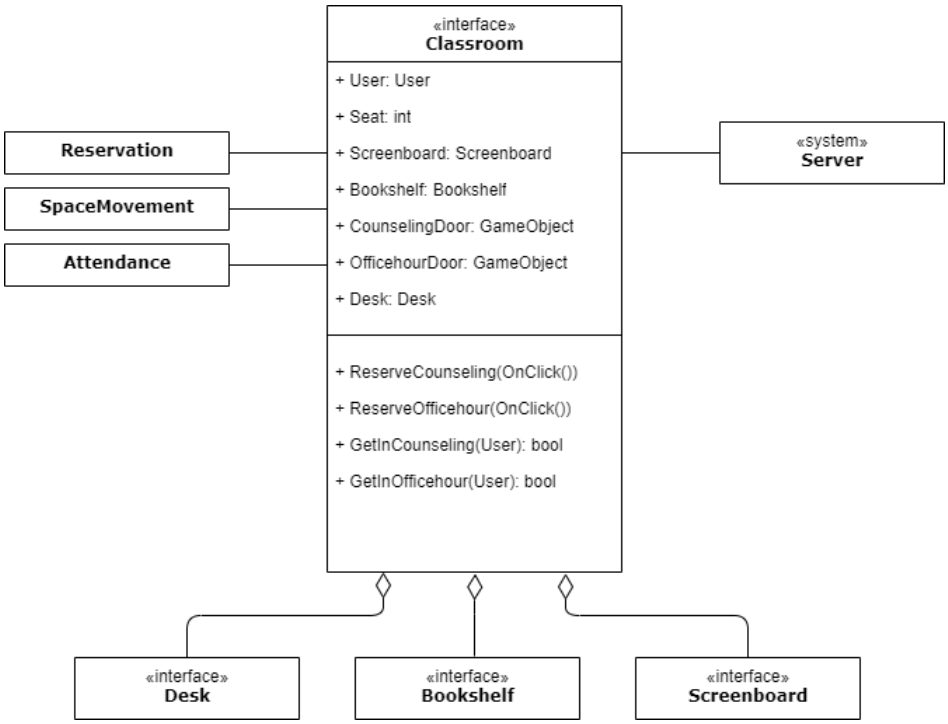
4.2.10.2.Methods

These are the methods that classroom class has.

- ReserveCounseling()
- ReserveOfficehour()
- GetInCounseling()
- GetInOfficehour()

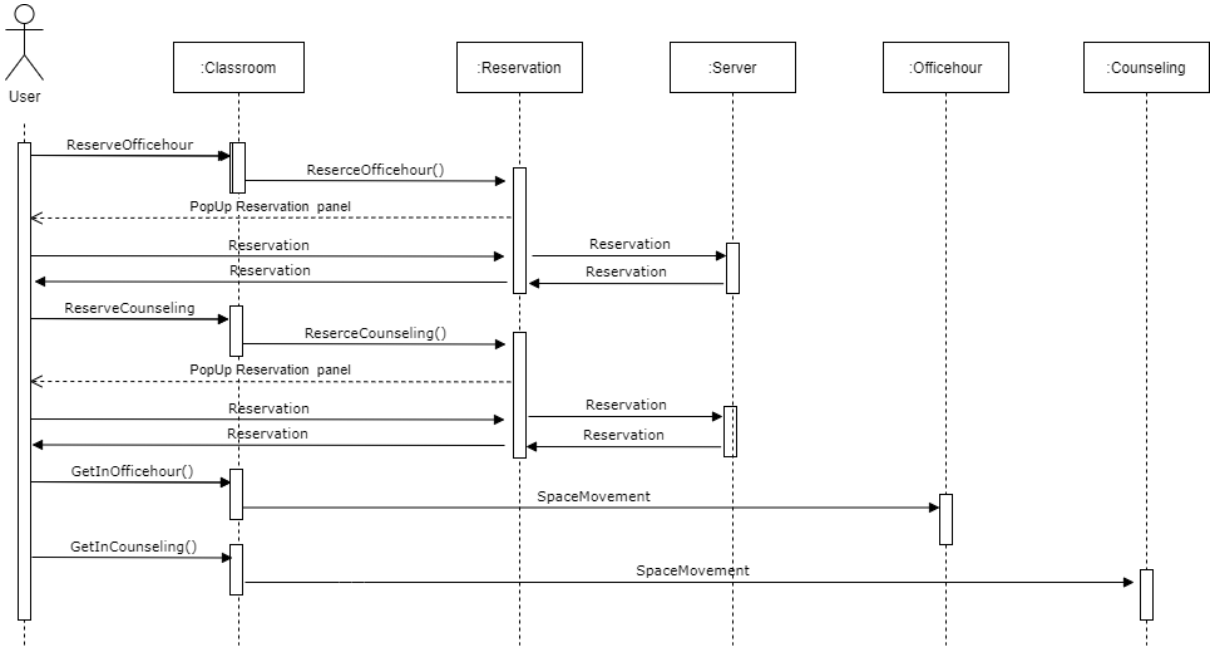
4.2.10.3.Class Diagram

[Diagram 22] Class diagram – Classroom



4.2.10.4.Sequence Diagram

[Diagram 23] Sequence diagram – Classroom



4.2.11. Attendance

The attendance deals with attendance of the class of users. The information is automatically checked by real time.

4.2.11.1.Attributes

These are the attributes that attendance object has.

- **User id**: id of the user (student ID)
- **Attendance**: whether student participate the class
- **Time**: time when the class start

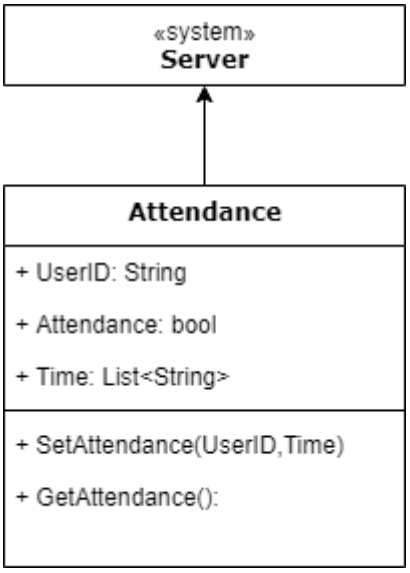
4.2.11.2.Methods

These are the methods that attendance class has.

- SetAttendance()
- GetAttendance()

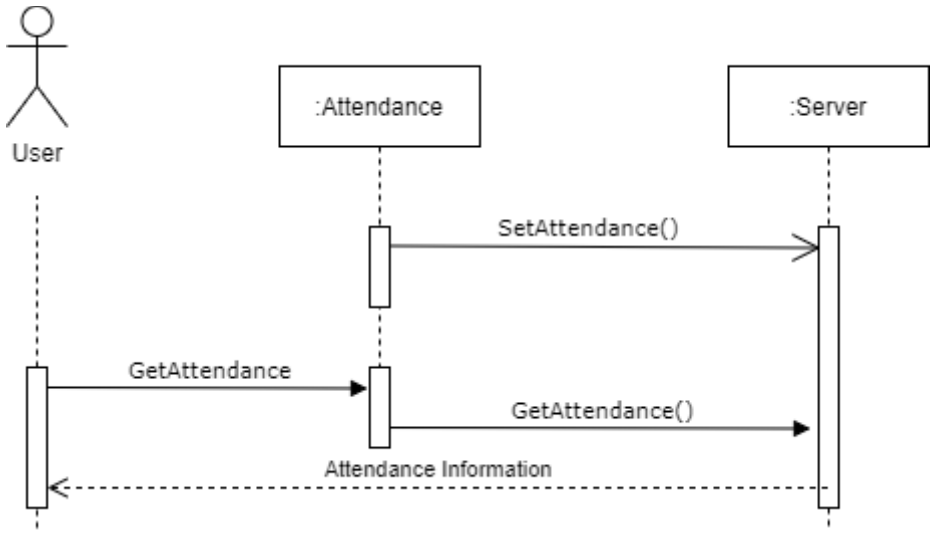
4.2.11.3.Class Diagram

[Diagram 24] Class diagram – Attendance



4.2.11.4.Sequence Diagram

[Diagram 25] Sequence diagram – Attendance



4.2.12. Screenboard

4.2.12.1.Attributes

These are the attributes that screenboard object has.

- **Screenboard**: a screen board panel to show uploaded file
- **Uploadframe**: a window user can insert url of file

- **FileURL**: a url of the file

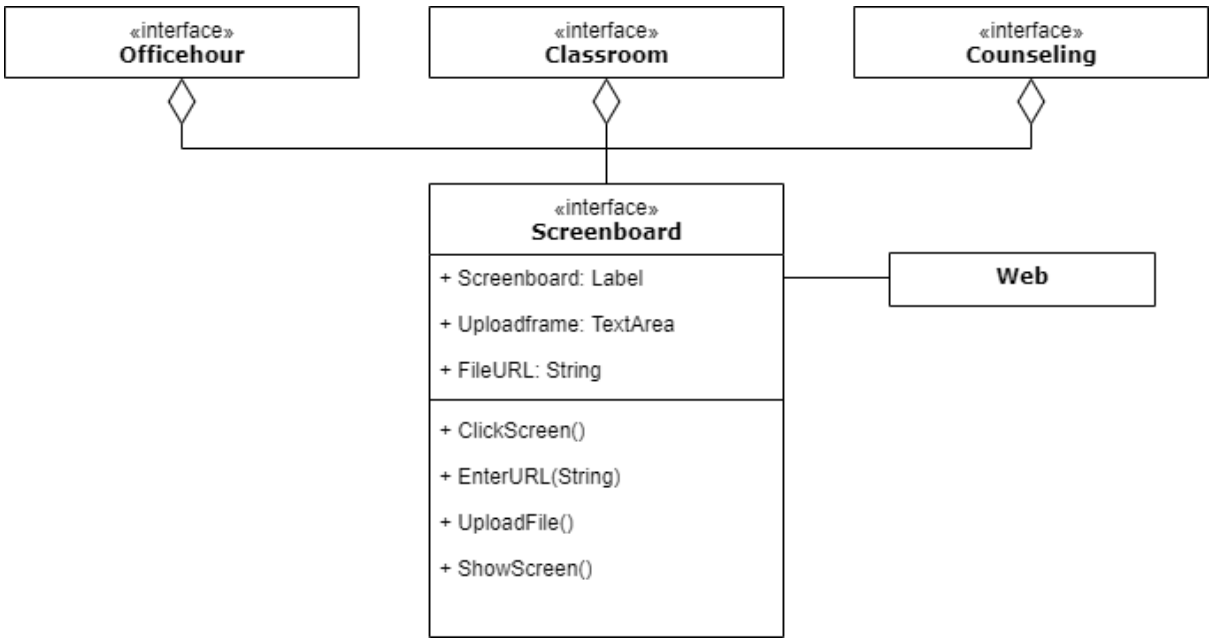
4.2.12.2.Methods

These are the methods that screenboard class has.

- ClickScreen()
- EnterURL()
- UploadFile()
- ShowScreen()

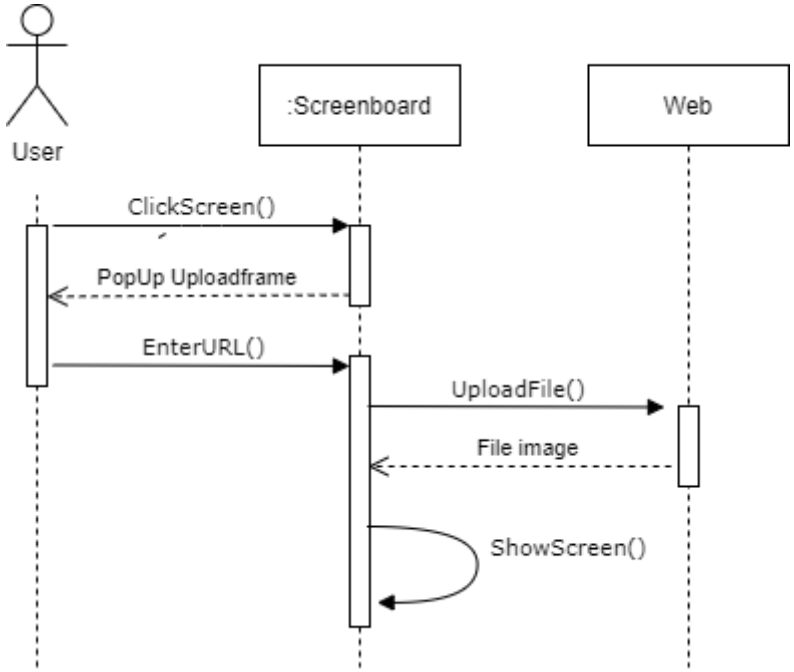
4.2.12.3.Class Diagram

[Diagram 26] Class diagram – Screenboard



4.2.12.4.Sequence Diagram

[Diagram 27] Sequence diagram – Screenboard



4.2.13. Desk

4.2.13.1.Attributes

These are the attributes that desk object has.

- **Desk**: a desk combined with a chair
- **Chair**: a chair where the users sit down
- **Seat**: whether someone use this desk
- **QuestionButton**: button for question
- **DeskCamera**: camera on the desk to change angle of the user to the screen board

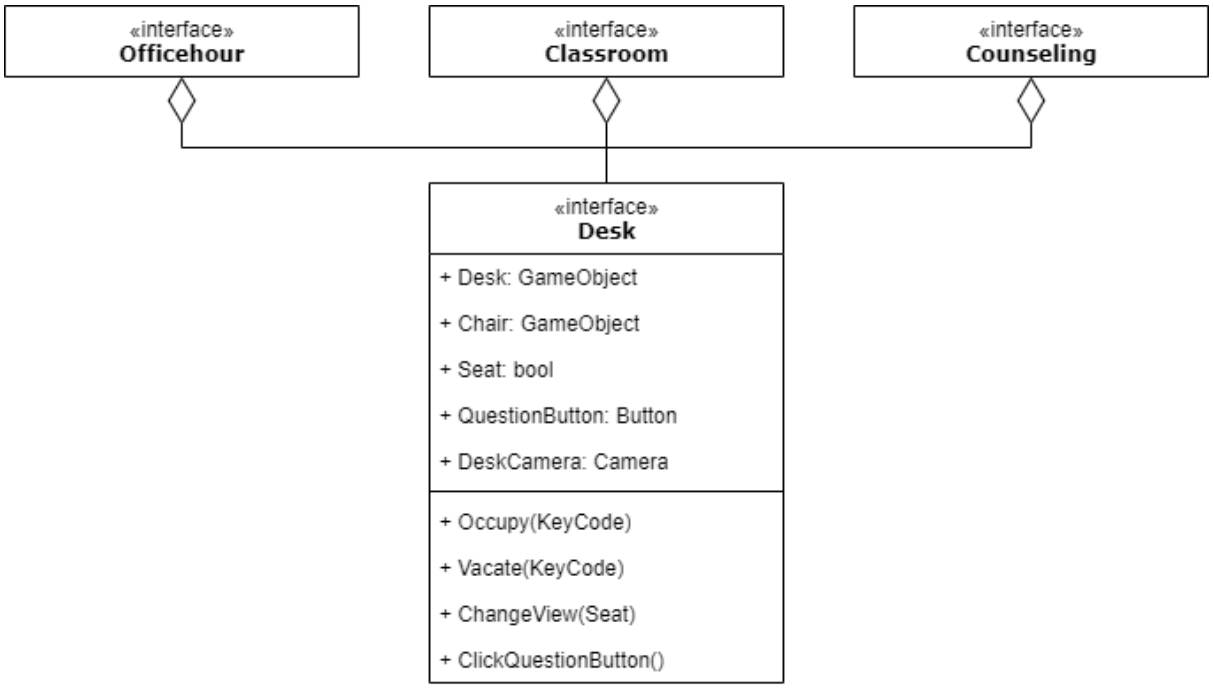
4.2.13.2.Methods

These are the methods that desk class has.

- Occupy()
- Vacate()
- ClickQuestionButton()
- ChangeView()

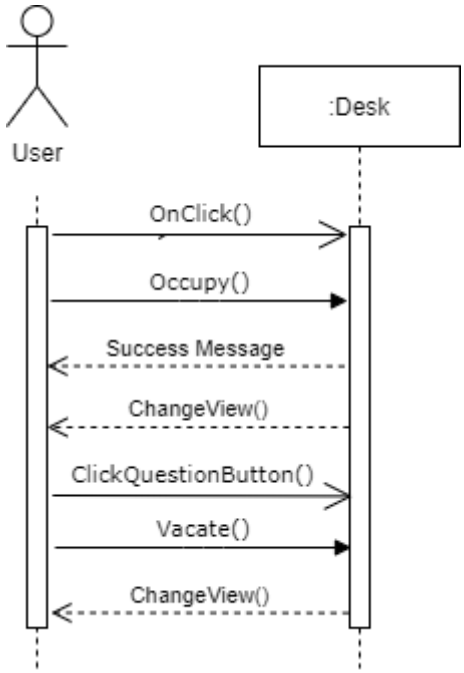
4.2.13.3.Class Diagram

[Diagram 28] Class diagram – Desk



4.2.13.4.Sequence Diagram

[Diagram 29] Sequence diagram – Desk



4.2.14. Automatic class enter

4.2.14.1.Attributes

These are the attributes that automatic class enter object has.

- **User id**: id of the user (student ID)
- **User timetable**: timetable of user in user's profile
- **Class number**: each class has a unique number to identify the space to move

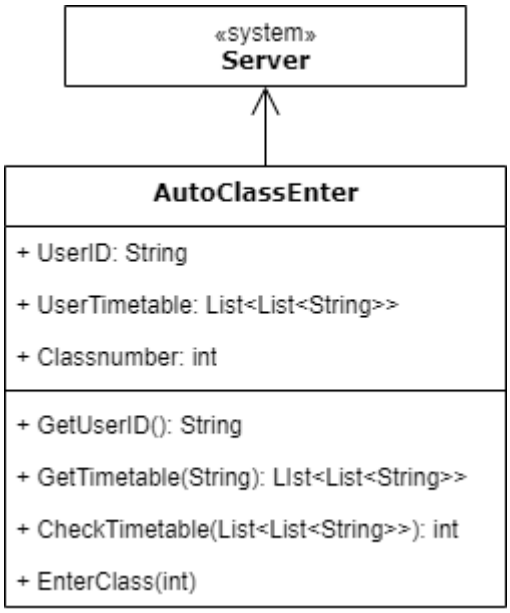
4.2.14.2.Methods

These are the methods that automatic class enter class has.

