SKKU Flea Market

Software Design Specification

2021.11.21.

Introduction to Software Engineering

2016314474 Seongbin Yoon

2018313475 Ji Young Kim

2019313260 Jihye Park

2017313841 Sanghun Kim

2016310493 Munsuk Jang

2017312097 HaJeong Lim

2016310487 SeHyun Choi

CONTENTS

1. Preface

This chapter contains the readership information, readership, scope, objective of this document and the document structure of this Software Design Document for metaverse flea market service.

1.1. Readership

This Software Design Document is divided into 10 sections with various subsections. The structure of the Software Design Document can be found as listed below, in the Document Structure subsection of this SDD. In this document, Team 10 is the main reader. Additionally, professors, Tas, and team members in the Introduction to Software Engineering class can be the main readers.

1.2. Scope

This Design Specification is to be used by Software Engineering and Software Quality Engineering as a definition of the design to be used to implement the Flea market Service on VRChat platform.

1.3. Objective

The primary purpose of this Software Design Document is to provide a description of the technical design aspects for the flea market service. This document describes the architecture and design decisions for the implementation of ‘World’ for flea market service in VRChat platform. It also provides an architectural overview of the system to depict different aspects of the system. It further specifies the structure and design of some of the modules discussed in the SRS document and in addition, displays some of the use cases that have been transformed into sequential and activity diagrams, including the class diagrams which show how the programming team would implement the specific module. The intended audience of this document is, but not limited to, the stakeholders, developers, designers, and software testers of the flea market service world on VRChat.

1.4. Document Structure

(1) Preface : this chapter describes readership, scope of this document, object of this system, and structure of this document.

(2) Introduction : this chapter describes several tools used for this document, several diagrams used in this document and the references, and object of this project.

(3) System Architecture – Overall : this chapter describes the overall structures of the design. It shows system organization, context diagram, sequence diagram, and use case diagram.

(4) System Architecture – Frontend : this chapter describes the structure of the front-end system including functions, structure, and execution.

(5) System Architecture – Backend : this chapter describes the structure of the back-end system include DB implemented using Udon objects in the VRChat world.

(6) Design : this chapter describes the overall structures which are used for the protocols between each subsystem, especially SKKU Flea Market system and the Udon.

(7) Entities : this chapter describes how the entities and variables works for the features in the system.

(8) Testing Plan : this chapter describes tests to evaluate and identify errors or missing requirements in contrast to actual requirements.

(9) Development Plan : this chapter describes technologies, environments(tools) for development of the system.

2. Introduction

2.1. Objectives

In this chapter, we describe the diagrams which we have applied to this project in the design phase, the scope of this project, and the references we used.

2.2. Applied Diagrams

2.2.1. UML

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

2.2.2. Data Flow Diagram

A data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

2.2.3. Use Case Diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

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.

2.2.5. Sequence Diagram

A sequence diagram or system sequence diagram (SSD) shows object interactions arranged in time sequence in the field of software engineering. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

2.3 Project Scope

The system is designed to hold an actual flea market in a metaverse space, encourage student users to socialize with each other, actively trade used items, and enjoy the metaverse campus life. The system is based on the metaverse platform ‘VRChat’ application which is windows application providing online VR environment. The system is not a program, nor including the front-end or back-end application, but the ‘World’ in VRChat which means the new service user can experience in VRChat. The system is developed by using VRchat SDK and Udon which are a Software Development Kit and a programming language provided by the VRChat Development Team.

2.4 References

Team 1. “SDD_TEAM1.docx”. SKKU, Last Modified: May. 24, 2020.
https://github.com/skkuse/2020spring_41class_team1/tree/master/docs/SDD_TEAM1.pdf

