Metaverse GLS School Bag System

Software Design Specification

2021.11.21

Introduction to Software Engineering 41

TEAM 1

Team Leader	Seowang Park
Team Member	Hyunseong Na
Team Member	Hanbeul Kim
Team Member	Seonye Lee
Team Member	Hyunjeong Lee
Team Member	Yebin Park
Team Member	Hwanseung Chang

CONTENTS

LIST OF FIGURES

1. Foreword

Based on prior requirement specifications, this document holds the information about the design of the system and the diagrams. Due to unexpected walls caused by the platform limitation, we had to make some adjustments. This chapter will tell a brief structure of this document and assumptions based underneath this project and its system.

1.1. Document Brief Description

- 1. Foreword : Abstract of this document
- 2. Introduction : Summary of project purpose and tools used in this document
- 3. System Architecture : Overall information of the system architecture with diagrams
- 4. System Architecture- Front End : More detail explanation of front end architecture
- 5. Testing Plan : Testing environment and requirements
- 6. Development Plan : Tools and method for upcoming development
- 7. Supporting Information : Format and history of this document

1.2. Scope and Assumptions

This document is assumed to be read by the ones which read the requirements specification of this project. The product of this project is meant for SKKU officials such as academic staff and faculties, students etc. The system is based on VRchat platform and assumed to be used with the school database.

1.3. Objectives

The purpose of this document is to inform the stakeholders how the system works with diagrams and test cases. It also provides a brief structure of the metaverse GLS school bag system. With architectural overview, the stakeholders of this project will have much specific knowledge and sight on the system and what can be done to modify the system if needed. The software architecture and software design shown in this document will show technical approach and implementation on the test environment. Lastly, this document has some changes from the previous document, requirements specification. All the changes will be informed while following the document.

2. Introduction

The project aims to implement a course registration system implemented within VRchat, suitable for the metaverse era. It aims to create a metaverse map where you can put subjects in a backpack, a place to discuss subjects with friends, and view videos and check explanations of subjects. It can be used as a component of the non-face-to-face lecture registration system in the Corona era and it is a project suitable for the metaverse era.

2.1. Objectives

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

2.2. Applied Diagram

2.2.1. UML

The Unified Modeling Language (UML) is a language used in the field of software engineering that represents the components of the Object-Oriented Programming concepts. It is the general way to define the whole software architecture or structure. In Object-Oriented Programming, we solve and interact with complex algorithms by considering themselves as objects or entities. These objects can be anything. It can be the bank or a bank manager too. The object can be a vehicle, animal, machine, etc. The thing is how we interact and manipulate them that they can perform tasks and they should. The tasks can be interacting with other objects, transferring data from one object to another, manipulating other objects, etc. The single software could have hundreds or even thousands of objects. So, UML provides us a way to represent and track those objects in a diagram to become a blueprint of our software architecture.

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 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.

2.2.4 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.

2.3. Project Scope

The era of metaverse has already come to the extent that Facebook, a giant IT company, has recently changed its name to Meta. In this untact environment, various metaverse platforms are attracting attention, and VRChat, one of the leading platforms currently leading this market, is going to introduce a non-face-to-face course registration system.

2.4. References

- Team 1, 2020 Spring. Software Design Document, SKKU.
- https://www.lucidchart.com/pages/uml-use-case-diagram
- VRChat : https://ko.wikipedia.org/wiki/VRChathttps://twitter.com/vrchat
- Unity :
https://ko.wikipedia.org/wiki/%EC%9C%A0%EB%8B%88%ED%8B%B0_(%EA%B2%8C%EC%9E%84_%EC%97%94%EC%A7%84)
- Software release life cycle : Reference site :
https://ko.wikipedia.org/wiki/%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4_%EB%B0%B0%ED%8F%AC_%EC%83%9D%EB%AA%85_%EC%A3%BC%EA%B8%B0