- GetUserID()
- GetTimetable()
- CheckTimetable()
- EnterClass()

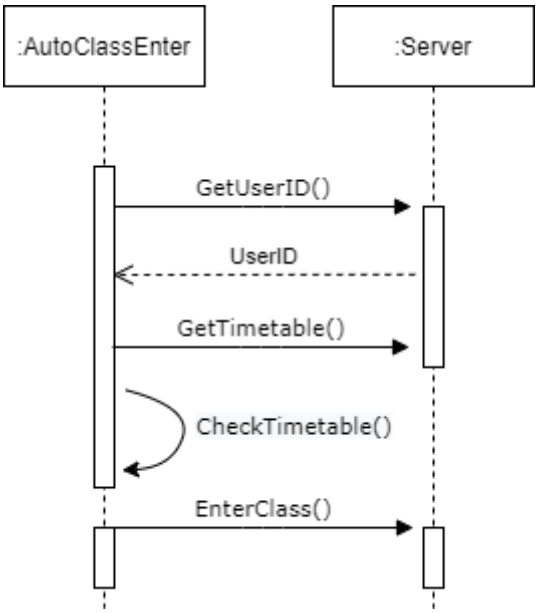
4.2.14.3.Class Diagram

[Diagram 30] Class diagram – Automatic classroom enter



4.2.14.4.Sequence Diagram

[Diagram 31] Sequence diagram – Automatic classroom enter



4.2.15. Reservation

4.2.15.1.Attributes

These are the attributes that reservation object has.

- **User id**: id of the user (student ID)
- **Timetable**: usable timetable
- **UsableTime**: to represent usable time for reservation
- **WhoReservate**: a window user can write booker's id
- **Booker**: list of users who reserve the space
- **Reservation**: context of reservation time and people

4.2.15.2.Methods

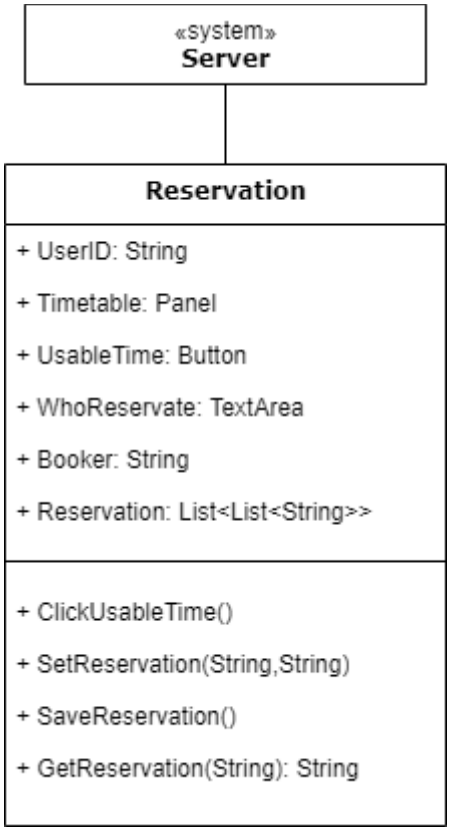
These are the methods that reservation class has.

- ClickUsableTime()

- SetReservation()
- SaveReservation()
- GetReservation()

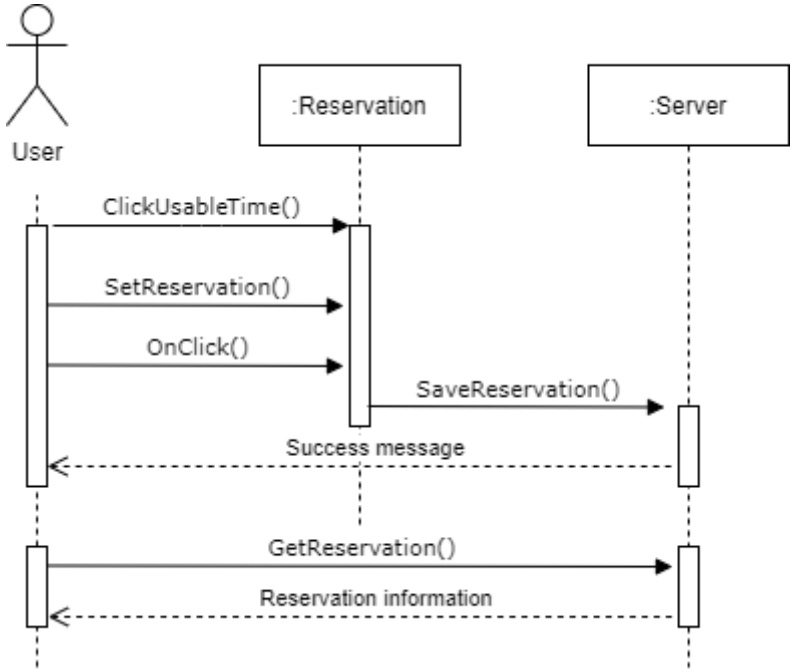
4.2.15.3.Class Diagram

[Diagram 32] Class diagram – Reservation



4.2.15.4.Sequence Diagram

[Diagram 33] Sequence diagram – Reservation



4.2.16. Office Hour

4.2.16.1.Attributes

These are the attributes that office hour object has.

- **Screenboard**: id of the user (email address)
- **Desk**: desk
- **Door**: door of office hour zone
- **InConversation**: whether some student users enter office hour zone

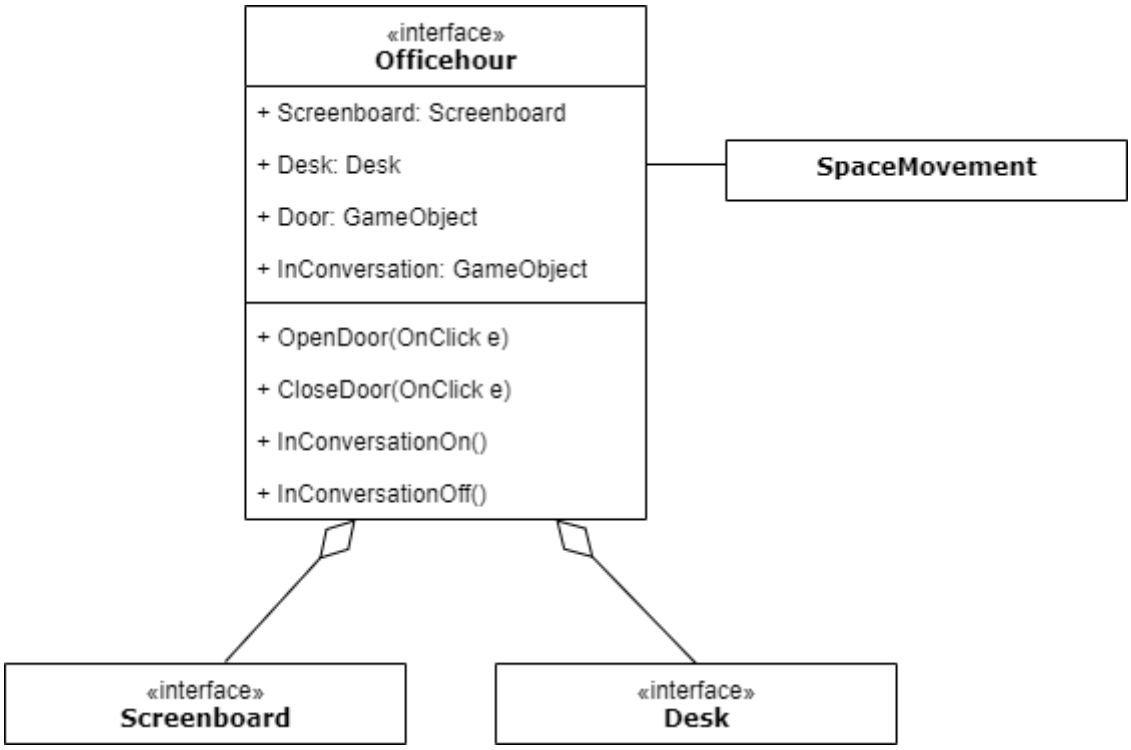
4.2.16.2.Methods

These are the methods that office hour class has.

- OpenDoor()
- CloseDoor()
- InConversationOn()
- InConversationOff()

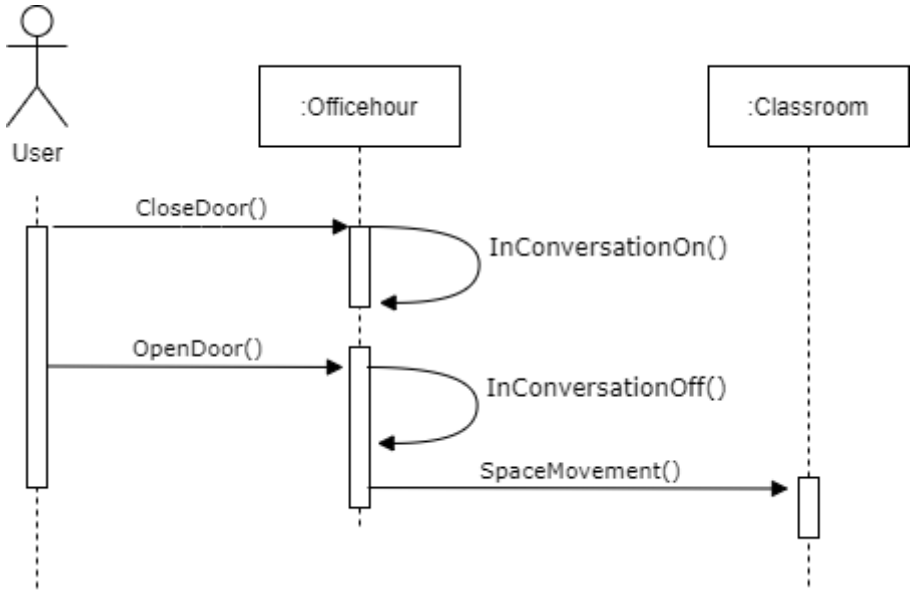
4.2.16.3.Class Diagram

[Diagram 34] Class diagram – Officehour



4.2.16.4.Sequence Diagram

[Diagram 35] Sequence diagram – Officehour



4.2.17. Counseling

4.2.17.1.Attributes

These are the attributes that counseling object has.

- **Screenboard**: screenboard
- **Desk**: desk
- **Door**: means of entering classroom

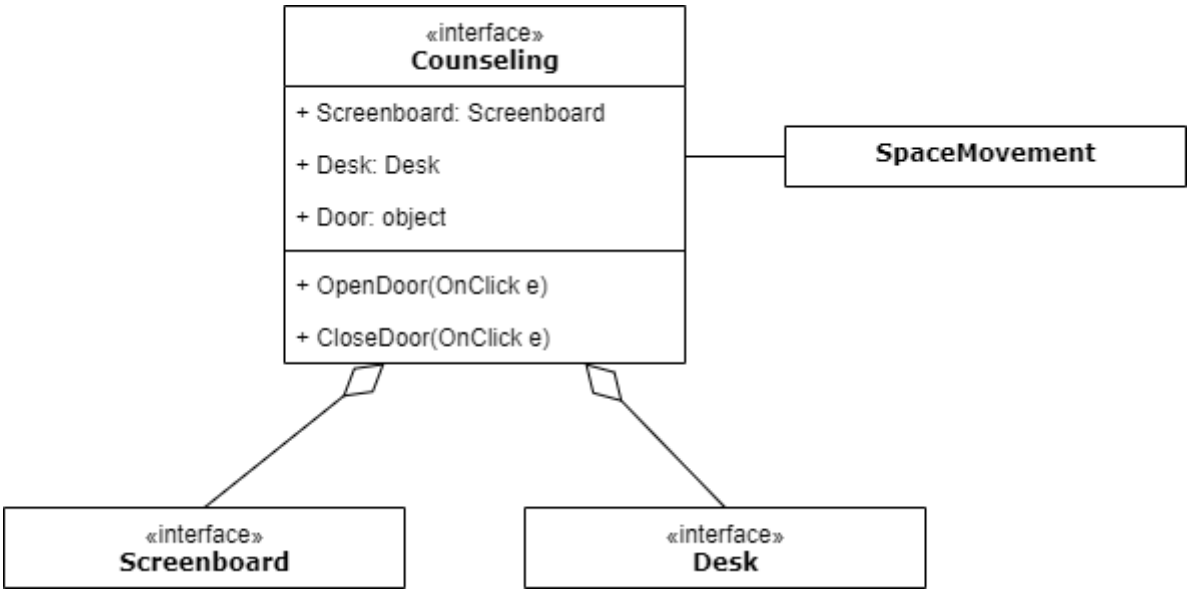
4.2.17.2.Methods

These are the methods that counseling class has.

- OpenDoor()
- CloseDoor()

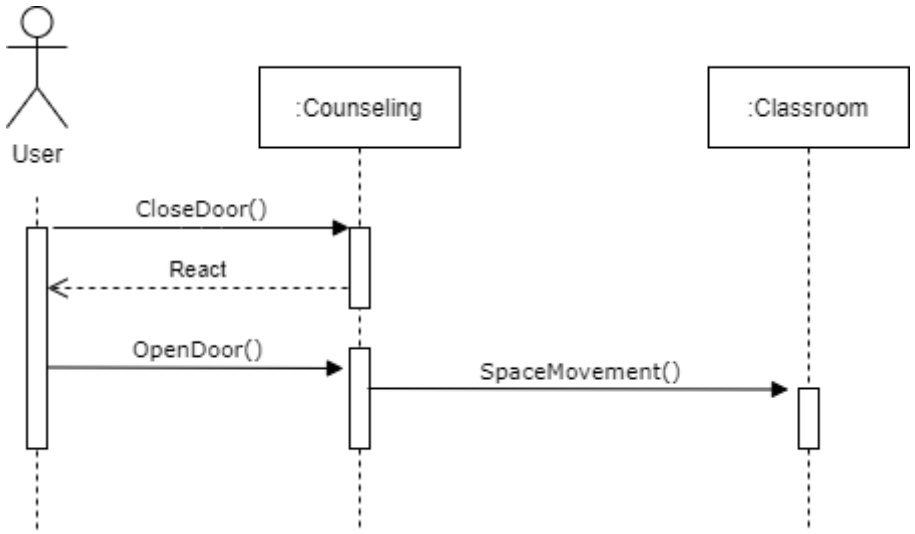
4.2.17.3.Class Diagram

[Diagram 36] Class diagram – Counseling



4.2.17.4.Sequence Diagram

[Diagram 37] Sequence diagram – Counseling



4.2.18. Bookshelf in classroom

4.2.18.1.Attributes

These are the attributes that bookshelf in classroom object has.

- **SearchPanel**: window where user can enter book name
- **SearchTextArea**: space to enter the book name in search panel
- **Searchedbook**: book name user searched
- **BookList:** result of searching
- **SearchResult**: a dropdown space to show results of searching
- **Textbook**: textbook of the class
- **Relatedbook**: related books of textbook.
- **Bookshelf**: panel of bookshelf
- **TextBookLabel**: show textbook
- **RelatedBookLabel**: show related book

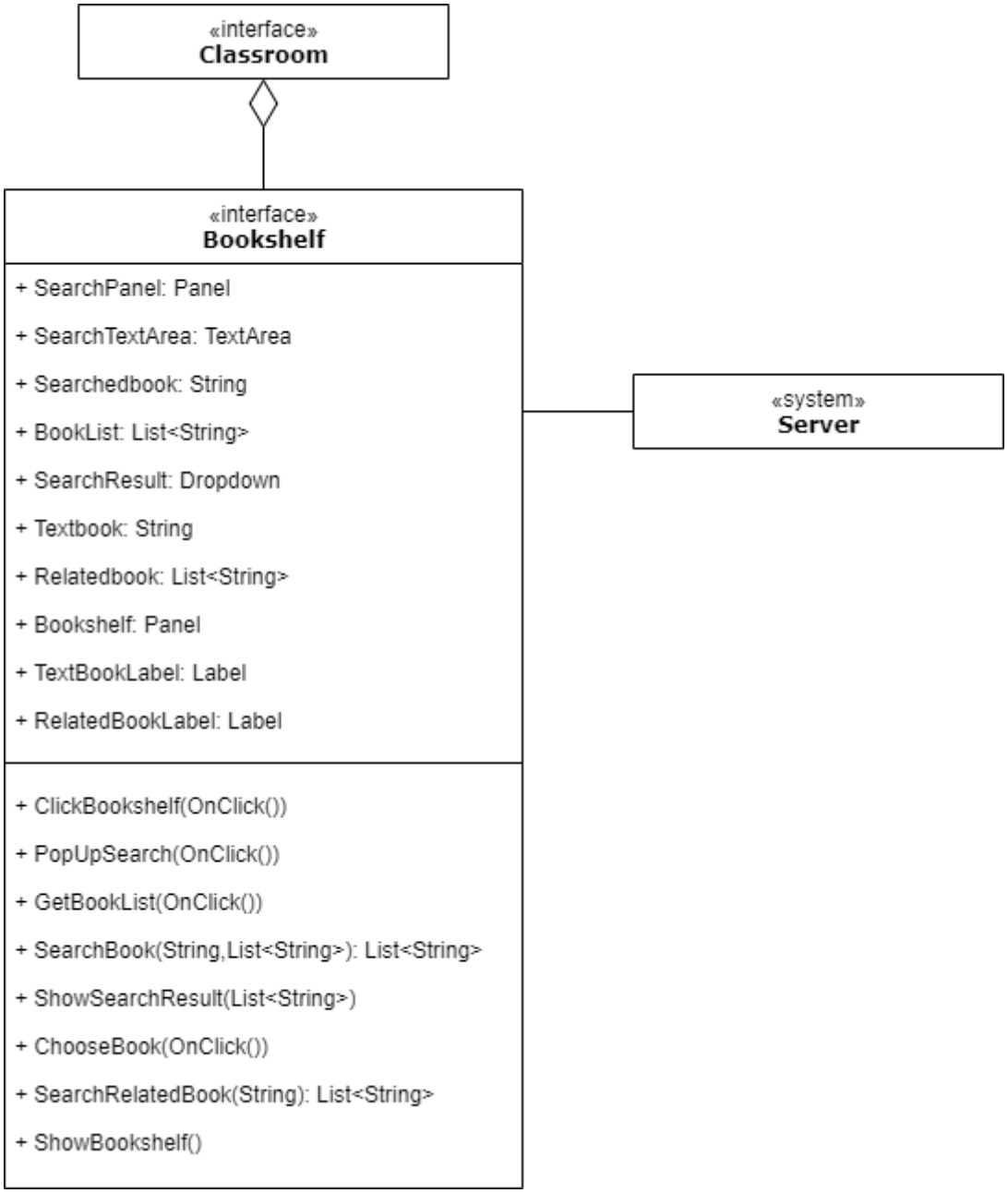
4.2.18.2.Methods

These are the methods that bookshelf in classroom class has.