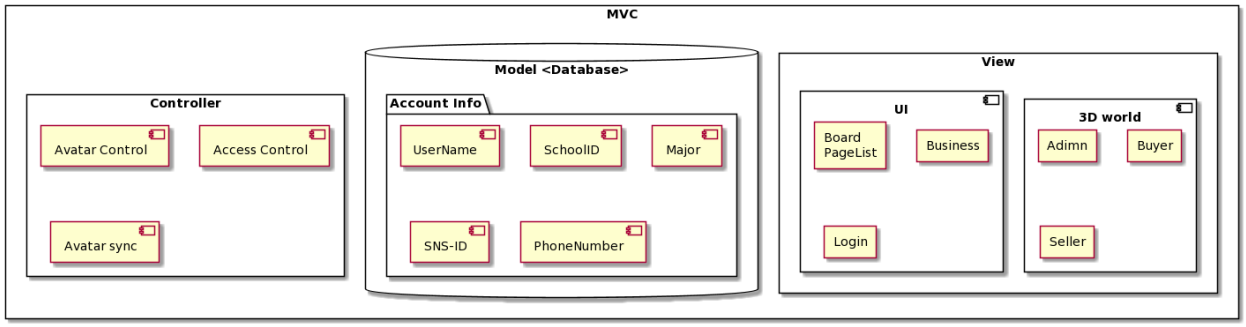
IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, In IEEEXplore Digital Library. https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc

3. System Architecture - Overall

This chapter describes the overall structures of the design. It shows system organization, context diagram, sequence diagram, and use case diagram.

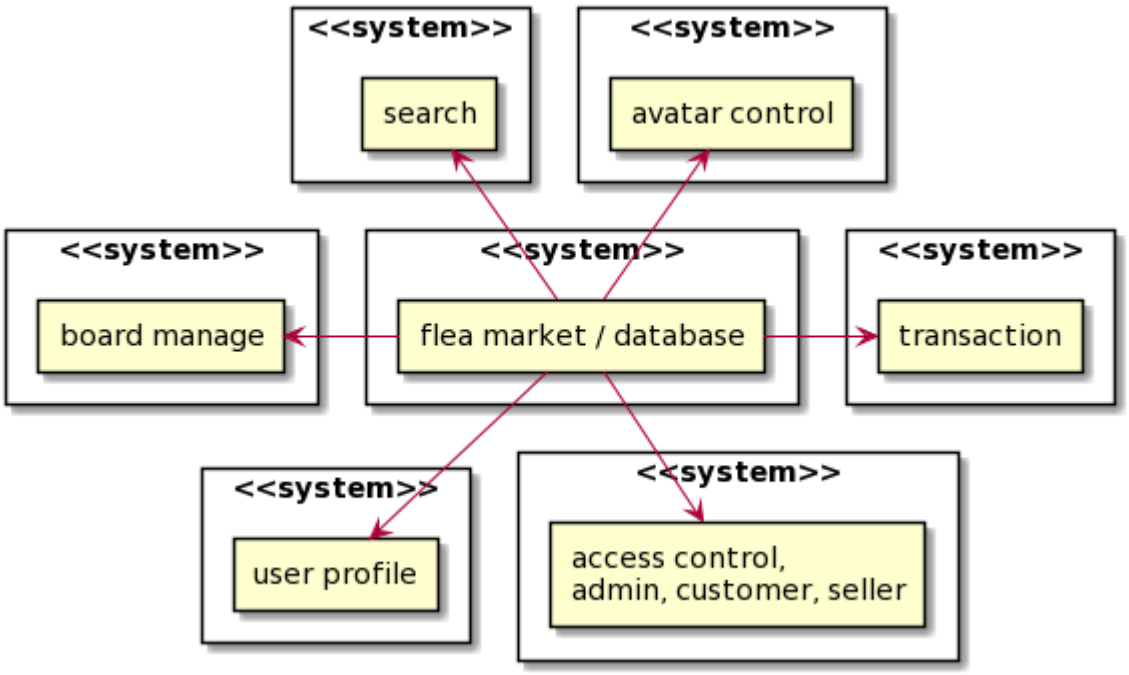
3.1. System Organization

In this system, the user communicates with the system through the 3d world and UI. 3d world and ui can receive user input and pass it to the database. There are things to be controlled at this time, and the user's authority, real-time movement of 3d objects, etc. is controlled by the controller. It is important to control database access rights. UI includes components necessary for users to register and use the system, such as board, pagelist, business, and login. The 3d world is a space where the user moves directly, and admins, buyers, sellers, etc. are active. These views provide users with an interface to access the database. Sensitive information such as username and school id is stored in the database. The stability and integrity of information are important as it is information that users use when transacting goods. By using the VRchat SDK, real-time avatar synchronization with the server can be easily implemented. Each user synchronizes his/her avatar with my avatar in real time.