3. System Architecture and Diagram

3.1. Objectives

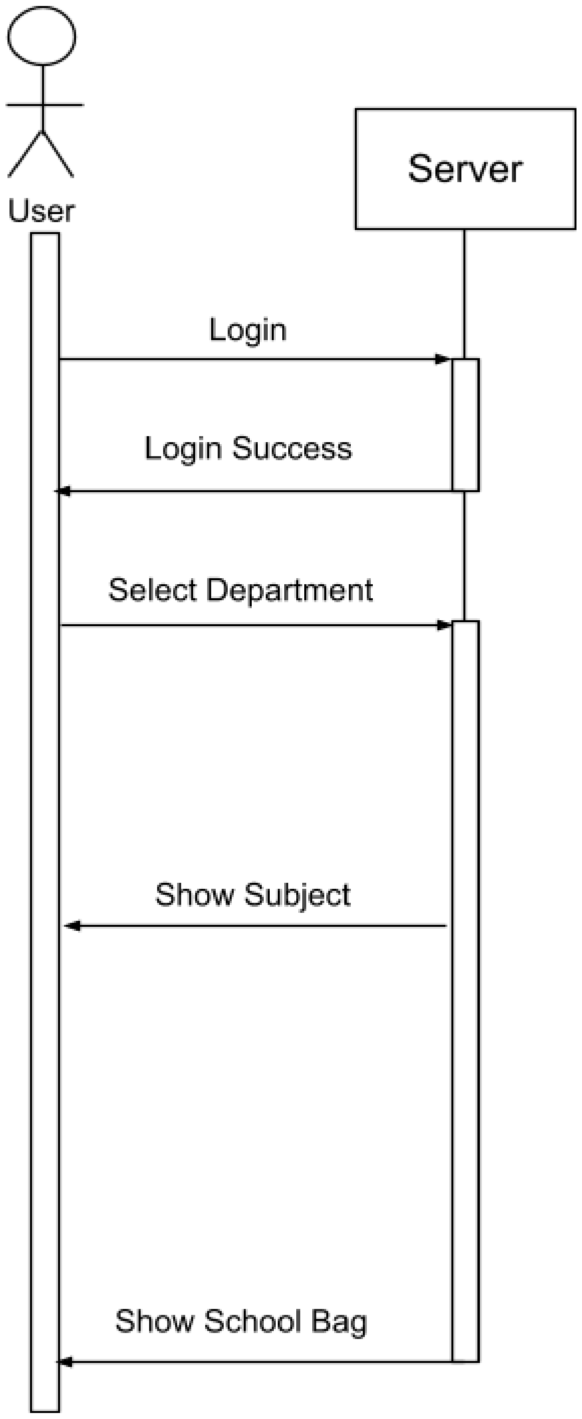
Here in this chapter, we describe and show the organization of the system, ranging from the front-end design to the back-end design of the application for the project.

3.2. System Organization

Basically, it is distributed in the form of a map that runs on VRChat. Users can enter the map, choose a subject, and put it in their backpack. It will be provided in 3D form, so it can be easily checked and operated intuitively. You can also view example videos or course descriptions in the subject object. Classification of departments is done by using rooms that exist in the map. You can also chat with other users. Assuming that the main host keeps the map open, what the user wrote on the map remains on the map and can be checked later.
Since VRChat does not provide network support by default at this time, it was decided that it is impossible to link with an external DB. Therefore, we plan to process everything in the map without using the DB.

3.2.1. Sequence Diagram

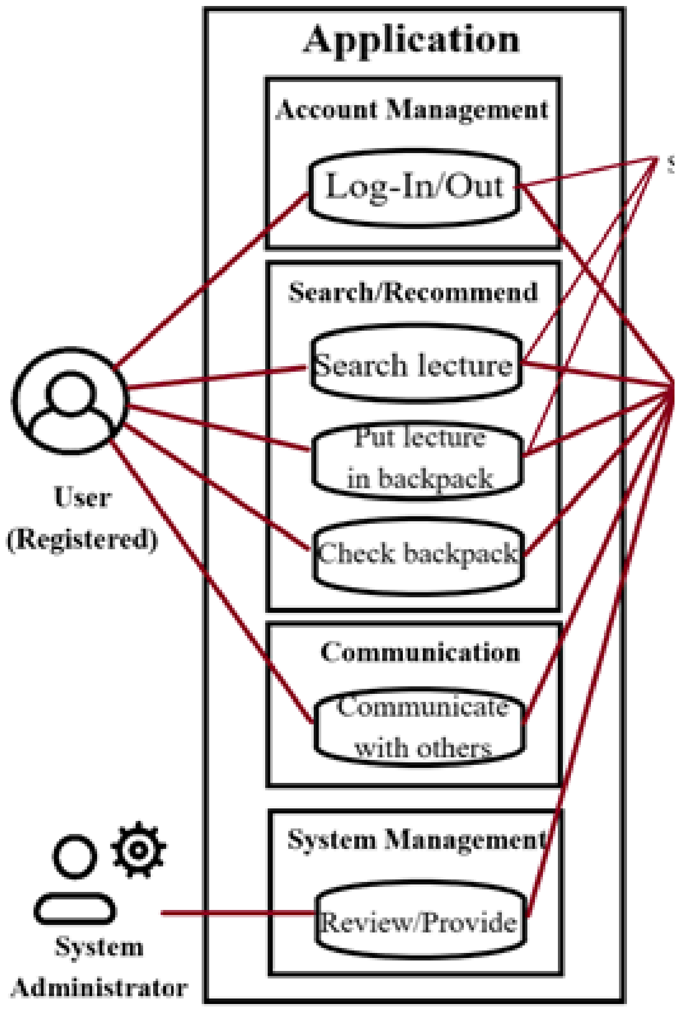
We had some changes here. There is no network system allowed for VRChat. So we can not gain or store data on DB. So No DB system here.