- ClickBookshelf()
- PopUpSearch()
- GetBookList()
- SearchBook()
- ShowSearchResult()
- ChooseBook()
- SearchRelatedBook()
- ShowBookshelf()

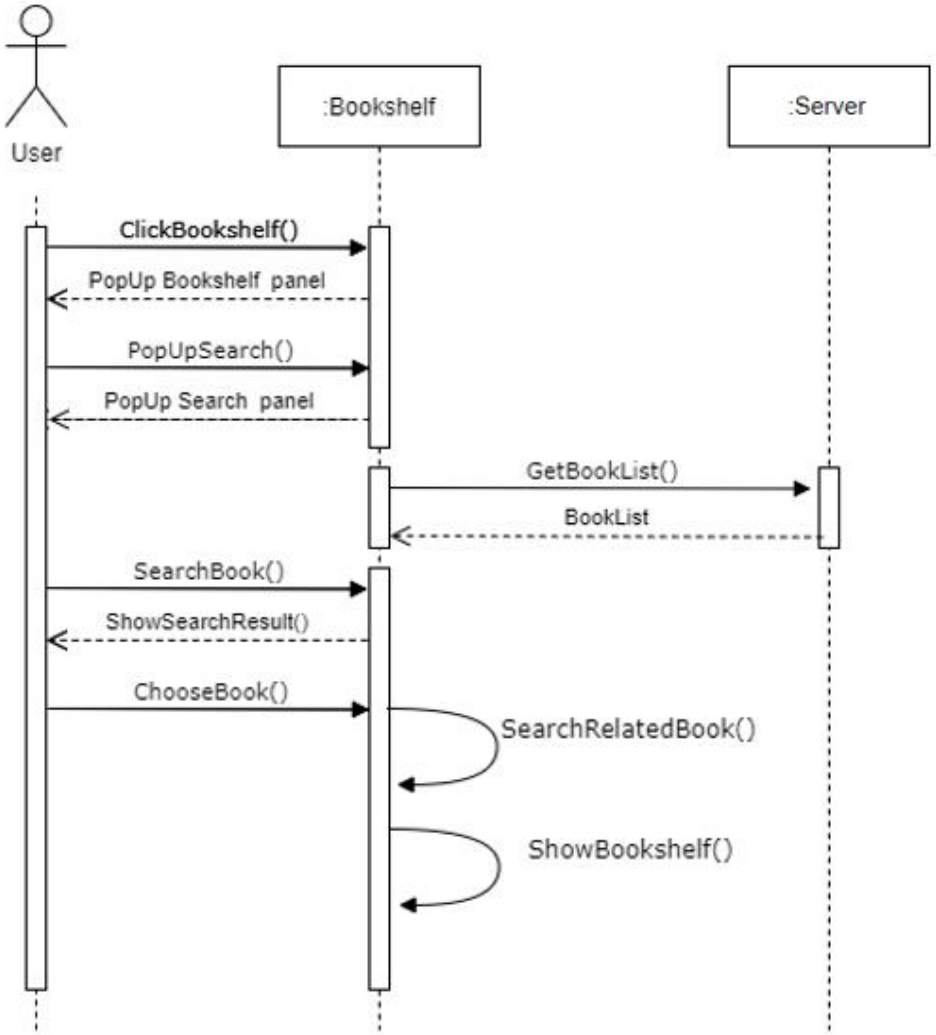
4.2.18.3.Class Diagram

[Diagram 38] Class diagram – Bookshelf in classroom



4.2.18.4.Sequence Diagram

[Diagram 39] Sequence diagram – Bookshelf in classroom



4.2.19. Elevator

4.2.19.1.Attributes

These are the attributes that elevator object has.

- **floor**: current floor in library
- **upbtn**: button to go up
- **downbtn**: button to go down

- **openbtn**: button to open the elevator
- **closebtn**: button to close the elevator
- **btns**: button to choose the number of floors

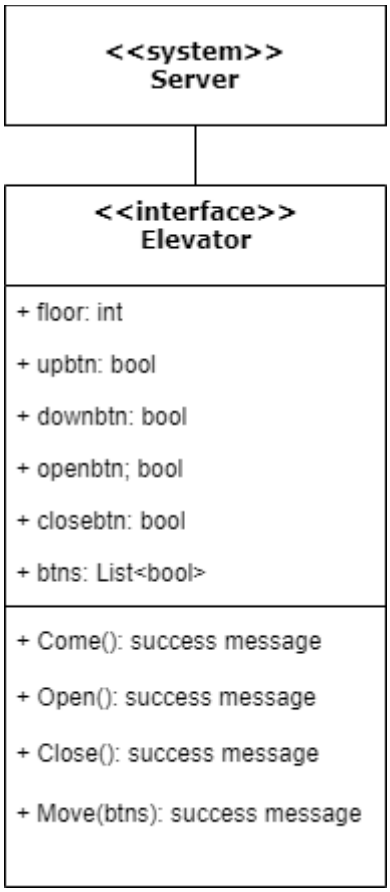
4.2.19.2.Methods

These are the methods that elevator class has.

- Come()
- Open()
- Close()
- Move(btns)

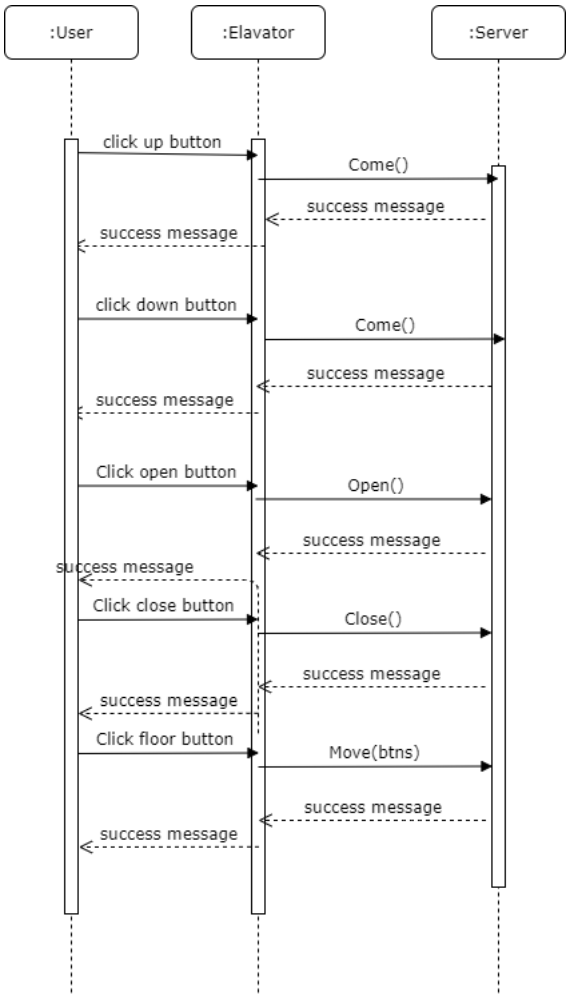
4.2.19.3.Class Diagram

[Diagram 40] Class diagram – Elevator



4.2.19.4.Sequence Diagram

[Diagram 41] Sequence diagram – Elevator



4.2.20. Bookshelves in Library

4.2.20.1.Attributes

These are the attributes that bookshelves in library object has.

- **user_id**: id of the user (email address)
- **click**: whether user clicked bookshelves or not
- **searchbtn**: button to search a desired book
- **book**: book in the library

- **borrowbtn**: button to borrow book
- **returnbtn**: button to return book
- **availablebook**: book that can be rented

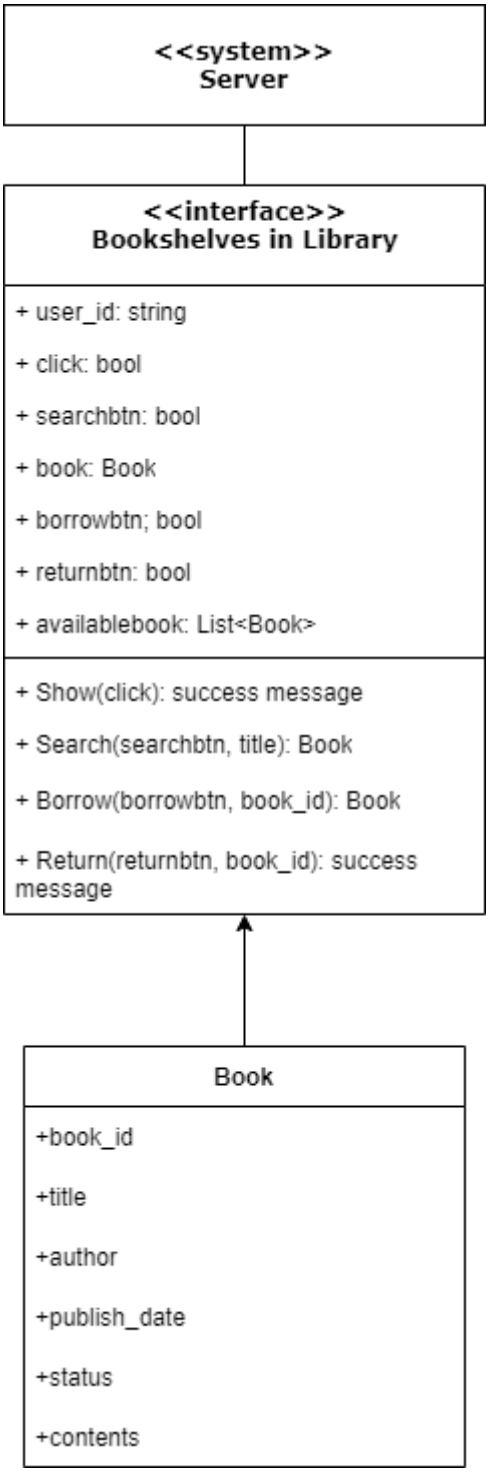
4.2.20.2.Methods

These are the methods that bookshelves in library class has.

- Show(click)
- Search(searchbtn, title)
- Borrow(borrowbtn, book_id)
- Return(returnbtn, book_id)

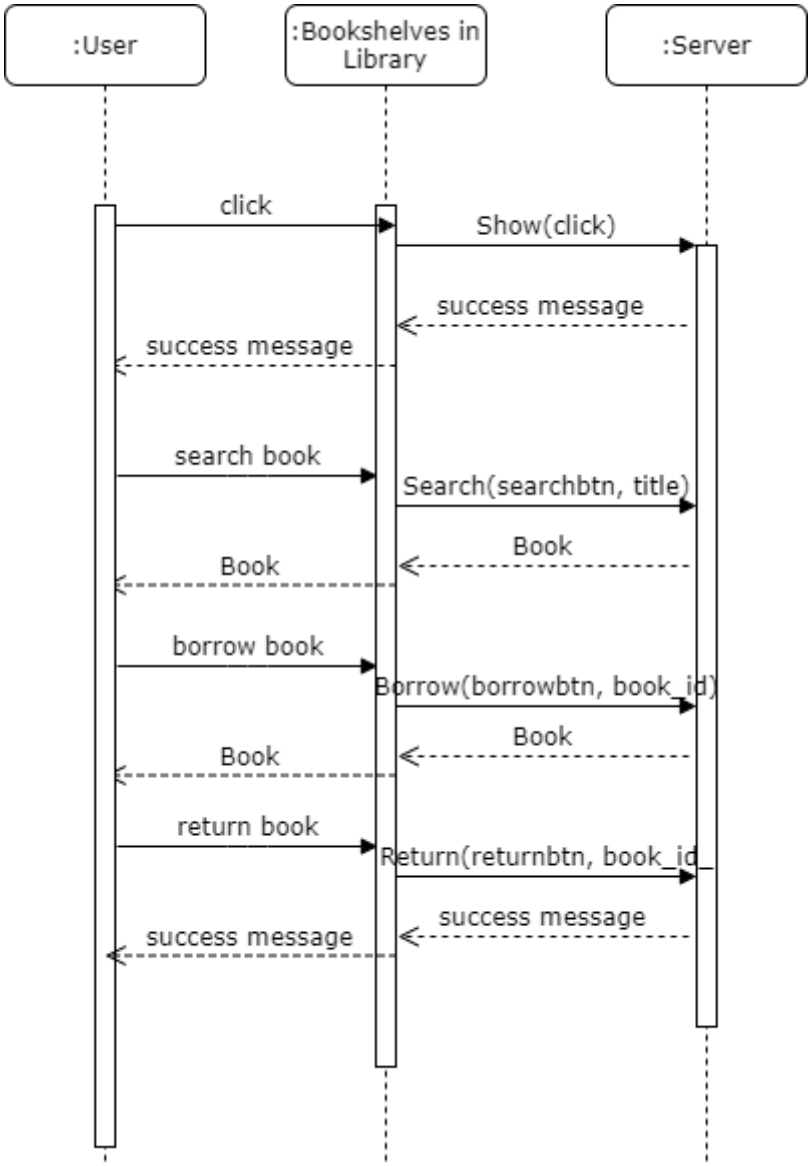
4.2.20.3.Class Diagram

[Diagram 42] Class diagram – Bookshelves in Library



4.2.20.4.Sequence Diagram

[Diagram 43] Sequence diagram – Bookshelves in Library



4.2.21. Posting space

4.2.21.1.Attributes

These are the attributes that posting space object has.

- **user_id**: id of the user (email address)
- **image**: url of image file where his/her post is uploaded
- **savebtn**: button to save image file where his/her post is uploaded

4.2.21.2.Methods

These are the methods that posting space class has.

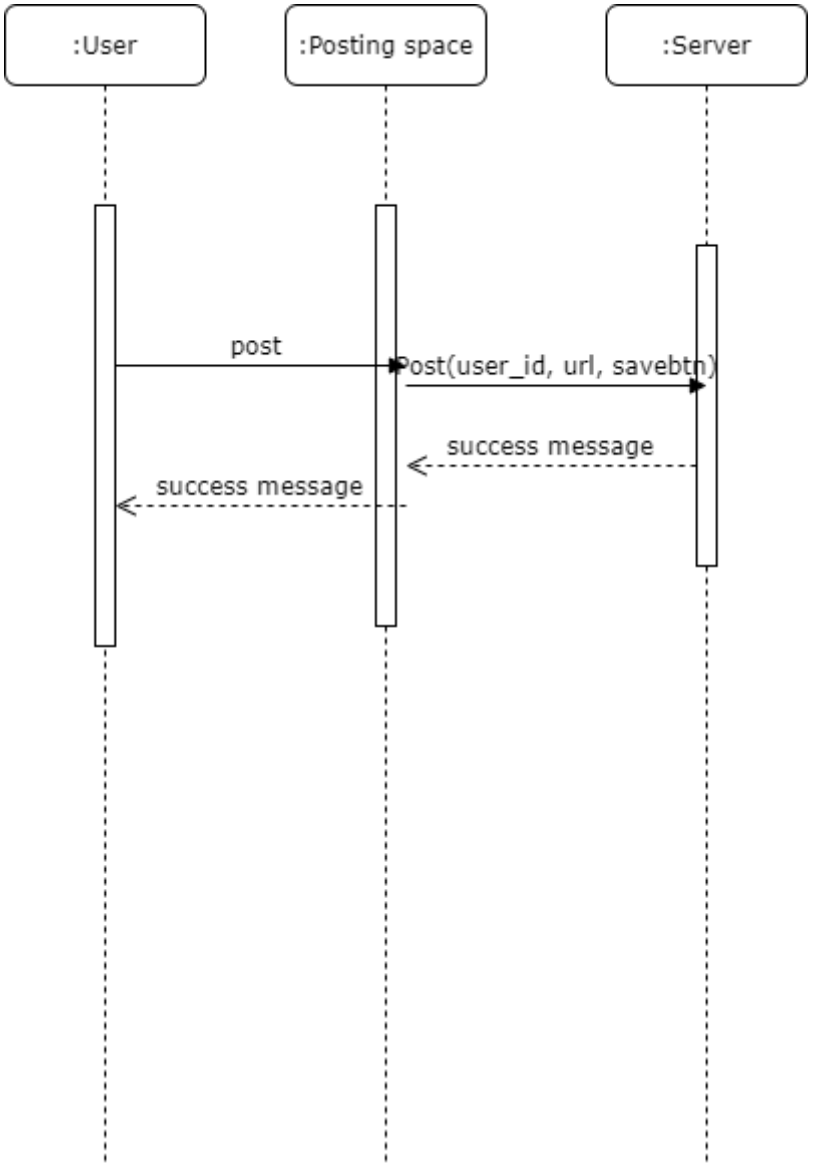
- Post(user_id, url, savebtn)

4.2.21.3.Class Diagram

[Diagram 44] Class diagram – Posting space

4.2.21.4.Sequence Diagram

[Diagram 45] Sequence diagram – Posting space



4.2.22. Reading room

4.2.22.1.Attributes

These are the attributes that reading room object has.

- **user_id**: id of the user (email address)
- **downbtn**: button to register an empty seat
- **upbtn**: button to leave out from the seat
- **Xbtn**: button to close the window guiding study time

- **rank**: high 10 users

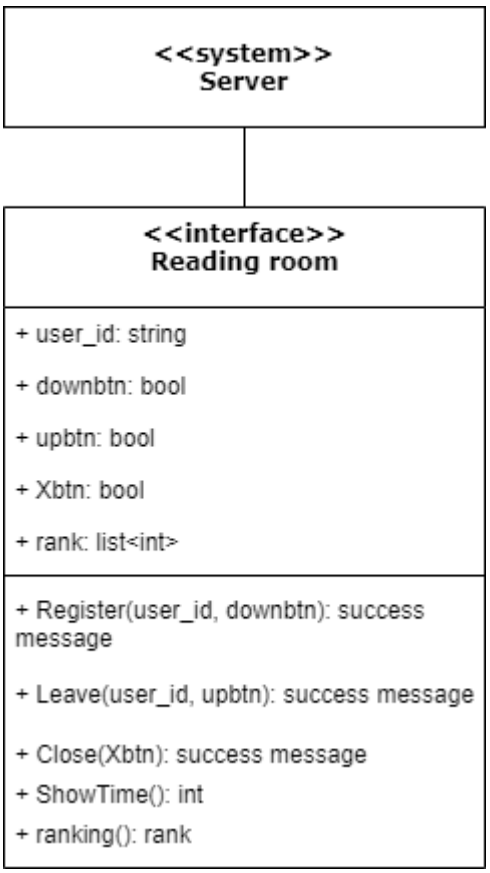
4.2.22.2.Methods

These are the methods that reading room class has.