[Figure 1] MVC – Overall System

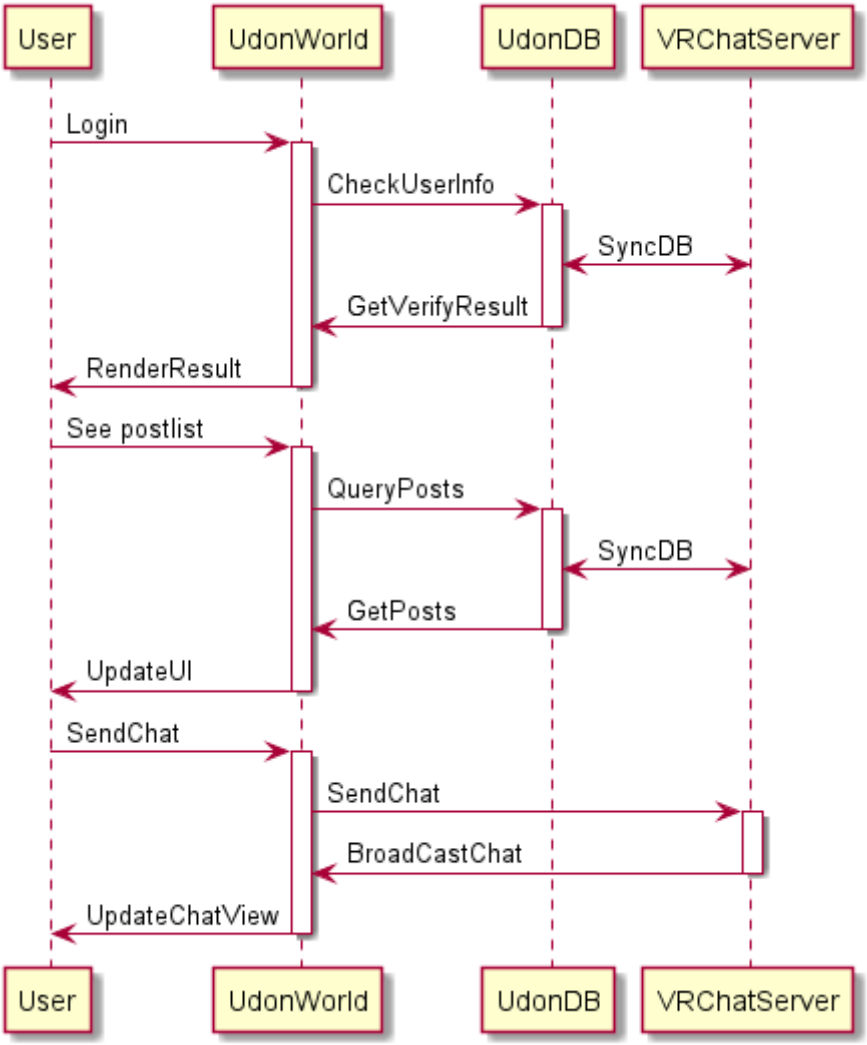
3.2. Context Diagram



[Figure 2] Context Diagram – Overall System

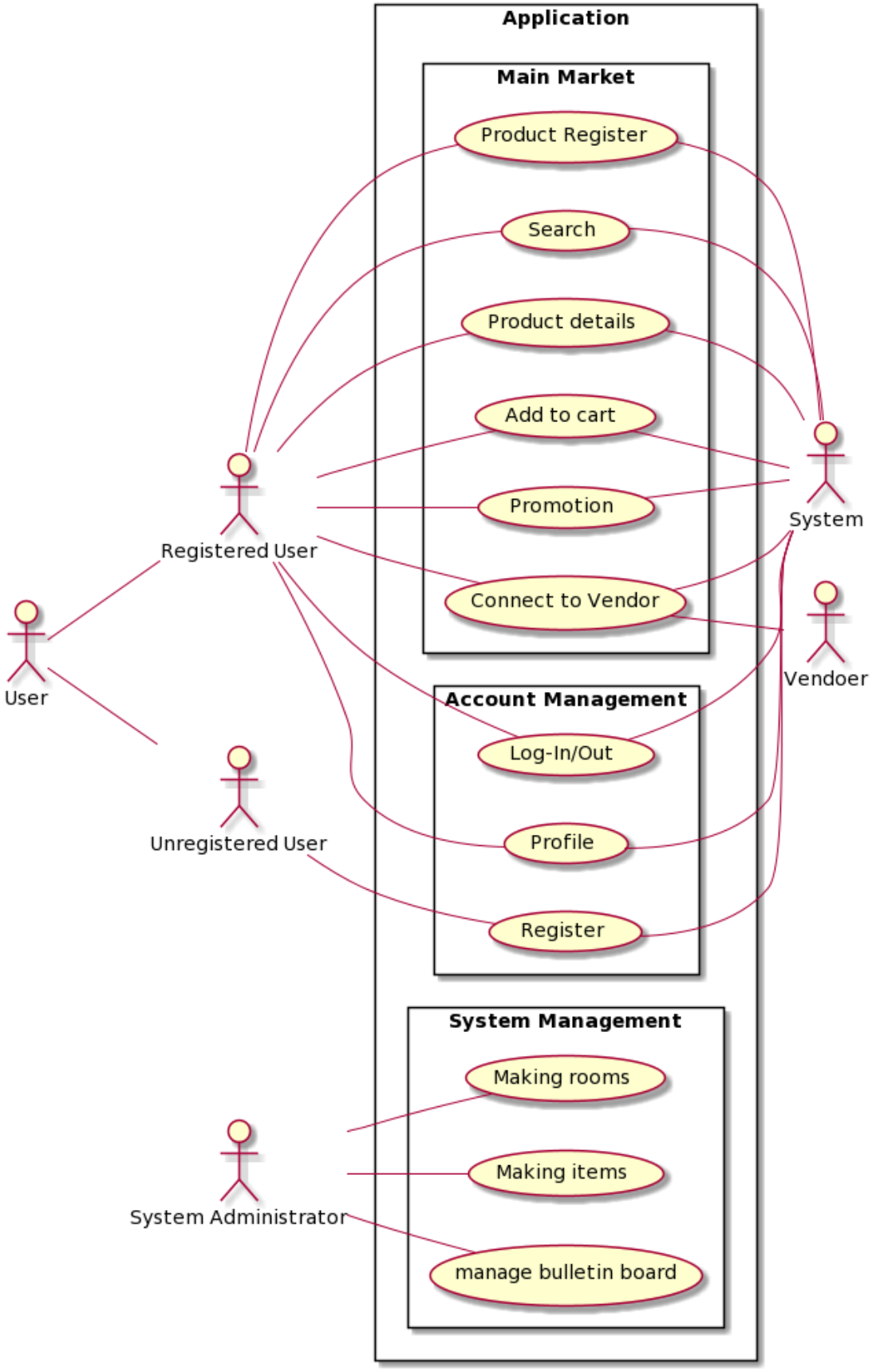
3.3. Sequence Diagram

The sequence diagram shows how users interact with udonworld and VRchat vrchat servers. It largely shows the chat to do business logging in and updating the page list. Because the DB is synchronized in real time due to the nature of the VRchat server, the user only needs to communicate with the local udonDB. Login and post list request queries go through udonDB, and chat does not go through udonDB because it is real-time communication.



[Figure 3] Sequence Diagram – Overall System

3.4. Use case diagram


Application
Main Market
Product Register
Search
Product details
Add to cart
Promotion
Connect to Vendor
Account Management
Log-In/Out
Profile
Register
System Management
Making rooms
Making items
manage bulletin board
Registered User
User
Unregistered User
System Administrator
System
Vendoer

[Figure 4] Use Case Diagram – Overall System

4. System Architecture - Frontend

4.1 Objectives

This chapter describes the functions and explanation of the functions, structure, and execution of the front-end system and how each component is made up.

4.2 Subcomponents

4.2.1. Profile

The profile class represents user information. When registering, unregistered users must enter their information. After logging in, you can check and modify the user's information.