[Figure 1] Sequence diagram

3.2.2. Use Case Diagram

We had some changes here. There is no network system allowed for VRChat. So we can not gain or store data on DB. So No DB system here.



[Figure 2] Use case diagram

4. System Architecture - Frontend

4.1. Objectives

In this chapter, we describe the structure, attributes, methods of the system and explain the relationship between each component.

4.2. Subcomponents

4.2.1. Search

4.2.1.1. Attributes

These are the attributes that a Search object has.

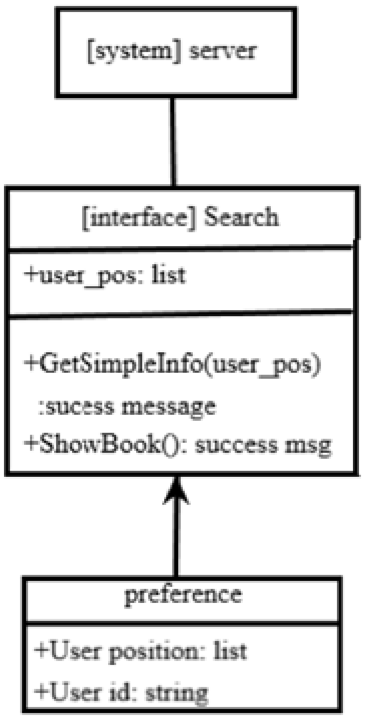
- User id: id of user
- User position: position of user

4.2.1.2. Methods

These are the methods that a Search class has.

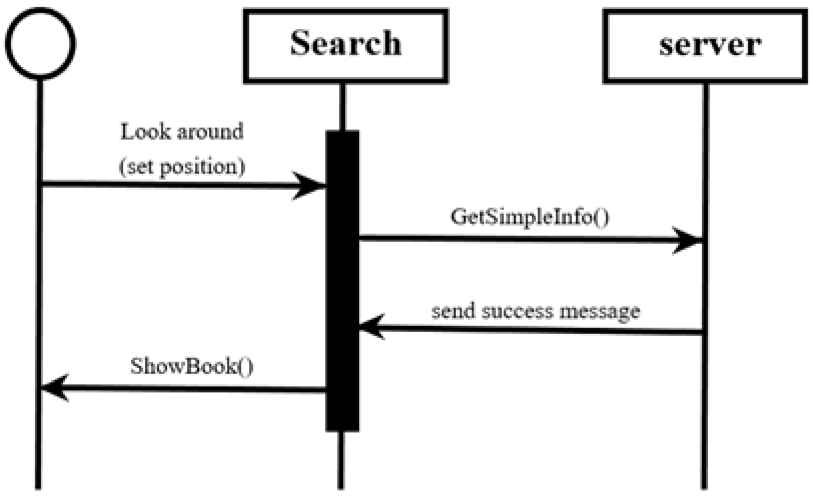
- GetSimpleLectureInfo()
- ShowBooks()

4.2.1.3. Class Diagram



[Figure 3] Class diagram - Search

4.2.1.4. Sequence Diagram



[Figure 4] Sequence diagram -Search

4.2.2. Search Result

4.2.2.1. Attributes

These are the attributes that a Search Result (details of a lecture) object has.

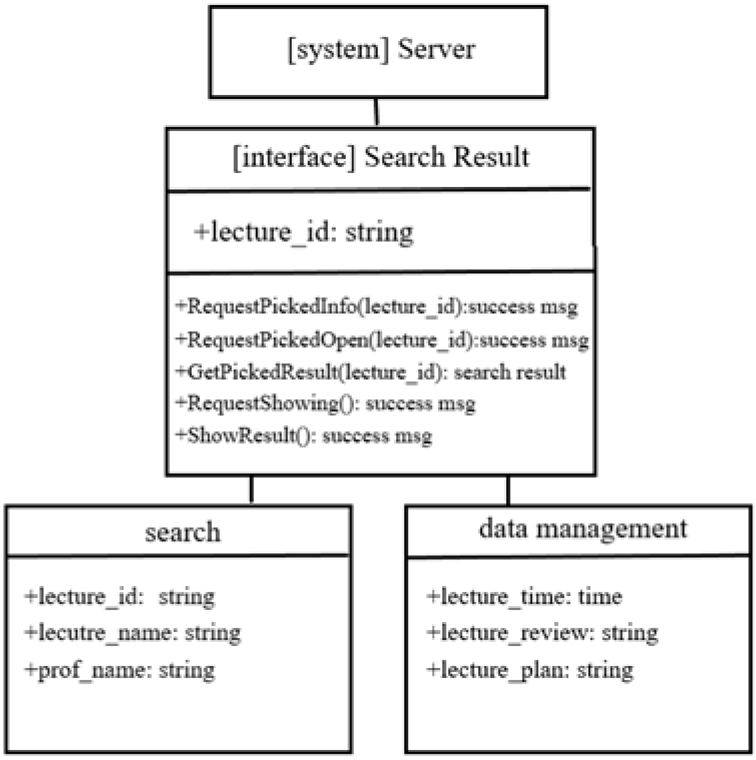
- Lecture id: identifier of lecture (Academic number)
- Lecture name: name of lecture
- Professor name: name of professor