- Register(user_id, downbtn)
- Leave(user_id, upbtn)
- Close(Xbtn)
- ShowTime()
- ranking()

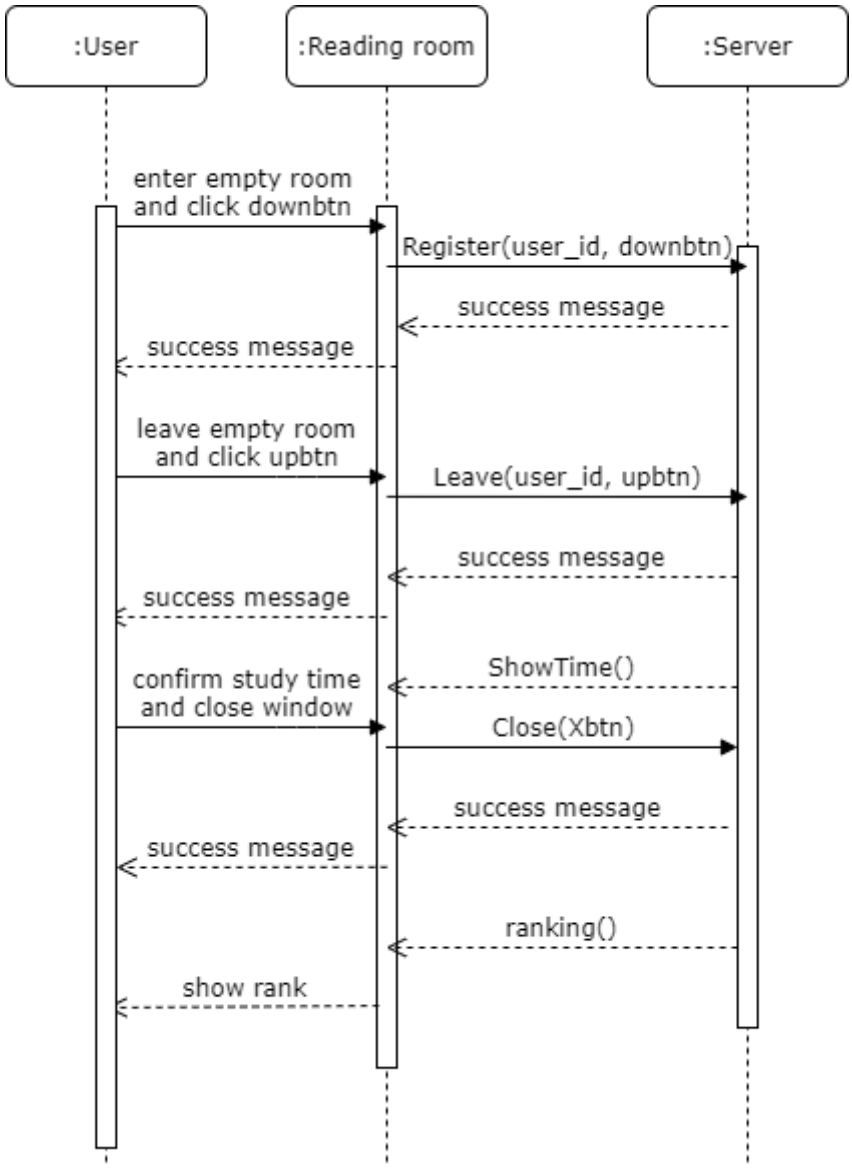
4.2.22.3.Class Diagram

[Diagram 46] Class diagram – Reading room



4.2.22.4.Sequence Diagram

[Diagram 47] Sequence diagram – Reading room



4.2.23. Study room

4.2.23.1.Attributes

These are the attributes that study room object has.

- **user_id**: id of the user (email address)
- **shareBoard**: screen board to share some data as image file

- **chatBoard**: discuss in the form of chat on the chatting board or voice
- **finishbtn**: button to leave out study room
- **whichRoom**: an empty study room chosen by the user

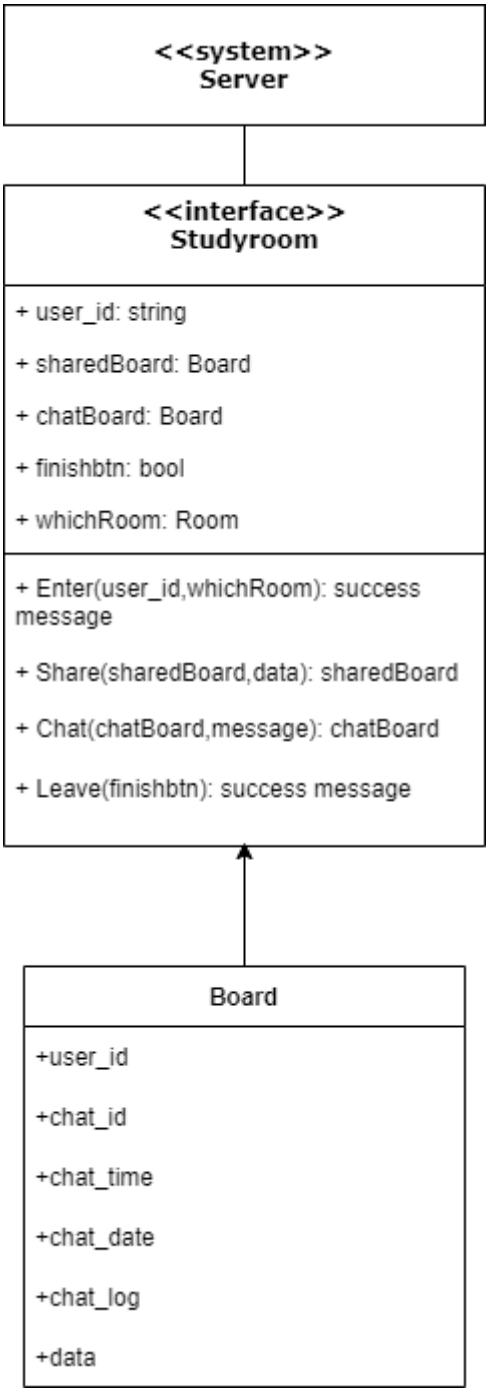
4.2.23.2.Methods

These are the methods that study room class has.

- Enter(user_id, whichRoom)
- Share(sharedBoard,data)
- Char(chatBoard, message)
- Leave(finishbtn)

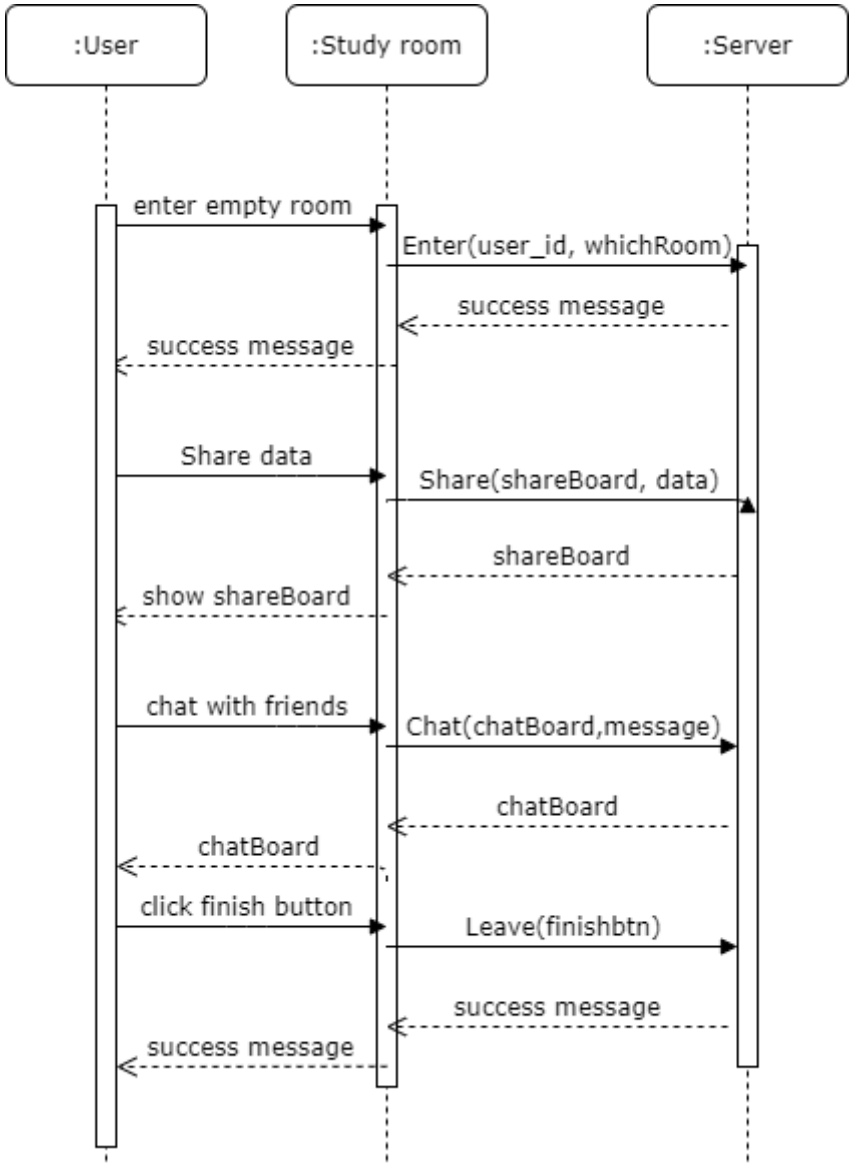
4.2.23.3.Class Diagram

[Diagram 48] Class diagram – Study room



4.2.23.4.Sequence Diagram

[Diagram 49] Sequence diagram – Study room



4.2.24. Exhibition

4.2.24.1.Attributes

These are the attributes that exhibition object has.

- **user_id**: id of the user (email address)
- **savebtn**: button to post image or work
- **image**: url of image file where his/her post is uploaded
- **message**: message, impressions or questions about work

- **move**: button to leave out the exhibition hall
- **works**: the list of work from database
- **vote**: the number of times people clicked on a work
- **item**: hidden items in the descriptions of the work and the presentation videos
- **position**: the location of hidden items

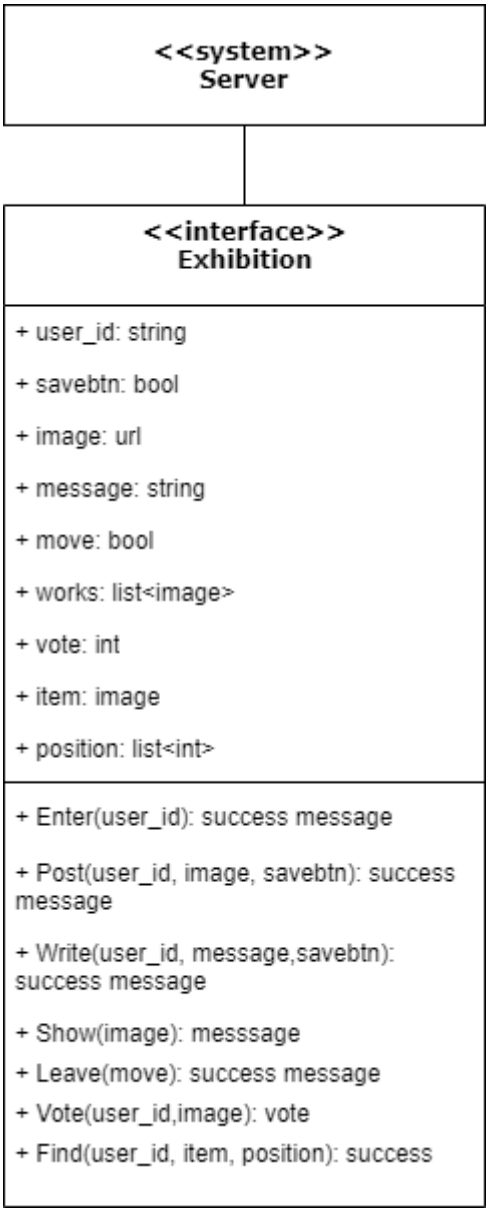
4.2.24.2.Methods

These are the methods that exhibition class has.

- Enter(user_id)
- Post(user_id,image,savebtn)
- Write(user_id, message, savebtn)
- Show(image)
- Leave(move)
- Vote(user_id, image)
- Find(user_id, item, position)

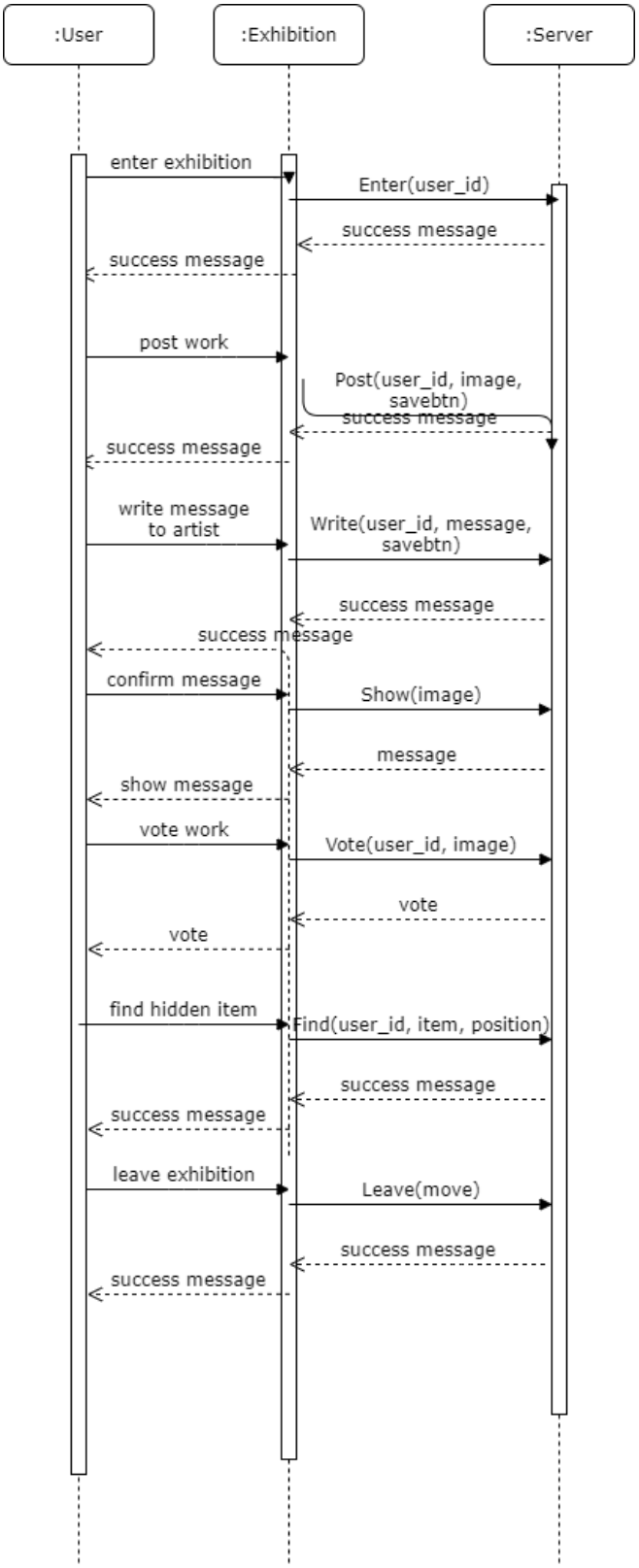
4.2.24.3.Class Diagram

[Diagram 50] Class diagram – Exhibition



4.2.24.4.Sequence Diagram

[Diagram 51] Sequence diagram – Exhibition


:User
:Exhibition
:Server
enter exhibition
Enter(user_id)
success message
success message
post work
Post(user_id, image, savebtn)
success message
success message
write message to artist
Write(user_id, message, savebtn)
success message
success message
confirm message
Show(image)
message
show message
vote work
Vote(user_id, image)
vote
vote
find hidden item
Find(user_id, item, position)
success message
success message
leave exhibition
Leave(move)
success message
success message

4.2.25. Event

4.2.25.1.Attributes

These are the attributes that event object has.

- **user_id**: id of the user (email address)
- **item**: hidden item in the exhibition room
- **position**: the location of hidden items

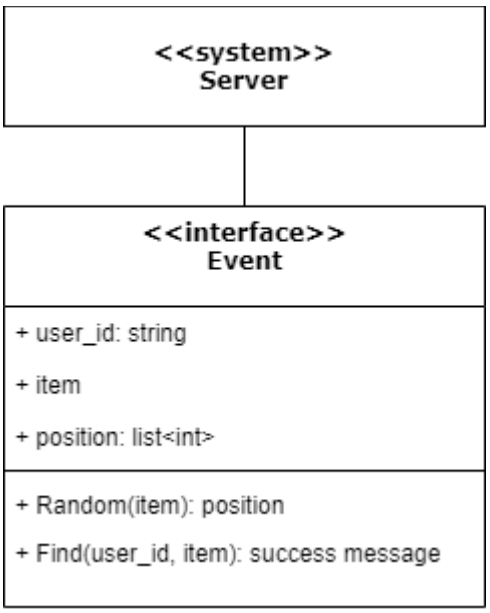
4.2.25.2.Methods

These are the methods that event class has.