4.2.1.1. Attributes

The attributes of the profile object are as follows:

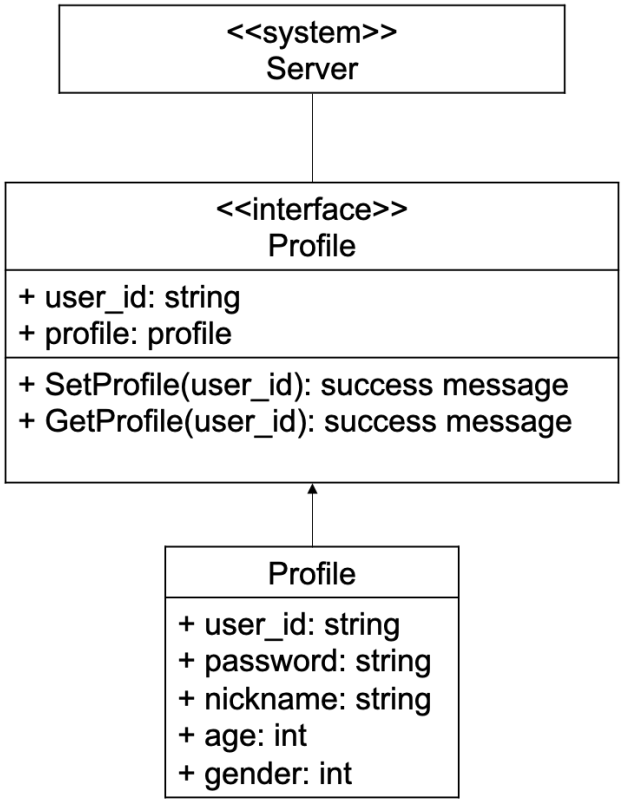
- **User id:** id of the user (email address)
- **Password:** password of the user
- **Nickname:** nickname of the user
- **Age:** age of the user
- **Gender:** gender of the user

4.2.1.2. Methods

The methods of profile object are as follows:

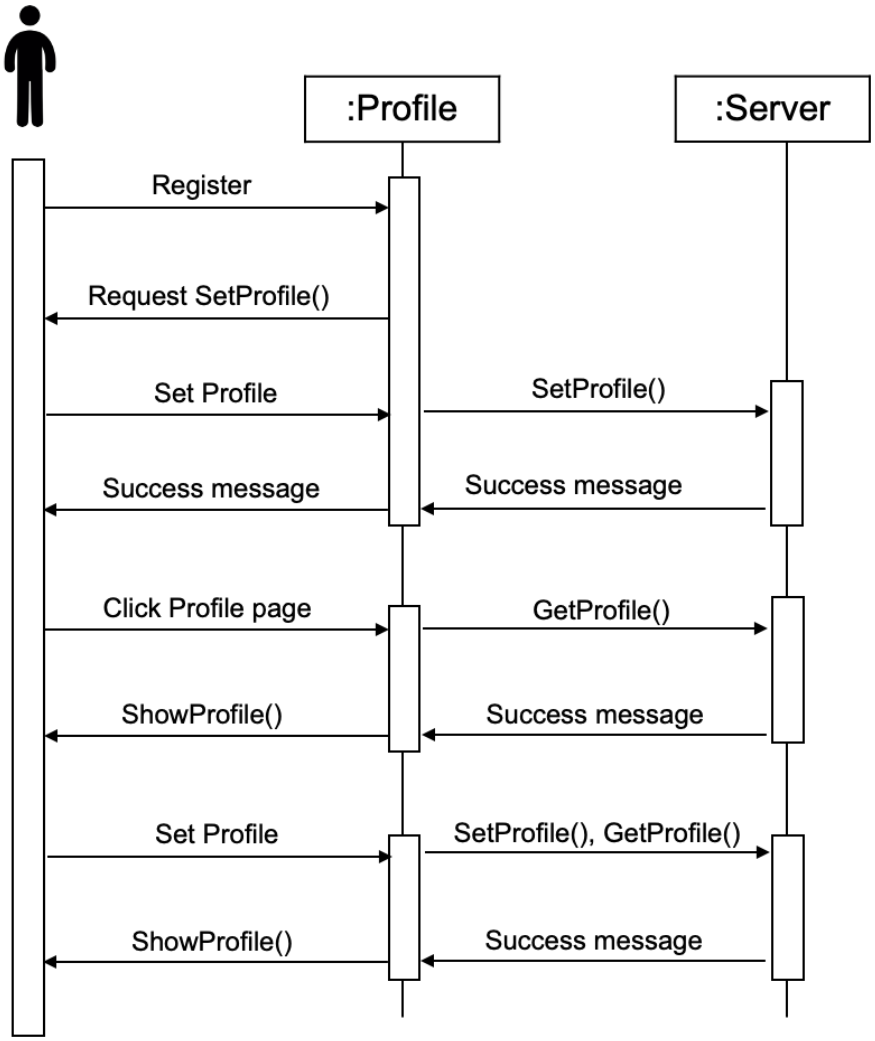
1. SetProfile()
2. GetProfile()
3. ShowProfile()