4.2.2.2. Methods

These are the methods that a Search Result class has.

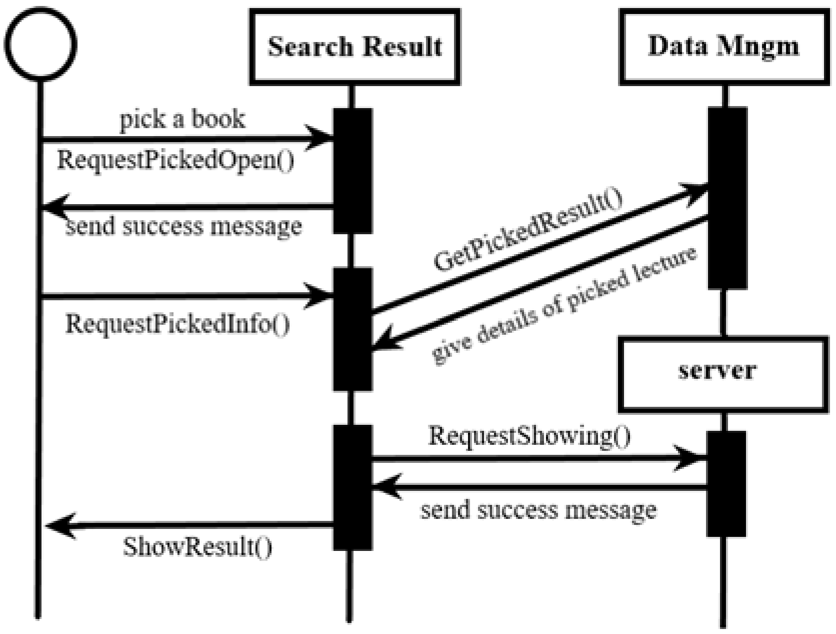
- RequestPickedInfo()
- RequestPickedOpen()
- GetPickedResult()
- RequestShowing()
- ShowResult()

4.2.2.3. Class Diagram


[system] Server
[interface] Search Result
+lecture_id: string
+RequestPickedInfo(lecture_id):success msg
+RequestPickedOpen(lecture_id):success msg
+GetPickedResult(lecture_id): search result
+RequestShowing(): success msg
+ShowResult(): success msg
search
+lecture_id: string
+lecutre_name: string
+prof_name: string
data management
+lecture_time: time
+lecture_review: string
+lecture_plan: string

[Figure 5] Class diagram - Search Result

4.2.2.4. Sequence Diagram



[Figure 6] Sequence diagram - Search Result

4.2.3. Put In

4.2.3.1. Attributes

These are the attributes that a Put In object has.

● Lecture id: identifier of class (Academic number)

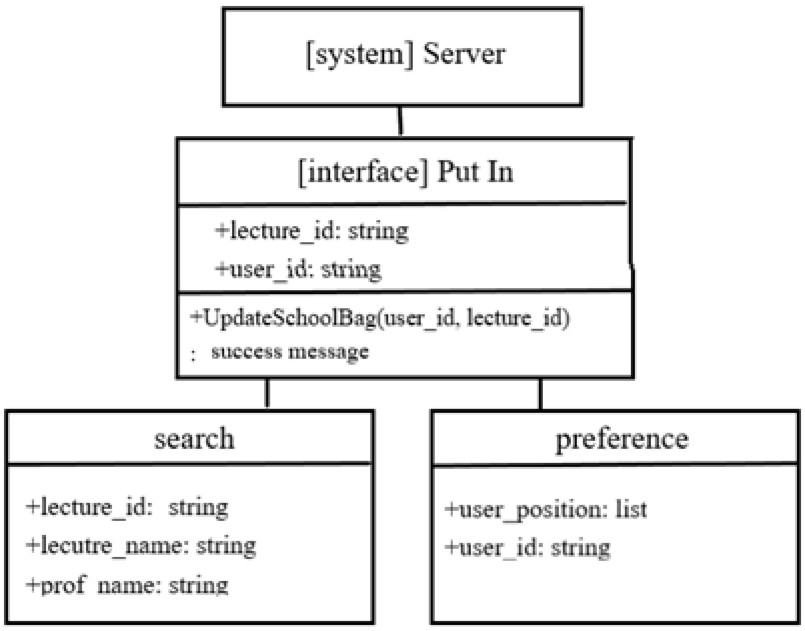
● User id: id of user

4.2.3.2. Methods

This is the method that a Put In class has.