- Random(item)
- Find(user_id, item)

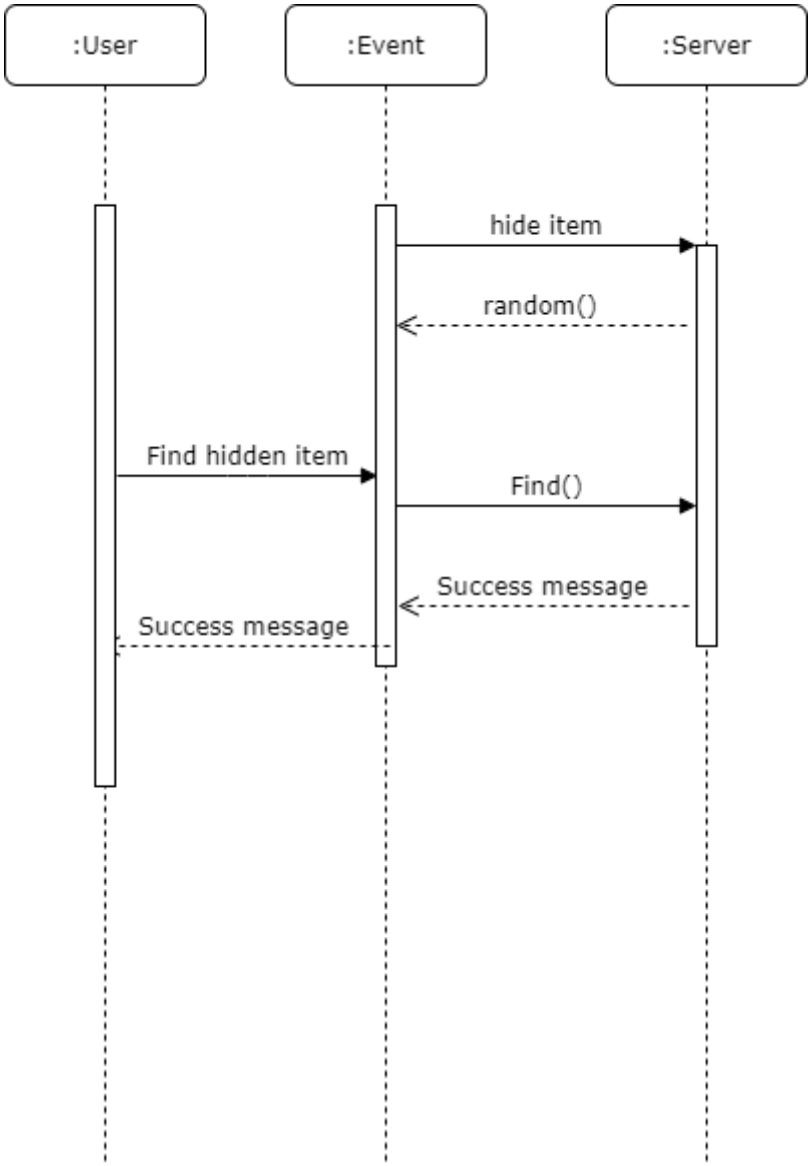
4.2.25.3.Class Diagram

[Diagram 52] Class diagram – Event



4.2.25.4.Sequence Diagram

[Diagram 53] Sequence diagram – Event



5. System Architecture - Backend

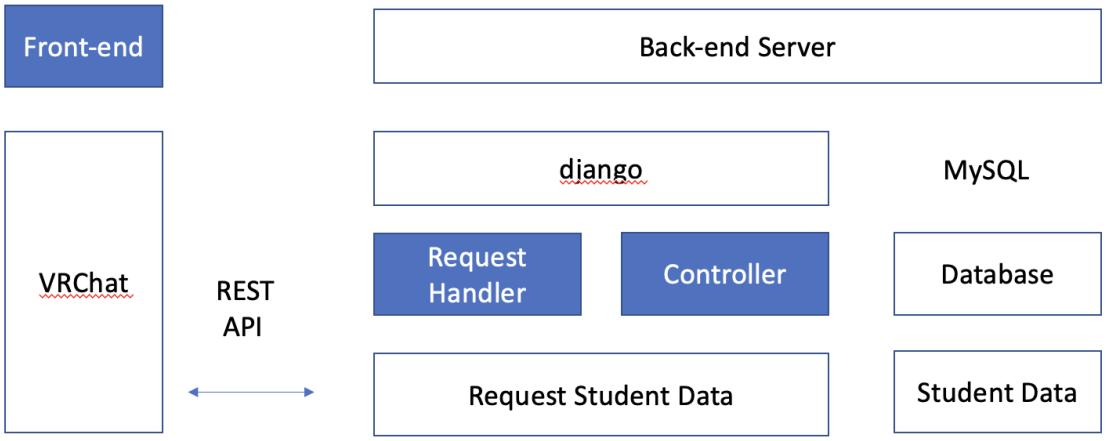
5.1. Objectives

This contains information on the backend except for the frontend that makes direct contact with the user. It shows the structure in which the backend system and each subsystem are connected. It mainly represents a model and several subsystems in which the server runs.

5.2. Overall Architecture

The overall architecture of the system is as below.

[Figure 54] Overall architecture



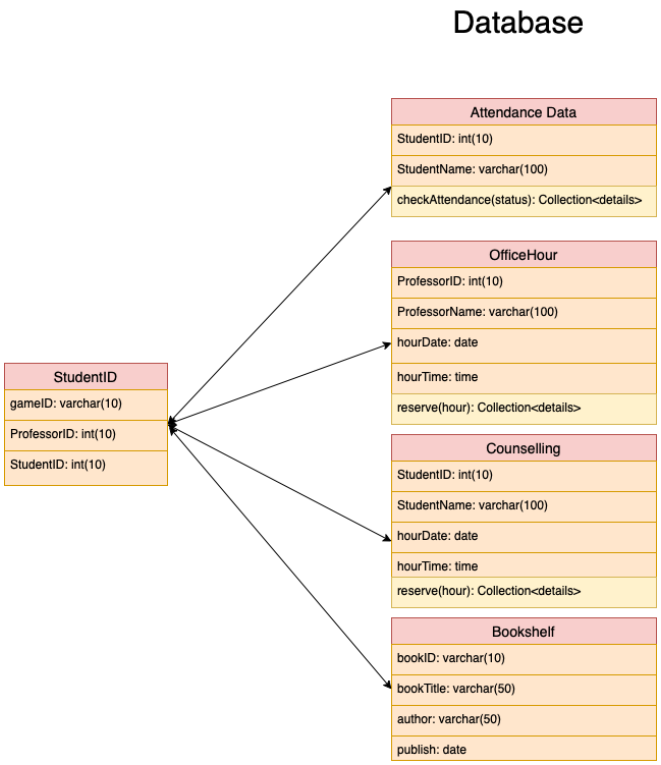
5.3. Subcomponent

5.3.1. Server

5.3.1.1. Class Diagram

The class diagram of server is as below.

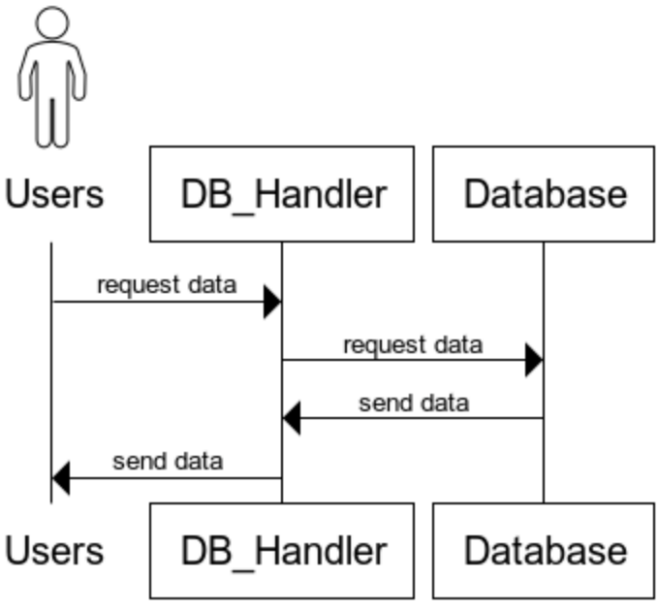
[Diagram 55] Class diagram of server



5.3.1.2. Sequence Diagram

The sequence diagram of server is as below.

[Diagram 56] Sequence diagram of server

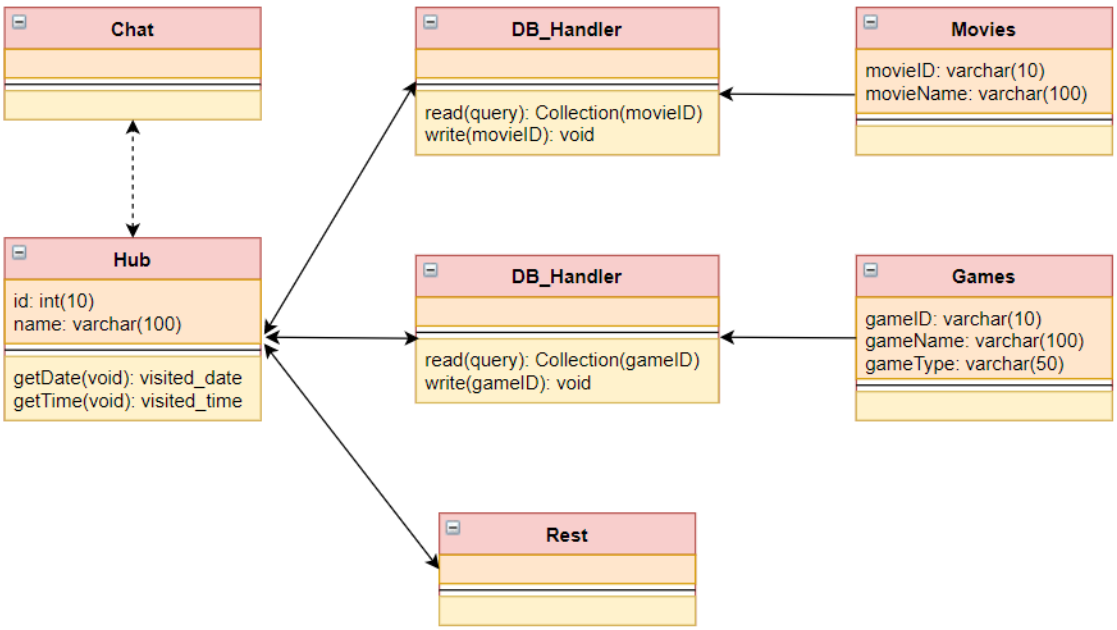


5.3.2. Hub

5.3.2.1. Class Diagram

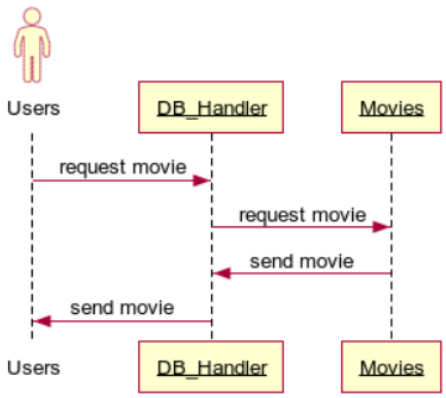
Diagram below is the interface for Hub. It allows users to get a selection of games and movies. Also, the Hub system is related to the Chat class and Rest class.

[Diagram 57] Class diagram of Hub system

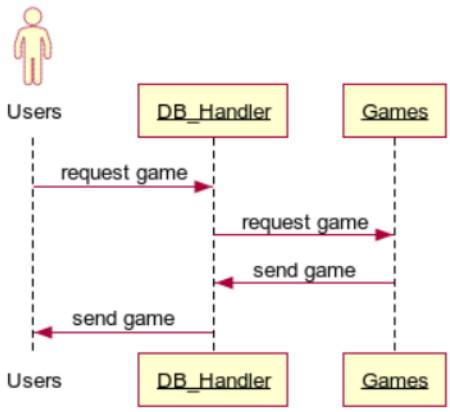


5.3.2.2. Sequence Diagram

[Diagram 58] Sequence diagram of movie system in Hub



[Diagram 59] Sequence diagram of game system in Hub

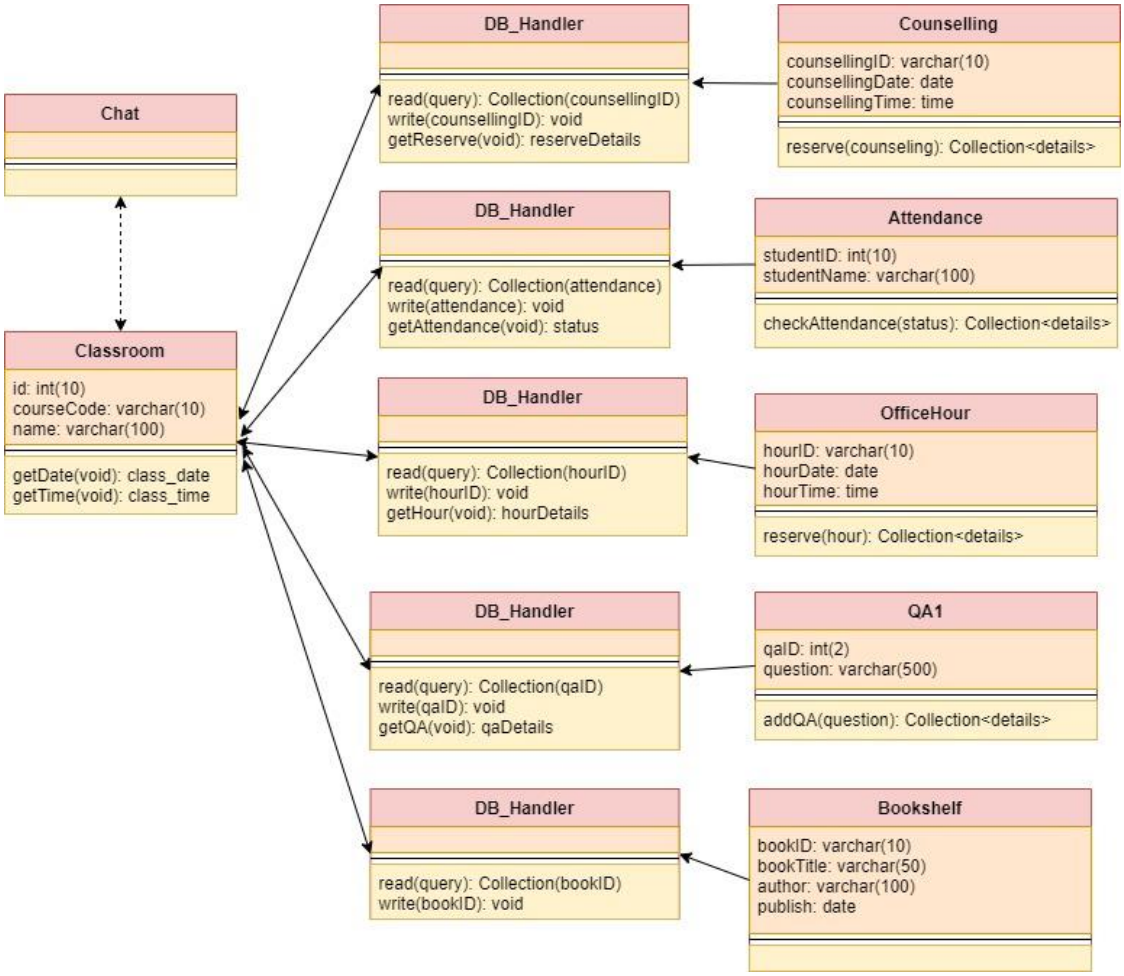


5.3.3. Classroom

5.3.3.1. Class Diagram

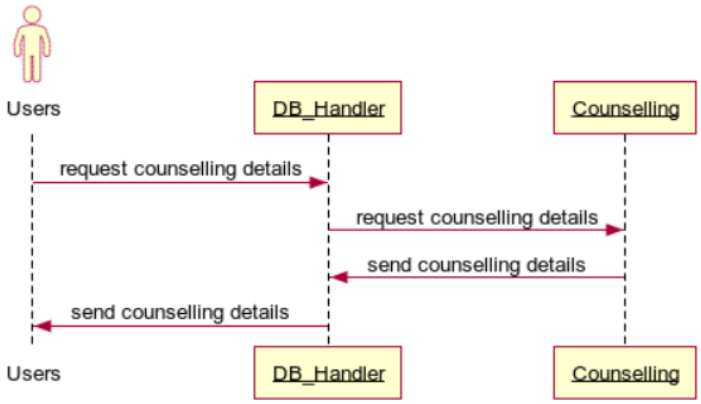
Interface for Classroom is shown as below diagram. It allows Professors to get counselling, attendance, office hour, and questions and answers information. Also, students are allowed to access the Bookshelf class to retrieve the books they want. Furthermore, in Counselling, Office Hour, and QA1 class, students are able to enter information for the respective purposes. In the Attendance class, there is a function to check the attendance of the students. Also, the Classroom system is related to the Chat class.