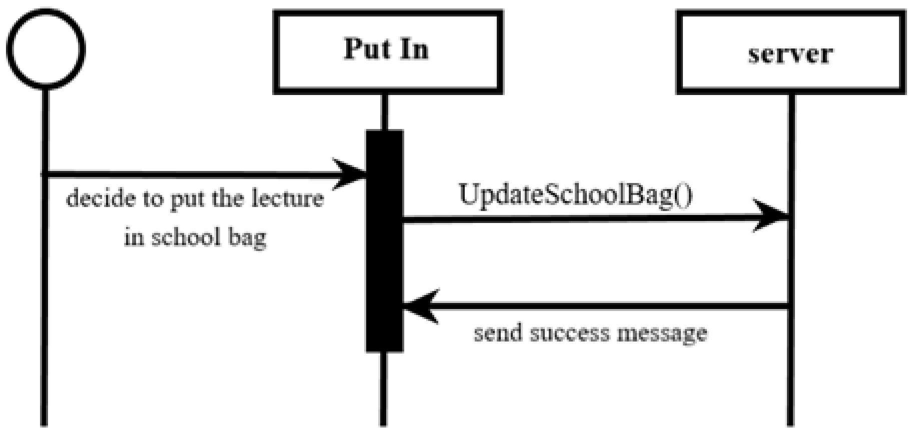
● UpdateSchoolBag()

4.2.3.3. Class Diagram



[Figure 7] Class diagram - Put In

4.2.3.4. Sequence Diagram



[Figure 8] Sequence diagram - Put In

4.2.4. Check School Bag

4.2.4.1. Attributes

These are the attributes that a Check School Bag object has.

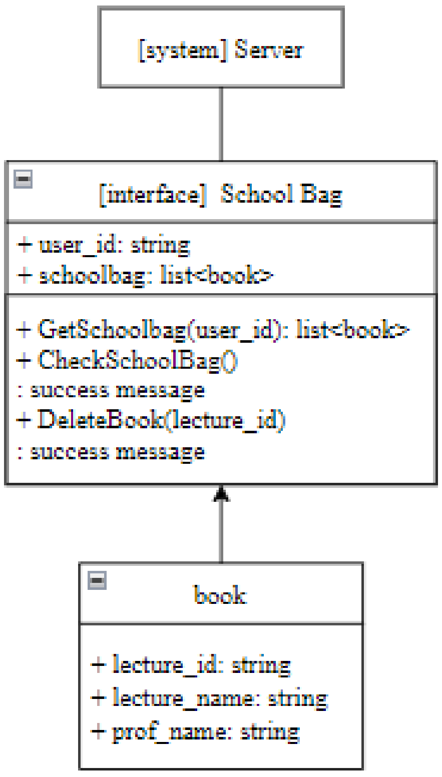
- User id: id of the user
- lecture_id: identifier of lecture

4.2.4.2. Methods

These are the methods that a Check School Bag class has.

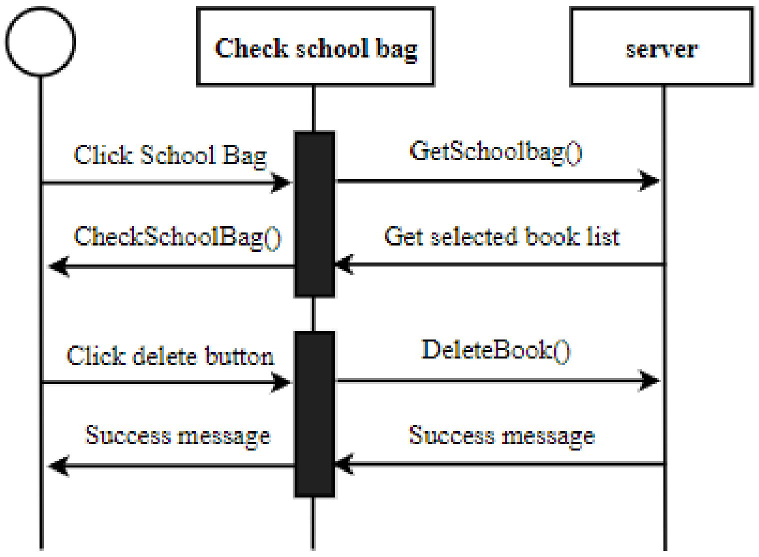
- GetSchoolBag()
- CheckSchoolBag()
- DeleteBook()

4.2.4.3. Class Diagram



[Figure 9] Class diagram - Check School Bag

4.2.4.4. Sequence Diagram



[Figure 10] Sequence diagram - Check School Bag

4.2.5. Communication

4.2.5.1. Attributes

These are the attributes that a Communication object has.

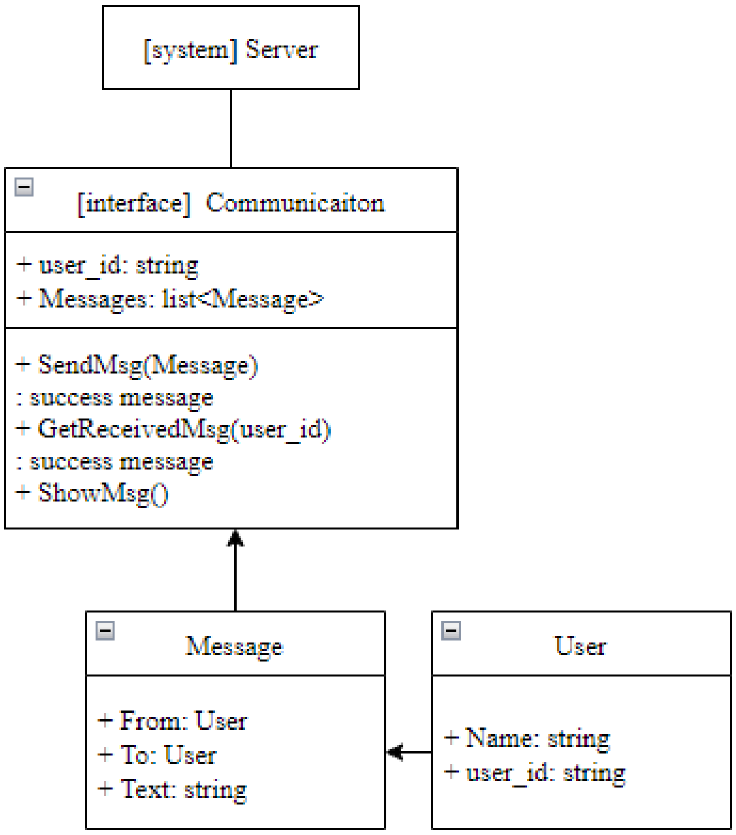
- User id: id of the user
- Messages: list of Message
- Message: includes sender, recipient, and content.

4.2.5.2. Methods

These are the methods that a Communication class has.

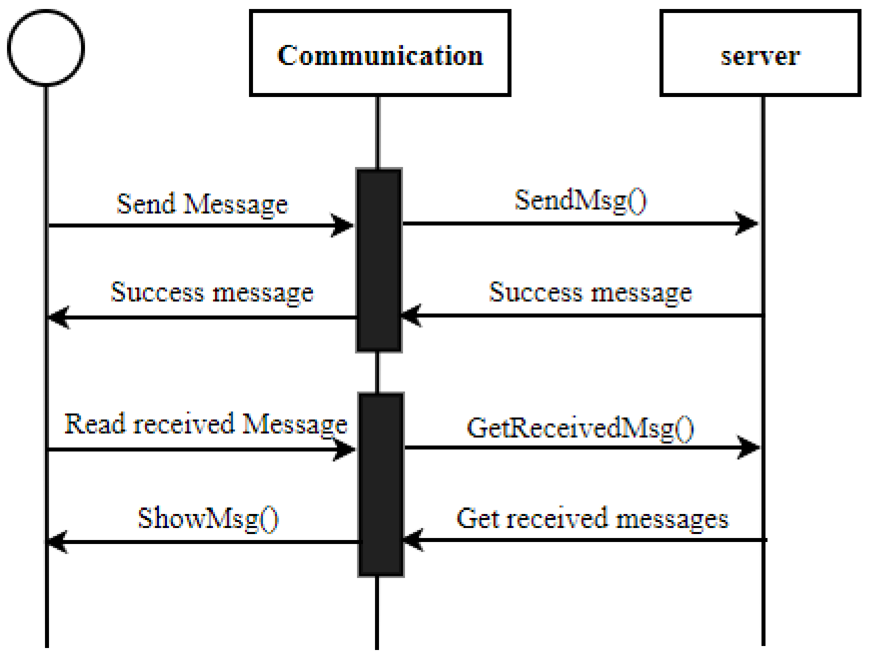
- SendMsg()
- GetReceivedMsg()
- ShowMsg()

4.2.5.3. Class Diagram



[Figure 11] Class diagram - Communication

4.2.5.4. Sequence Diagram



[Figure 12] Sequence diagram - Communication

4.2.6. Change User Info

4.2.6.1. Attributes

These are the attributes that a Change User Info object has.