[Diagram 60] Class diagram of Classroom system

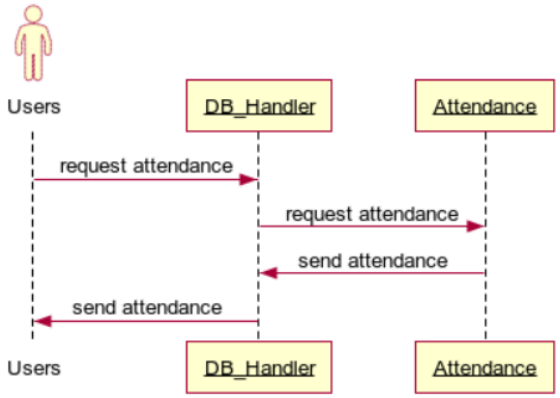


5.3.3.2. Sequence Diagram

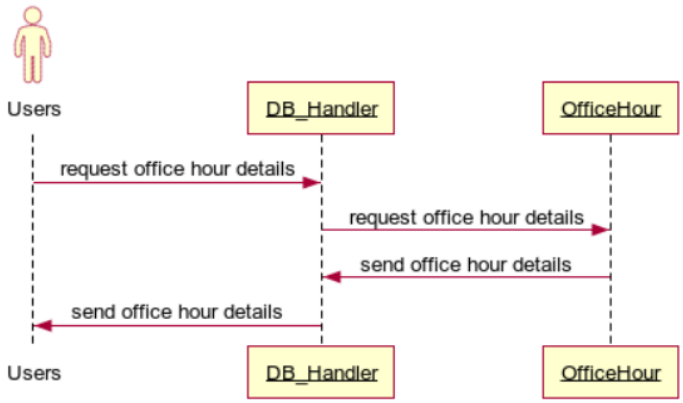
[Diagram 61] Sequence diagram of Counseling system in Classroom



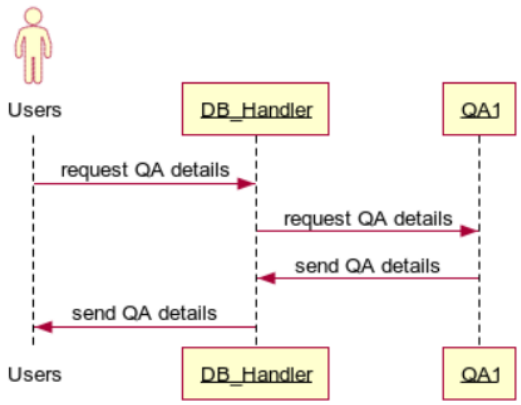
[Diagram 62] Sequence diagram of Attendance system in Classroom



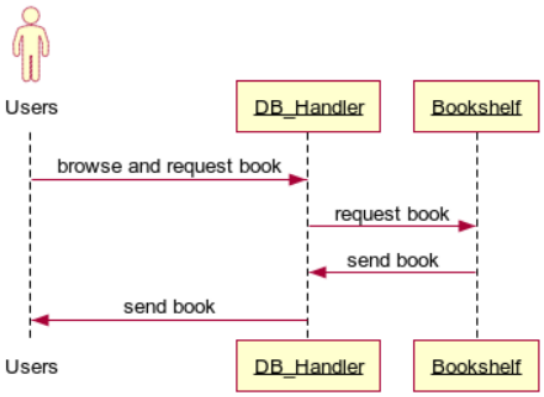
[Diagram 63] Sequence diagram of Office Hour system in Classroom



[Diagram 64] Sequence diagram of QA1 system in Classroom



[Diagram 65] Sequence diagram of Bookshelf system in Classroom

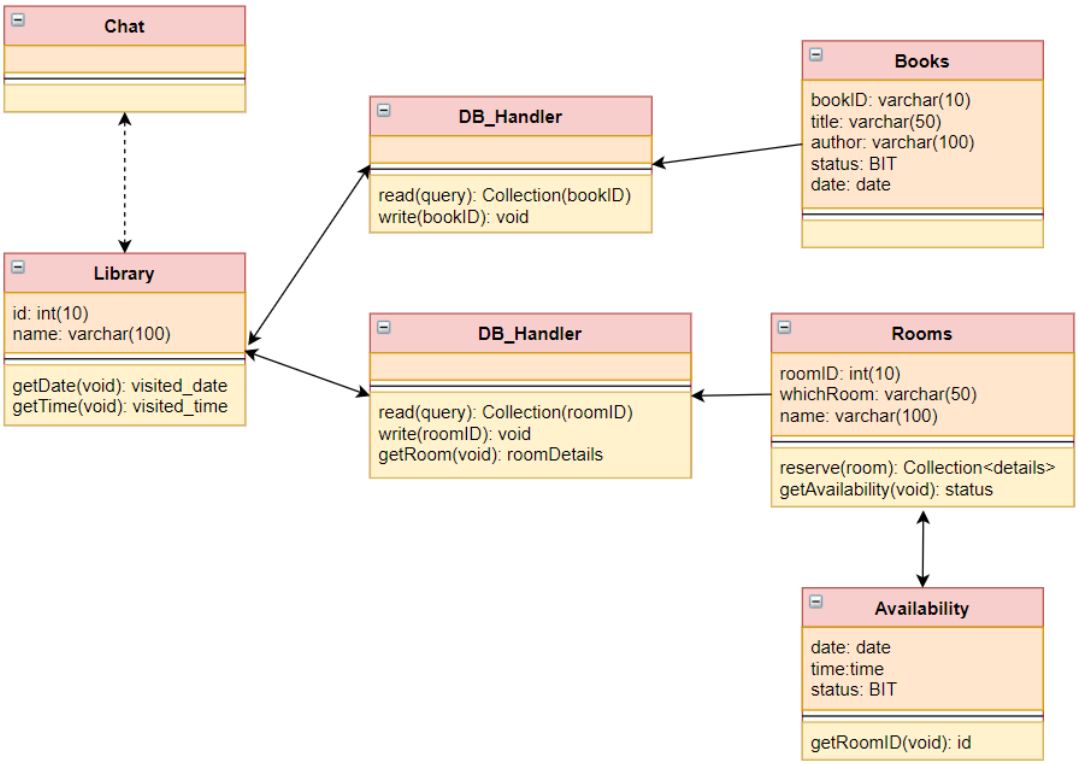


5.3.4. Library

5.3.4.1. Class Diagram

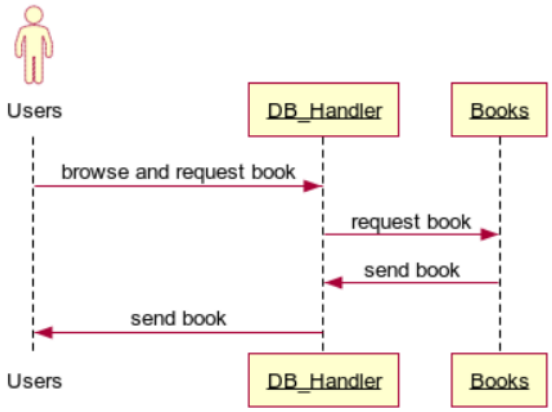
Below diagram shows the interface for Library. Students can browse and request for books for their choice by the Book class. Also, students can reserve study or reading room by the Rooms class. The Rooms class will first check the availability of the chosen room by Availability class before approving it to students. Also, the Library system is related to the Chat class.

[Diagram 66] Class diagram of Library system

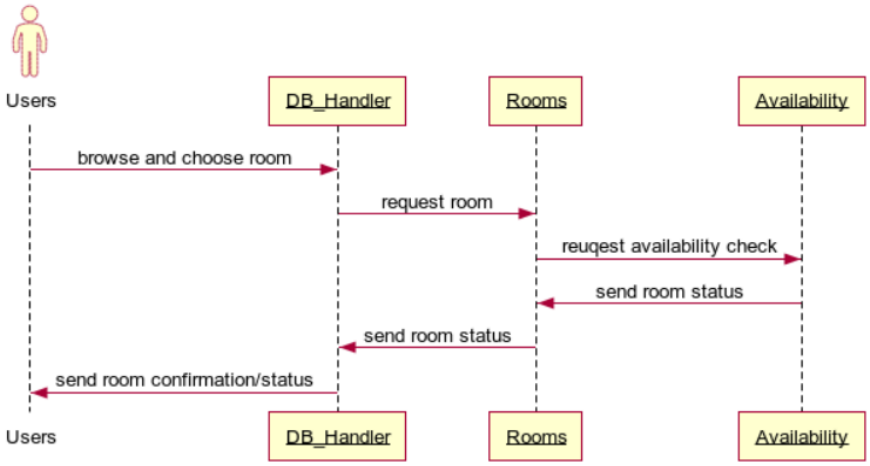


5.3.4.2. Sequence Diagram

[Diagram 67] Sequence diagram of Book system in Library



[Diagram 68] Sequence diagram of Rooms system in Library

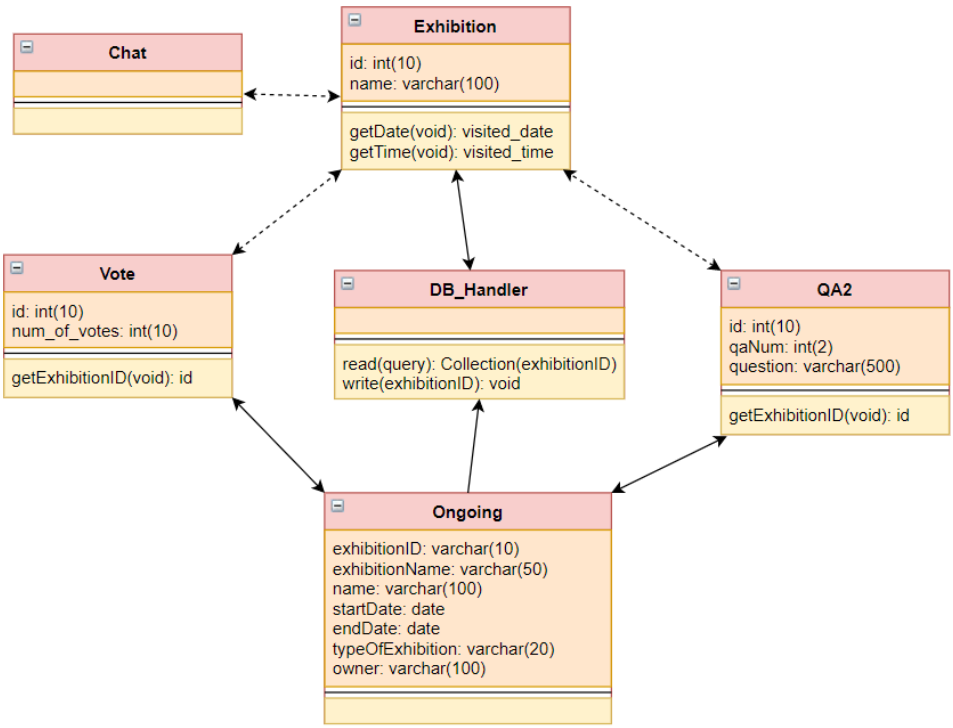


5.3.5. Exhibition Hall

5.3.5.1. Class Diagram

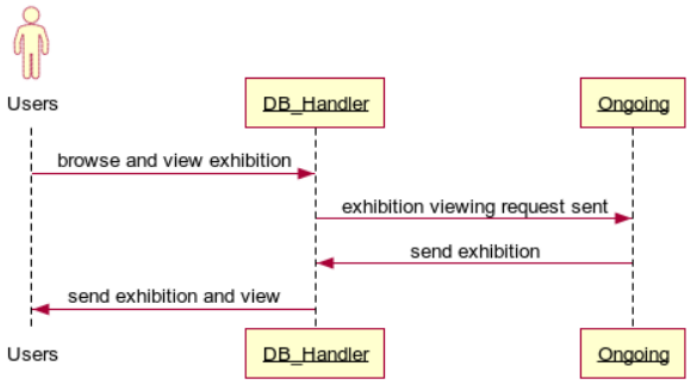
Here is the interface for Exhibition Hall. Students can browse through the ongoing exhibition by the Ongoing class. Furthermore, students can vote and ask questions about the exhibitions through the Vote and QA2 classes respectively. Also, the Exhibition Hall system is related to the Chat class.

[Diagram 69] Class diagram of Exhibition hall system

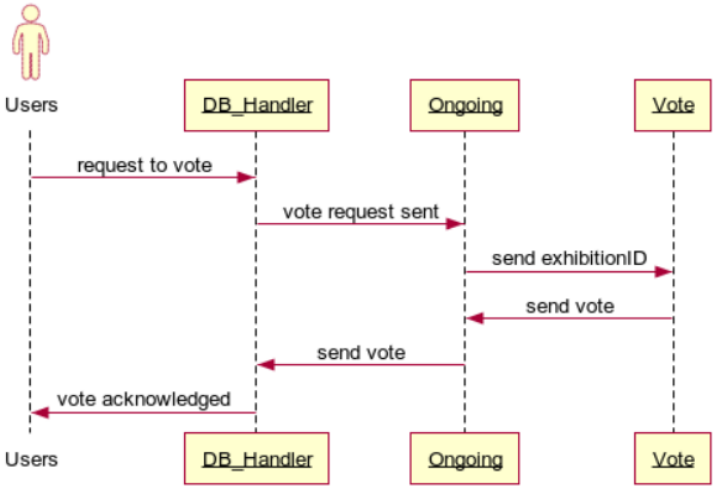


5.3.5.2. Sequence Diagram

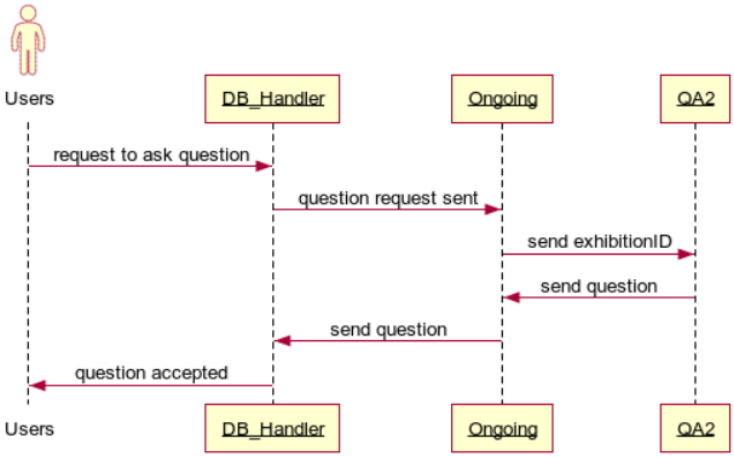
[Diagram 70] Sequence diagram of Ongoing system in Exhibition hall



[Diagram 71] Sequence diagram of Vote system in Exhibition hall



[Diagram 72] Sequence diagram of QA2 system in Exhibition hall



6. Protocol Design

6.1. Objective

This chapter describes what structures are used for the protocols which are used for interaction between each subsystem, especially the Campus Hub and the server. Also, this chapter describes how each interface is defined.

6.2. JSON

JSON(JavaScript Object Notation) is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types or any other serializable value. It is a very common data format, with a diverse range of applications, such as serving as a replacement for XML in AJAX systems.

All database data used in the Campus Hub is stored as JSON objects. Since the HTTP protocol is limited in VRChat, this service partially utilizes SQL database in table format. When needed data are crawled in JSON format, some of them would be parsed in related table data.

6.3. Details

6.3.1. Authentication

Each user ID(student ID or professor ID) prove belong to the specific class or major. So far, users can approach to matched class which has a unique class number. User ID is a main identification factor of the user in the campus hub when they do something that need to link to database.

6.3.2. Student information for class

6.3.2.1. Planned Class

- Request

[Table 2] Table of class list request

Attribute	Detail	
Method	GET	
URI	/api/class/{id}	
Parameter	Basic information of class	
Request Body	Class_Lists	Class id
Header	Authorization	User authentication

- Response

[Table 3] Table of class list response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Class_Lists	List objects
Failure Response Body	message	Empty

6.3.2.2. Class Attendance

- Request

[Table 4] Table of associated users of class request

Attribute	Detail	
Method	GET	
URI	/api/class/id/users	
Parameter	Class information (id)	
Request Body	Class_Attendance	User information (students, professor)
Header	Authorization	User authentication

- Response

[Table 5] Table of associated users of class response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Class_Attendance	Associated Users List objects
Failure Response Body	message	Empty

6.3.3. Movie in Hub

We would use utilize URL of Youtube video to watch movie.

- Request

[Table 6] Table of Youtube movie request

Attribute	Detail	
Method	GET	
URI	/api/movie	
Parameter	-	
Request Body	Movie_URL	url information of pre-stored Youtube video
Header	Authorization	User authentication

- Response

[Table 7] Table of Youtube movie response

Attribute	Detail
Success Code	200 OK
Failure Code	HTTP error code = 404 (Not Found)

Attribute	Detail	
Success Response Body	Movie_URL	Selected movie
Failure Response Body	message	Empty

6.3.4. Book Information

6.3.4.1. Bookshelves in Library

- Request

[Table 8] Table of book list request

Attribute	Detail	
Method	GET	
URI	/api/book	
Request Body	Book_Lists	-
Header	Authorization	User authentication

- Response

[Table 9] Table of book list response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Book_Lists	Book objects
Failure Response Body	message	Empty

6.3.4.2. Get Book Information

- Request

[Table 10] Table of book information request

Attribute	Detail	
Method	GET	
URI	/api/book/{id}	
Parameter	Basic book information (id) of selected one	
Request Body	Book_Title	Title of book
	Book_Author	Author of book
	Book_Content	Content of book
Header	Authorization	User authentication

- Response

[Table 11] Table of book information response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Book_Info	Selected book's information such as title, author and content
Failure Response Body	message	Empty

6.3.4.3. Bookshelf in Classroom

- Request

[Table 12] Table of reference book information request

Attribute	Detail
Method	GET

Attribute	Detail	
URI	/api/class/book	
Request Body	Book_Lists	-
Header	Authorization	User authentication

- Response

[Table 13] Table of reference book information response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Book_Lists	Book objects
Failure Response Body	message	Empty

6.3.4.4. Reference books in Classroom

All reference book used in classroom would be in library, therefore book information in bookshelf in classroom is from the same data table (in similar format) with **6.3.3.2**. Additionally, there can be registration of reference book in classroom.

- Request

[Table 14] Table of add reference book request

Attribute	Detail	
Method	POST	
URI	/api/class/book	
Request Body	Book_Title	Title of reference book
	Book_Author	Author of reference book
Header	Authorization	User authentication

- Response

[Table 15] Table of add reference book response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	message	Success message
Failure Response Body	message	Fail message

6.3.5. Posting space

6.3.5.1. Read(All)

- Request

[Table 16] Table of posting list request

Attribute	Detail	
Method	GET	
URI	/api/posting	
Parameter	-	-
Header	Authorization	User authentication
Request Body	-	

- Response

[Table 17] Table of posting list response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Posting_Lists	List objects

Attribute	Detail	
Failure Response Body	message	Empty

6.3.5.2. Read(Detail)

- Request

[Table 18] Table of detailed posting list request

Attribute	Detail	
Method	GET	
URI	/api/posting/{id}	
Parameter	Search_Posting	-
Header	Authorization	User authentication
Request Body	-	

- Response

[Table 19] Table of detailed posting list response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Posting_Information	Information of posting objects
Failure Response Body	message	Empty

6.3.5.3. Create

- Request

[Table 20] Table of create posting request

Attribute	Detail

Attribute	Detail	
Method	POST	
URI	/api/posting/{id}	
Parameter	Search_Posting	Posting object
Header	Authorization	User authentication
Request Body	Post_id	Id of posting
	Content	Contents of posting
	Title	Title of posting

- Response

[Table 21] Table of create posting response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	message	Success message
Failure Response Body	message	Fail message

6.3.5.4. Delete

- Request

[Table 22] Table of delete posting request

Attribute	Detail	
Method	DELETE	
URI	/api/posting/{id}	
Parameter	Query_id	Posting object that user wants to delete
Header	Authorization	User authentication

- Response

[Table 23] Table of delete posting response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	message	Success message
Failure Response Body	message	Fail message

6.3.5.5. Update

- Request

[Table 24] Table of update posting request

Attribute	Detail	
Method	PUT	
URI	/api/posting/{id}	
Request Body	Title	Title of posting (Information to change)
	Content	Content of posting (Information to change)
Header	Authorization	User authentication

- Response

[Table 25] Table of update posting response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	message	Success message

Attribute	Detail	
Failure Response Body	message	Fail message

6.3.6. Exhibition Hall

6.3.6.1. Select exhibition

- Request

[Table 26] Table of read exhibition list request

Attribute	Detail	
Method	GET	
URI	/api/exhibition/	
Parameter	-	-
Header	Authorization	User authentication
Request Body	-	

- Response

[Table 27] Table of read exhibition list response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Exhibition_Lists	List objects
Failure Response Body	message	Empty

6.3.6.2. Enter exhibition and enjoy

- Request

[Table 28] Table of enter exhibition request

Attribute	Detail	
Method	GET	
URI	/api/exhibition/{exhibition name}	
Request Body	Search_Exhibition	Exhibition object
Header	Authorization	User authentication

- Response

[Table 29] Table of enter exhibition response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	Exhibition_Information	Information of posting objects
Failure Response Body	message	Empty

6.3.6.3. Create work

- Request

[Table 30] Table of create work request

Attribute	Detail	
Method	POST	
URI	/api/exhibition/{exhibition name}/{id}	
Parameter	Search_Exhibition	Exhibition object
Header	Authorization	User authentication
Request Body	Post_id	Id of work
	Content	Contents of work
	Title	Title of work

- Response

[Table 31] Table of create work response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	message	Success message
Failure Response Body	message	Fail message

6.3.6.4. Delete work

- Request

[Table 32] Table of delete work request

Attribute	Detail	
Method	DELETE	
URI	/api/exhibition/{exhibition name}/{id}	
Request Body	Query_id	Work object that user wants to delete
Header	Authorization	User authentication

- Response

[Table 33] Table of delete work response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	message	Success message
Failure Response Body	message	Fail message

6.3.6.5. Update

- Request

[Table 34] Table of update work request

Attribute	Detail	
Method	PUT	
URI	/api/exhibition/{exhibition name}/{id}	
Request Body	Title	Title of work (Information to change)
	Content	Content of work (Information to change)
Header	Authorization	User authentication

- Response

[Table 35] Table of update work response

Attribute	Detail	
Success Code	200 OK	
Failure Code	HTTP error code = 404 (Not Found)	
Success Response Body	message	Success message
Failure Response Body	message	Fail message

7. Database Design

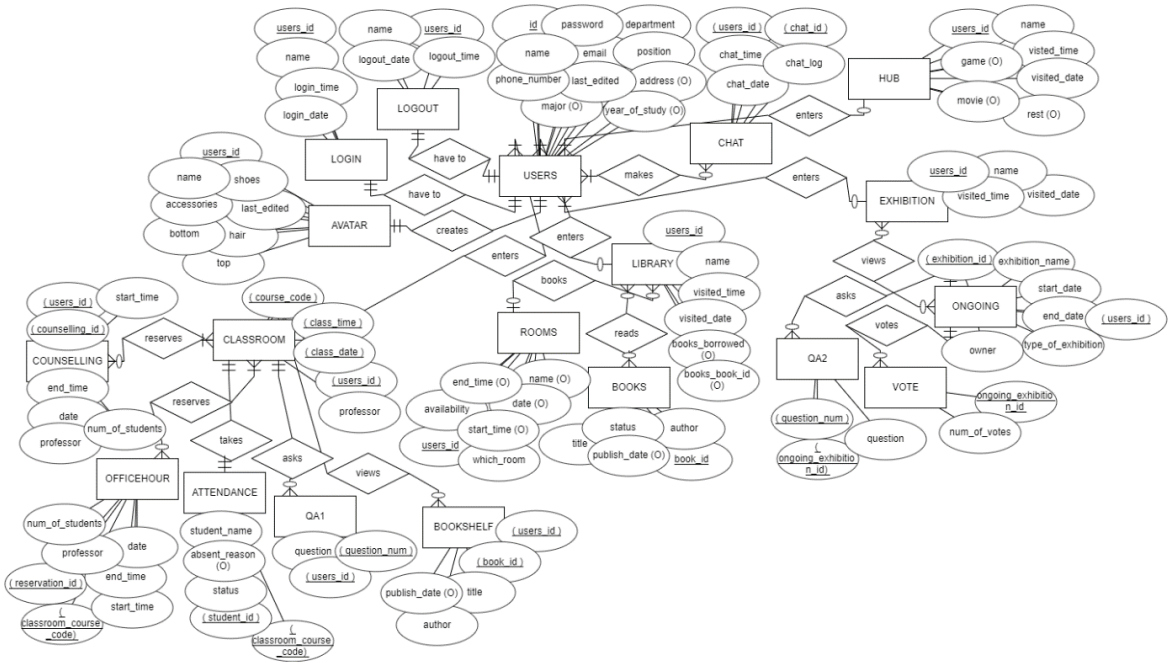
7.1. Objective

This section describes the system data structures and how these are to be represented in a database. It first identifies entities and their relationship through ER-diagram (Entity Relationship diagram). Then, it generates Relational Schema and SQL DDL (Data Description Language) specification.

7.2. ER Diagram

The system consists of five entities; Users, Classroom, Hub, Exhibition Hall, and Library. ER-diagram expresses each entity as rectangular and their relationship as rhombus. When an entity has multiple relationship with another entity, trident (three line) is used to indicate it. When an entity has just one relationship with another entity, the cross (two line) is used to indicate it. When an entity is of optional relationship with another entity, an oval is used to indicate it. The attribute of an entity is expressed as an ellipse. The unique attribute which uniquely identifies an entity is underlined. The composite attributes which uniquely identifies an entity are in brackets.

[Diagram 73] ER Diagram

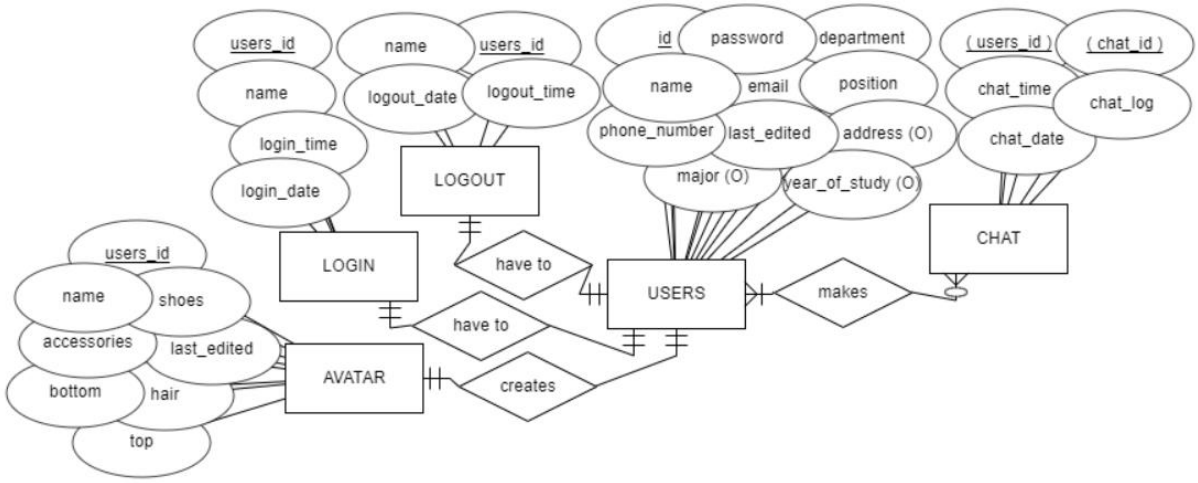


7.2.1. Entities

7.2.1.1. User

USERS and LOGOUT relationship is that all users are only allowed to one logout and it is mandatory to logout. USERS and LOGIN relationship is that all users are only allowed to one login and it is mandatory to login. USERS and AVATAR relationship is that all users are only allowed to one avatar and it is a must to create the avatar. USERS and CHAT relationship is that many users can be in the same chat room and many chat rooms can be used by one user. Also, using the chat room is not mandatory.

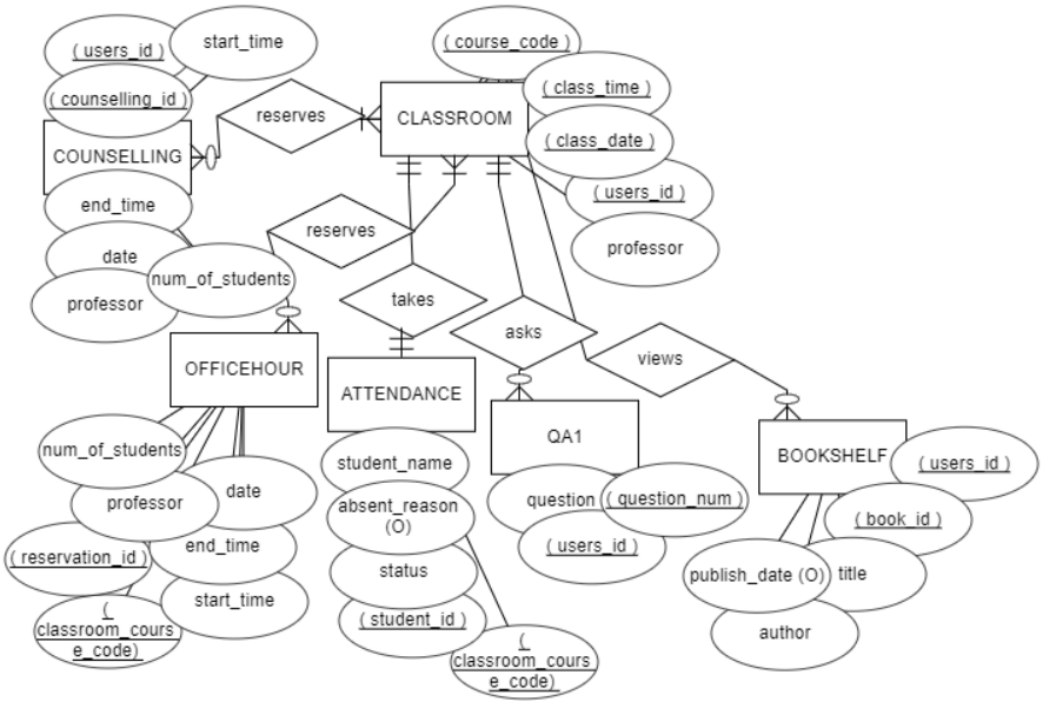
[Diagram 74] ER diagram, Entity, User



7.2.1.2. Classroom

CLASSROOM and COUNSELLING relationship is that a user in the classroom can reserve many counselling sessions and a counselling room can be used by many users but using the counselling room is not a must. CLASSROOM and OFFICEHOUR relationship is that a user in the classroom can reserve many office hour sessions and an office hour session can be used by many users but having an office hour is not a must. CLASSROOM and ATTENDANCE relationship is that all students must take their attendance and all students can only take attendance for him or herself only. CLASSROOM and QA1 relationship is that a user can ask many questions and a question must be asked by at least a user; however, asking a question is not mandatory. CLASSROOM and BOOKSHELF relationship is that a user in the classroom can view many bookshelves and a bookshelf can be viewed by many users but viewing a bookshelf is not a must.

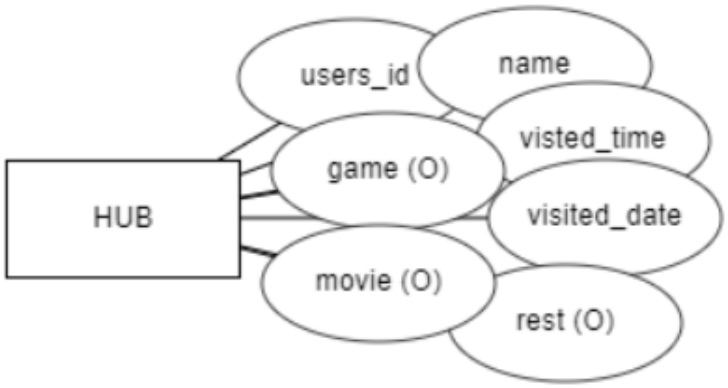
[Diagram 75] ER diagram, Entity, Classroom



7.2.1.3. Hub

The HUB entity does not have any sub-entities but the HUB is connected with the USER entity whereby the hub can be entered by many users but it is not mandatory to enter the hub.

[Diagram 76] ER diagram, Entity, Hub

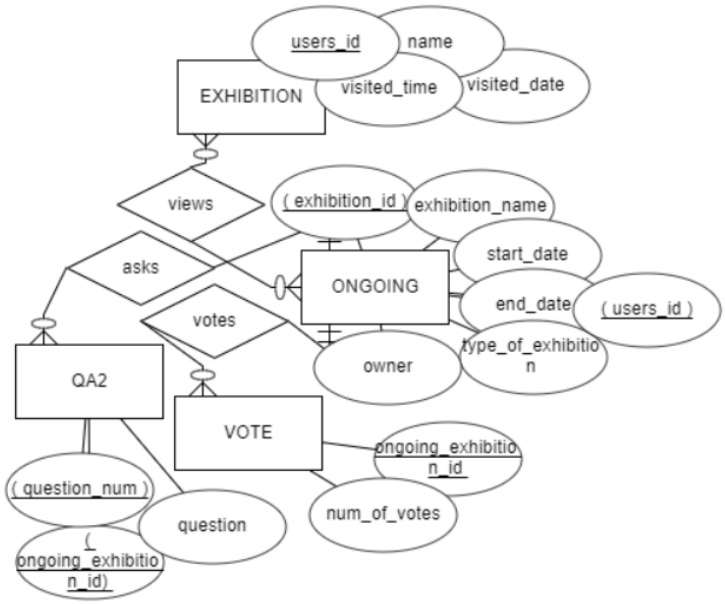


7.2.1.4. Exhibition Hall

EXHIBITION and ONGOING relationship is that a user can view many ongoing exhibitions and an ongoing exhibition can be viewed by many users. Also, it is not a must to view an exhibition and an exhibition may not be viewed by any users too. ONGOING and

QA2 relationship is that a user can ask many questions about an exhibition and an exhibition's question must be asked by at least a user; however, asking a question is not mandatory. ONGOING and VOTE relationship is that a user can vote many exhibitions and an exhibition must be vote by only users; however, voting is not mandatory.

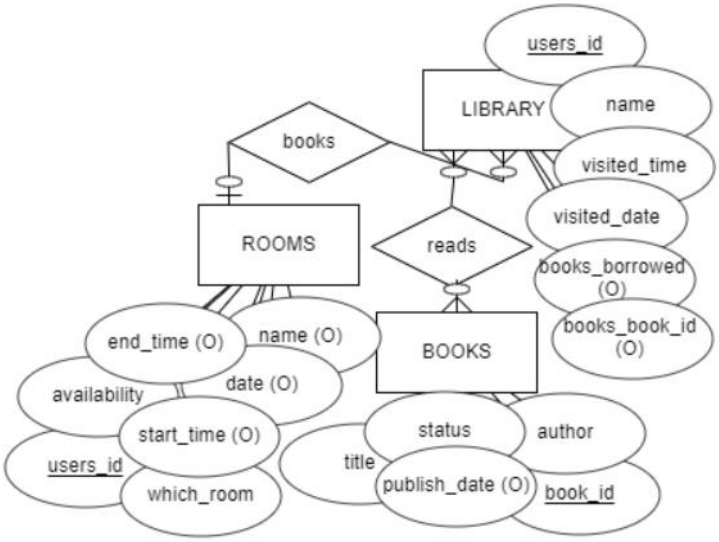
[Diagram 77] ER diagram, Entity, Exhibition Hall



7.2.1.5. Library

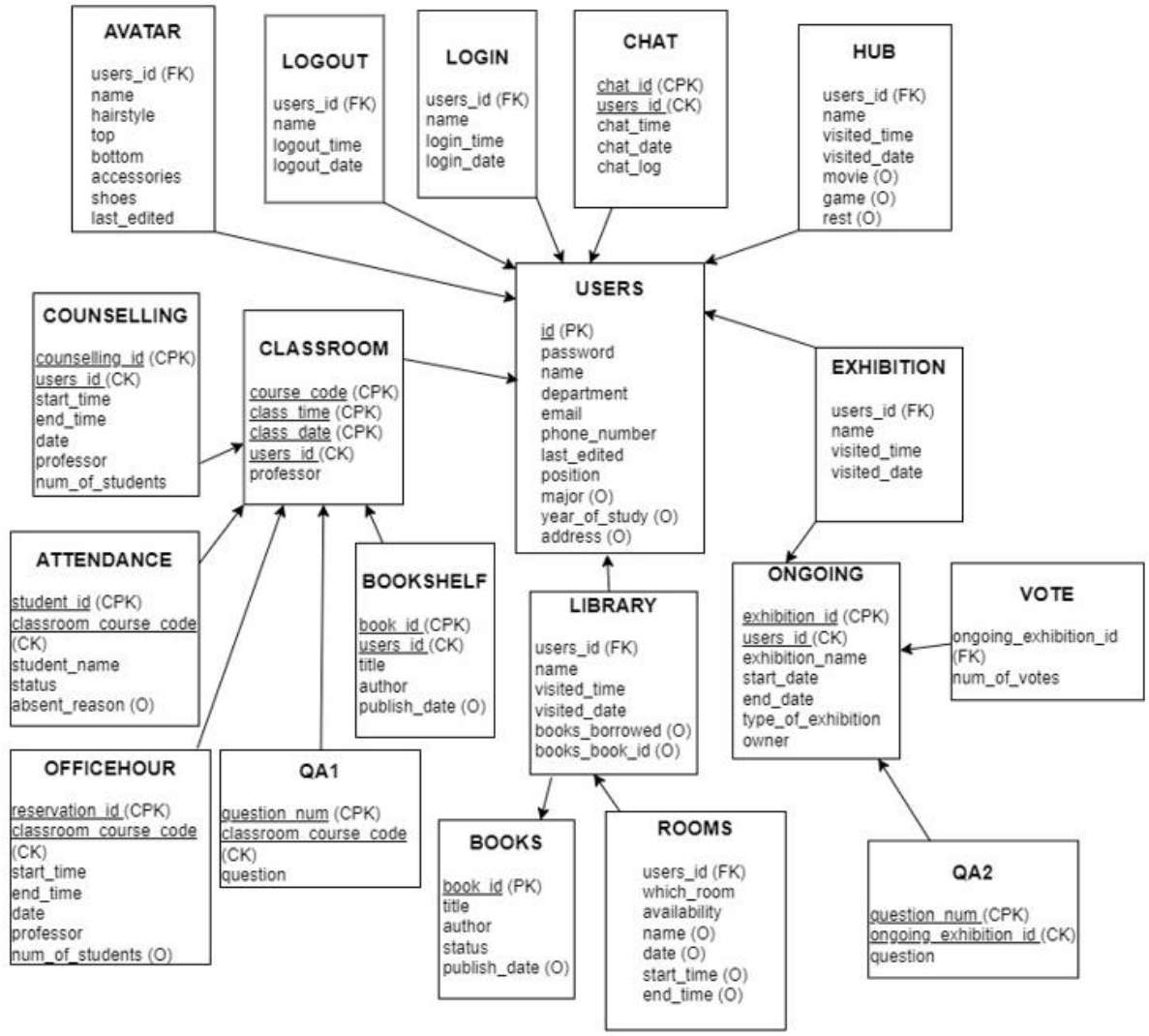
LIBRARY and ROOMS relationship is such that a user in the library can reserve many rooms and a room can only be reserved by a user. Also, it is not a must to reserve a room. BOOKS and LIBRARY relationship is such that a user in the library can read many books and a book can be shared and read by many users; however, it is optional to read a book when entering the library.