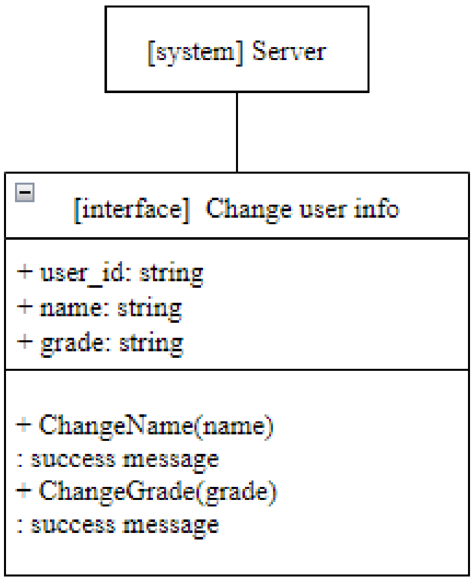
- User id: id of the user
- Nick name : nick name of user
- grade: grade of the user

4.2.6.2. Methods

These are the methods that a Change User Info class has.

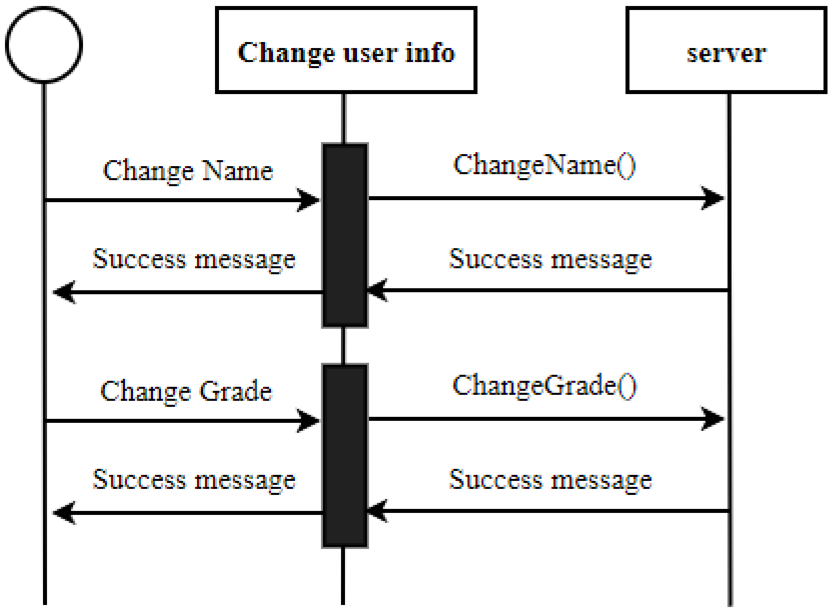
- ChangeName()
- ChagneGrade()

4.2.6.3. Class Diagram



[Figure 13] Class diagram - Change user Info

4.2.6.4. Sequence Diagram



[Figure 14] Sequence diagram - Change user Info

5. Testing Plan

5.1. Objectives

In this chapter, we are going to illustrate how to plan our test. Purpose of the test is to detect potential errors and defects by checking whether the developed system or software is designed for customer requirements and running in the intended direction.

5.2. Testing Policy

5.2.1. Development Testing

Development testing is a method of applying testing practices consistently throughout the software development life cycle process. Through the development testing, developed systems or software could ensure the detection of errors and bugs at the right time which further ensures delay of any kind of risk in terms of time and cost.

In this process, we are mainly focused on performance and reliability.

5.2.1.1. Performance

The most important thing in the GLS school bag system is to show accurate information about the course and put the course in the school bag which they choose immediately. It is crucial for developers to show accurate information about the course and be able to safely put the selected course into a school bag. We would prepare various test cases and situations to see if the information on the course was accurate and safely put the course into a school bag.

5.2.1.2. Reliability

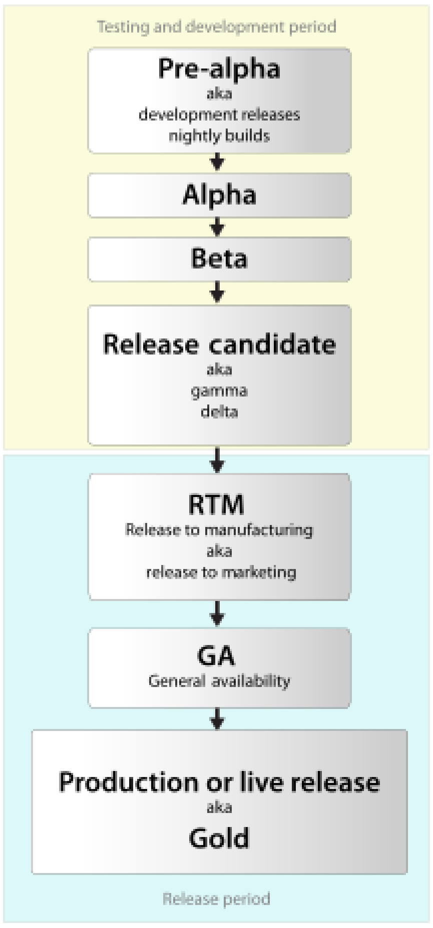
In order for the system to operate safely without failure, the Unity asset package and object comprising the system map should operate correctly without collisions. Therefore, we need to undergo development testing from the map developing stage and check failure iteratively while each Unity asset package and object are added to construct the map.

5.2.2. Release Testing

Release testing is the process that involves testing a particular release of a system. Before the system is going to be released on the market, the most important thing is to proceed with the release test to convince the customer of the system that it is good enough for use.

Therefore, release testing has to show that the system delivers its specified functionality, performance and dependability, and that it does not fail during normal use.

Based on the Software Release Life Cycle, our system testing starts from the Alpha version. Alpha version is the first step of the software test and the basic implementation of the software should be completed at this step. After the Alpha version, we release the next Beta version. From the Beta version, we aim to receive opinions from many users, reflect them into the system, and complete the official version better.



[Figure 15] Software Release Life Cycle

5.2.3. User Testing

For the most accurate evaluation of user testing, user test is executed under possible scenarios and realistic situations. We assumed that 20 users accessed and used the system at the same time. First, let users use various functions freely and receive user feedback, and then we also get feedback by performing the test case we set.

5.2.4. Testing Case

Testing case was set according to performance and reliability aspects. In various situations, performance focused on course information and school bag functions, and reliability focused on whether there was a collision between multiple Unity asset packages and objects. Each aspect will have 10 test cases.

6. Development Plan

6.1. Objectives

This chapter illustrates the environment which we used when developing our system.

6.2. Environment

6.2.1. VRChat

[Figure 16] VRChat logo

VRChat is a large-scale multi-user online virtual reality(VR) social service. VRChat makes it easy to create and explore virtual reality together. It also provides the transformation of 3D content to social experience.

By using VRChat, 3D maps made of Unity could be transformed into virtual space. It also provides a server that allows users to communicate with each other in virtual space. Since VRChat provides a server, we don't have to care about the server.

6.2.2. Unity

[Figure 17] Unity logo

Unity is a game engine that provides a development environment for 3D and 2D video games and an integrated production tool for creating interactive content such as 3D animation and VR.

By using Unity, we made 3D maps for our system. 3D maps can be created easily using packages provided by Unity and packages developed by Unity users in the asset store.

6.2.3. C#

[Figure 18] C# logo

C# is a object-oriented programming language, Unity basically supports C# programming languages. By using C# in Unity, we could make the object by ourselves.

6.3. Constraints

In implementing our system, it is based on the contents of this document, but it is possible to add new functions and change functions, but the following constraints must be observed.

- Paid packages can be used in the Unity asset store, but they must be selected within an acceptable budget.
- When using a new package in the Unity asset store, package should test whether there is a collisions with other existing package
- System must be developed by VRChat SDK 3, SDK 2 is not compatible with SDK 3.
- Consider the maintenance cost of the system.
- Consider scalability and availability of the system.
- Consider the system should be easier for users to understand and use.

6.4. Assumptions and Dependencies

This document is written on the assumption that the system was implemented based on the Unity version 2019.4.30f1 and VRChat SDK 3. Therefore, in other versions except for the specified one, the system may not function normally.

7. Supporting Information

7.1. Software Design Specification

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

7.2. Documentation History

Date	Description	Writer	Remarks
2021.11.12	Cover page	Hyunseong Na	
2021.11.14	Addition of 6.1 - 6.2	Hanbeul Kim	
2021.11.15	Addition of 6.3 - 6.4	Hanbeul Kim	
2021.11.16	Addition of 5, 7.2	Hanbeul Kim	
2021.11.19	Addition of 4.1 - 4.2.3	Hyunjeong Lee	
2021.11.19	Addition of 1.1 - 1.3	Hyunseong Na	
2021.11.20	Addition of 4.2.4 - 4.2.6	Seonye Lee	
2021.11.20	Addition of 2 - 3	Hwan Seung Chang	
2021.11.21	Review	Seowang Park	