[Diagram 78] ER diagram, Entity, Library



7.2.2. Relational Schema

[Figure 79] Relational Schema



7.2.3. SQL DDL

7.2.3.1. Users

```
CREATE TABLE USERS (
    id INT NOT NULL,
    password VARCHAR(15) NOT NULL,
    name VARCHAR(100) NOT NULL,
    department VARCHAR(20) NOT NULL,
    email VARCHAR(20) NOT NULL,
```

```
    phone_number  INT NOT NULL,
    last_edited DATE NOT NULL,
    position  VARCHAR(10) NOT NULL,
    major VARCHAR(20),
    year_of_study INT,
    address VARCHAR(100),
    PRIMARY KEY (id)
);
```

7.2.3.2. Avatar

```
CREATE TABLE AVATAR (
    users_id INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    hairstyle  VARCHAR(100) NOT NULL,
    top VARCHAR(100) NOT NULL,
    bottom  VARCHAR(100) NOT NULL,
    accessories VARCHAR(100) NOT NULL,
    shoes VARCHAR(100) NOT NULL,
    last_edited DATE NOT NULL,
    PRIMARY KEY (users_id),
    FOREIGN KEY (users_id) REFERENCES USERS(id)
);
```

7.2.3.3. Logout

```
CREATE TABLE LOGOUT (
    users_id INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    logout_time  TIME NOT NULL,
    logout_date  DATE NOT NULL,
    PRIMARY KEY (users_id),
```

```
    FOREIGN KEY (users_id) REFERENCES USERS(id)
);
```

7.2.3.4. Login

```
CREATE TABLE LOGIN (
    users_id INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    login_time  TIME NOT NULL,
    login_date  DATE NOT NULL,
    PRIMARY KEY (users_id),
    FOREIGN KEY (users_id) REFERENCES USERS(id)
);
```

7.2.3.5. Chat

```
CREATE TABLE CHAT (
    users_id INT NOT NULL,
    chat_id INT NOT NULL,
    chat_time TIME NOT NULL,
    chat_date DATE NOT NULL,
    chat_log VARCHAR(500) NOT NULL,
    PRIMARY KEY (users_id, chat_id),
    FOREIGN KEY (users_id) REFERENCES USERS(id)
);
```

7.2.3.6. Hub

```
CREATE TABLE HUB (
    users_id INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    visited_time    TIME NOT NULL,
```

```
    visited_date DATE NOT NULL,
    movie BIT,
    game BIT,
    rest BIT,
    PRIMARY KEY (users_id),
    FOREIGN KEY (users_id) REFERENCES USERS(id)
);
```

7.2.3.7. Classroom

```
CREATE TABLE CLASSROOM (
    users_id INT NOT NULL,
    course_code VARCHAR(10) NOT NULL UNIQUE,
    class_time TIME NOT NULL,
    class_date DATE NOT NULL,
    professor VARCHAR(100) NOT NULL,
    PRIMARY KEY (users_id, course_code, class_time, class_date),
    FOREIGN KEY (users_id) REFERENCES USERS(id)
);
```

7.2.3.8. Counseling

```
CREATE TABLE COUNSELING (
    counseling_id INT NOT NULL,
    classroom_course_code VARCHAR(10) NOT NULL,
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    date DATE NOT NULL,
    professor VARCHAR(100) NOT NULL,
```

```
    num_of_students INT NOT NULL,
    PRIMARY KEY (counseling_id, classroom_course_code),
    FOREIGN KEY (classroom_course_code) REFERENCES CLASSROOM(course_code)
);
```

7.2.3.9. Attendance

```
CREATE TABLE ATTENDANCE (
    classroom_course_code VARCHAR(10) NOT NULL,
    student_id INT NOT NULL,
    student_name VARCHAR(100) NOT NULL,
    status BIT NOT NULL,
    absent_reason VARCHAR(50),
    PRIMARY KEY (classroom_course_code, student_id),
    FOREIGN KEY (classroom_course_code) REFERENCES CLASSROOM(course_code)
);
```

7.2.3.10. Bookshelf

```
CREATE TABLE BOOKSHELF (
    book_id VARCHAR(10) NOT NULL,
    classroom_course_code VARCHAR(10) NOT NULL,
    title VARCHAR(50) NOT NULL,
    author VARCHAR(100) NOT NULL,
    publish_date DATE,
    PRIMARY KEY (book_id, classroom_course_code),
    FOREIGN KEY (classroom_course_code) REFERENCES CLASSROOM (course_code)
```

```
);
```

7.2.3.11. Office Hour

```
CREATE TABLE OFFICEHOUR (
    reservation_id INT NOT NULL,
    classroom_course_code VARCHAR(10) NOT NULL,
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    date DATE NOT NULL,
    professor VARCHAR(100) NOT NULL,
    num_of_students INT,
    PRIMARY KEY (reservation_id, classroom_course_code),
    FOREIGN KEY (classroom_course_code) REFERENCES CLASSROOM(course_code)
);
```

7.2.3.12. QA1

```
CREATE TABLE QA1 (
    classroom_course_code VARCHAR(10) NOT NULL,
    question_num INT NOT NULL,
    question VARCHAR(500) NOT NULL,
    PRIMARY KEY (classroom_course_code, question_num),
    FOREIGN KEY (classroom_course_code) REFERENCES CLASSROOM(course_code)
);
```

7.2.3.13. Library

```
CREATE TABLE LIBRARY (
    users_id INT NOT NULL,
```

```
    books_book_id  VARCHAR(10) NOT NULL,
    name VARCHAR(100) NOT NULL,
    visited_time  TIME NOT NULL,
    visited_date  DATE NOT NULL,
    books_borrowed  VARCHAR(50),
    PRIMARY KEY (users_id, books_book_id),
    FOREIGN KEY (users_id) REFERENCES USERS(id),
    FOREIGN KEY (books_book_id) REFERENCES BOOKS(book_id)
);
```

7.2.3.14. Books

```
CREATE TABLE BOOKS (
    book_id  VARCHAR(10) NOT NULL,
    title  VARCHAR(50) NOT NULL,
    author  VARCHAR(100) NOT NULL,
    status BIT NOT NULL,
    publish_date  DATE,
    PRIMARY KEY (book_id)
);
```

7.2.3.15. Rooms

```
CREATE TABLE ROOMS (
    users_library_id  INT NOT NULL,
    which_room  VARCHAR(50) NOT NULL,
    availability  BIT NOT NULL,
    name VARCHAR(100),
    date DATE,
    start_time TIME,
    end_time  TIME,
```

```
    PRIMARY KEY (users_library_id),
    FOREIGN KEY (users_library_id) REFERENCES LIBRARY(users_id)
);
```

7.2.3.16. Exhibition

```
CREATE TABLE EXHIBITION (
    users_id INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    visited_time TIME NOT NULL,
    visited_date DATE NOT NULL,
    PRIMARY KEY (users_id),
    FOREIGN KEY (users_id) REFERENCES USERS(id)
);
```

7.2.3.17. Ongoing

```
CREATE TABLE ONGOING (
    exhibition_id VARCHAR(10) NOT NULL,
    exhibition_name VARCHAR(50) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    type_of_exhibition VARCHAR(20) NOT NULL,
    owner VARCHAR(100) NOT NULL,
    PRIMARY KEY (exhibition_id)
);
```

7.2.3.18. Vote

```
CREATE TABLE VOTE (
    ongoing_exhibition_id VARCHAR(10) NOT NULL,
    num_of_votes INT NOT NULL,
```

```
    PRIMARY KEY (ongoing_exhibition_id),

    FOREIGN KEY (ongoing_exhibition_id) REFERENCES ONGOING(exhibition_id)

);
```

7.2.3.19. QA2

```
CREATE TABLE QA2 (

    ongoing_exhibition_id VARCHAR(10) NOT NULL,

    question_num  INT NOT NULL,

    question VARCHAR(500) NOT NULL,

    PRIMARY KEY (ongoing_exhibition_id, question_num),

    FOREIGN KEY (ongoing_exhibition_id) REFERENCES ONGOING(exhibition_id)

);
```

8. Testing Plan

8.1. Objective

The purpose of testing's planning and planning is to find defects and errors that may appear during or after system development, correct them, and evaluate whether the system has been developed according to the planned intention or its performance is appropriate. In addition, the pre-planned testing plan will provide some guidelines for development. The testing plan will be largely carried out in three stages: development testing, release testing, and user testing. These tests are very important in that they detect potential errors and defects in the product, have perfect operation, and reliably launch the product to the market and customers.

8.2. Testing Policy

8.2.1. Development Testing

Development tests during development can be unstable and components can collide with each other because the software has not been tested sufficiently. Therefore, this step focuses on sub-system evaluation and requires static code an Through this process, we will conduct the evaluation based on performance, reliability, and usability.

8.2.1.1. Performance

As many users access Campus Hub at the same time and enjoy Campus life, it is important to provide services provided by The Campus Hub, such as lectures and books, without delay. In order to provide services that do not require databases such as space movement, lectures, and microphones within 5 seconds, and services that require databases such as book loans and returns and exhibitions within 10 seconds, we will prepare various test cases to measure each reaction speed and evaluate the efficient system communication.

8.2.1.2. Reliability

The system operates safely without failure. Because it integrates various functions such as lecture rooms, hubs, libraries, and exhibition rooms, not only should the components and units of the place in The Campus Hub work correctly, but also should be free from errors

in the process of moving to the place. Therefore, the development test should be conducted from the integration stage to check the failure repeatedly.

8.2.1.3. Usability

The Campus Hub should be easy to use and configured to minimize user errors even for all users, including the first user. The use of technical terms should be minimized, and if necessary, it should be explained easily. Each user should be able to use all system functions instinctively without training time, that is, without a guide page. That is, the user interface should be simple, intuitive, and easy to use. To this end, we would like to select tests of about 10 people in user testing and survey the user experience of the application.

8.2.1.4. Security

Securing information of users and the system is a crucial matter to be handled by developers. Regardless of the value of the information, it should be protected from unwanted visitors to the system.

This system is a community of students from the same university. So, the users should be properly authenticated before using the system. It should make sure that an unauthorized user cannot gains access as system manager and makes system unavailable. Therefore, it is necessary to ensure that the school is authenticated to prevent unrelated users from joining the system. Because vrchat or steam's own membership registration system is used for basic user registration, it takes a reliable step in security. Also, an unauthorized user cannot gain access as system user to confidential information such as user’s personal Information, ID, and password. Only authenticated users can modify profiles, move spaces, and chat. In addition, only users who have registered for the class can participate in the classroom. In addition, not only students but also professors who meet students in classrooms and office hours zones must go through an accurate verification process so that only authorized people can use it. Users who are not reserved are not allowed to enter the study room. Only users who borrowed the book can access the book.

8.2.2. Release Testing

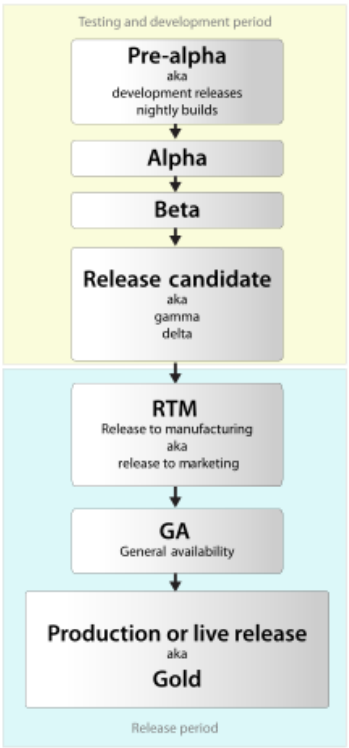
One of the most critical parts of any software development project is to release the

product to the market, and customers.

A technically good software can go wrong due to a wrong way of release. Therefore, release testing is inevitable for better connection between the product and the market. Release testing is testing a new version of a software/application to verify that that the software can be released in a flawless and doesn`t have any issues and works as intended. It should be carried out before release.

Based on Software Release Life Cycle, testing is generally initiated from the 'Alpha' version, where a basic implementation of the software is completed. We would start development testing in Alpha version, and release Beta version for the further testing including user and release testing.

[Figure 80] Software Release Life Cycle



8.2.3. User Testing

This step tests the system in the user's real-world environment. User testing can be explained by the concept of usability testing. We should set up scenario and realistic situation that can proceed necessary user tests. Assume that 20 users are testing The Campus hub.

After setting this situation, we would distribute Beta version of The Campus Hub to them Along with feedback from users, you will hear various reviews such as whether it is similar to the real environment or the unique features of our system. At the same time as the open beta, we will also conduct a close beta to test it from the developer's point of view and find points to fix.

8.2.4. Testing Case

Testing case would be set according to 3 fundamental aspects of the application– function(interaction), performance, and security. We would set several testcases for each part of our services and proceed testing on The Campus Hub with the evaluation sheet. Here is the some examples.

- Login Service
  - Does the Login Service work normally? ex) e-mail format, duplicate, password format
- Set Profile
  - Does it look normal when the user applies Avatar?
  - In case of various errors such as information conflict between users and the format of the information in the profile
- Hub
  - Is moving to another space proceeding normally?
  - Is it possible to match users or import data from functions such as movies and games in the entertainment area without errors?
- Classroom
  - Is it possible for users to enter and leave the classroom according to their timetable?
  - Is it possible to use without errors when using bookshelf?
  - When using the office hour zone, does the function to confirm the reservation with the user's id and check whether it has been processed in advance?
- Library
  - Does the function to check the user's ID when entering the library work properly?
  - Are various functions of the library such as group study, seat reservation, and

study room functioning normally?

- Exhibition hall
  - Since various data are displayed according to the name of the exhibition, is it normally displayed to users?
  - How do the functions that leave user reviews for various works, such as writing support messages, leaving questions, and leaving gifts, work properly?
  - Are the features that users vote for their favorite works and give gifts through various events working properly?

9. Development Plan

9.1. Objective

This chapter illustrates the environments programs, and plugins for the Unity which we will use for the development. Main development will be on Unity and other programs will be used to support the development on Unity.

9.2. Frontend Environment

9.2.1. Photoshop

Photoshop is a raster graphics editor developed and published by Adobe Inc. for Windows and macOS. Photoshop can edit and compose raster images in multiple layers and supports masks, alpha compositing and several color models including RGB, CMYK, CIELAB, spot color, and duotone. This program could provide texture adjustment or creation during our project. We will use multiple 3d models in unity, and we should edit the textures

[Figure 81] Adobe Photoshop logo

9.2.2. Blender

Blender is a free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, motion graphics, interactive 3D applications, virtual reality, and computer games. Blender's features include 3D modeling, UV unwrapping, texturing, raster graphics editing, rigging and skinning, fluid and smoke

simulation, particle simulation, soft body simulation, sculpting, animating, match moving, rendering, motion graphics, video editing, and compositing. We will likely to use 3D model editing and UV unwrapping and rigging feature. Blender is open-source free program so we chose this to be used.

[Figure 82] Blender logo

9.2.3. Unity2019

Unity is a cross-platform game engine developed by Unity Technologies. The engine supports a variety of desktop, mobile, console and virtual reality platforms. It is popular for iOS and Android mobile game development and used for many games. VRChat currently support Unity's development environment. We will use Unity 2019.4.30f to develop our project.

[Figure 83] Unity2019 logo

9.2.4. UdonSharp

UdonSharp is a compiler that compiles C# to Udon assembly which is a programming language built by VRChat development team. We could develop VRChat's world by built in Udon Graph program which is made by VRChat development team, but there are many restrictions. Udon Graph is for the users who are not familiar at the programming languages so it is very inefficient to programmers like our team. UdonSharp is far better way to develop VRChat world for the programmers. So we chose this plugin to develop our project.

[Figure 84] UdonSharp logo

9.3. Backend Environment

9.3.1. Github

It provides hosting for software development version control using Git. It enables teammates to develop a single project together, resulting in easy integration of components. We are now using GitHub for developing The Campus Hub and controlling version of it.

[Figure 85] Github logo

9.3.2. MySQL

MySQL is the world's most popular open source relational database management system (RDBMS). It is a multi-threaded, multi-user structured query language database management system, managed and supported by Oracle. In this system, MySQL will be used as the RDBMS that manages the database of the backend application. This is because MySQL has been used by many people for a long time, so there are various sources, and team members are more familiar with MySQL.

[Figure 86] MySQL logo

9.3.3. Youtube

YouTube is a video sharing platform provided by Google. As the world's largest video sharing and hosting site, users can watch, upload, and share videos. Since the types of video players that can be used within the VR chat are limited, we decided to access the necessary link or url through youtube, which is cost-free and easy to use.

[Figure 87] Youtube logo

9.3.4. Django

Django is a Python-based open-source web framework that follows the Model-View-Controller (MVC) pattern. The advantage of Djagno is that the components are highly reusable and can be developed quickly, because there are various functions that support easy solving of various problems such as database management and security problems. Django is currently used as a web framework in various places such as Instagram, Disqus, and Mozilla, and Django was used when writing the Backend Application in this system.

[Figure 88] Django logo

9.4. Schedule

[Table 36] Schedule

Phase	Expected	Actual
Concept & Proposal	21.09.26	21.09.26
Requirements Specification	21.10.24	21.10.24
Design Documentation	21.11.21	21.11.21
Environment Setting	21.11.12	21.11.12
Frontend Development	~21.11.28	-
Backend –	~21.11.28	-
System Integration and Testing	~21.11.30	-

10. Index

10.1. Table Index

10.2. Figure Index