4.2.1.3. Class diagram



[Figure 5] Class Diagram - Profile

4.2.1.4. Sequence Diagram



[Figure 6] Sequence Diagram - Profile

4.2.2. Search

The Search class indicates when the user searches for a specific product. When the user searches for a specific product on the search screen, it checks if any of the products on the sales list match.

4.2.2.1. Attributes

The attributes of the search object are as follows:

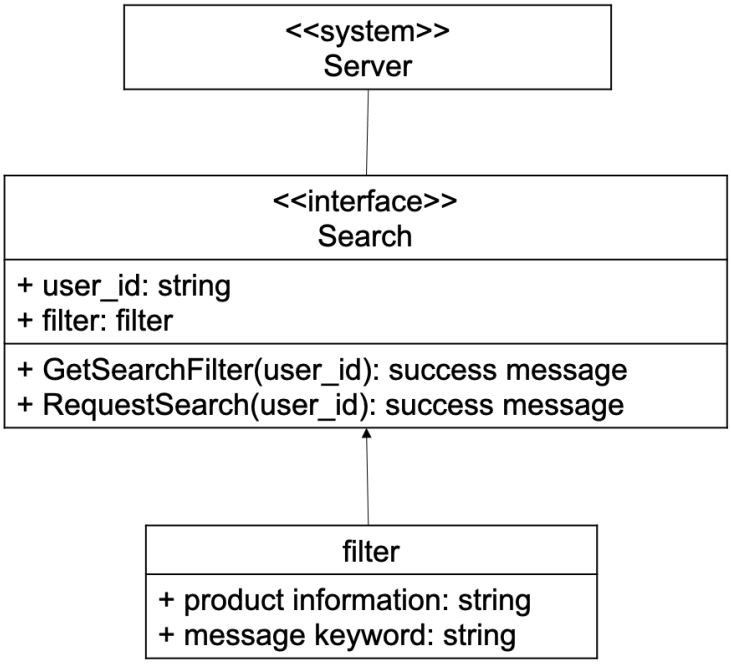
- **Product information:** Information to specify the product (product name or ID)
- **Message keyword:** Keywords for description of the product

4.2.2.2. Methods

The methods of search object are as follows:

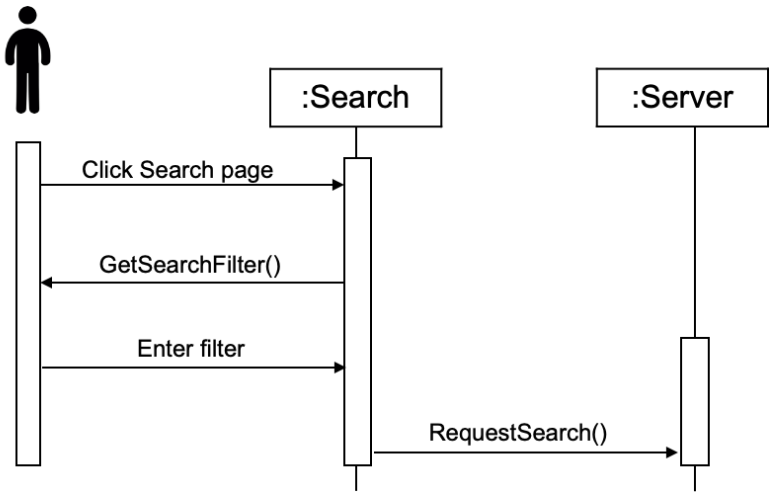
4. GetSearchFilter()
5. RequestSearch()

4.2.2.3. Class Diagram



[Figure 7] Class Diagram - Search

4.2.2.4. Sequence Diagram



[Figure 8] Sequence Diagram - Search

4.2.3. Search Result

The search result class represents a case where the result is displayed on the screen when the user completes the search. Find results that match the input searched by the user in the sales list. It shows the search results to the user.

4.2.3.1. Attributes

The attributes of the search result object are as follows:

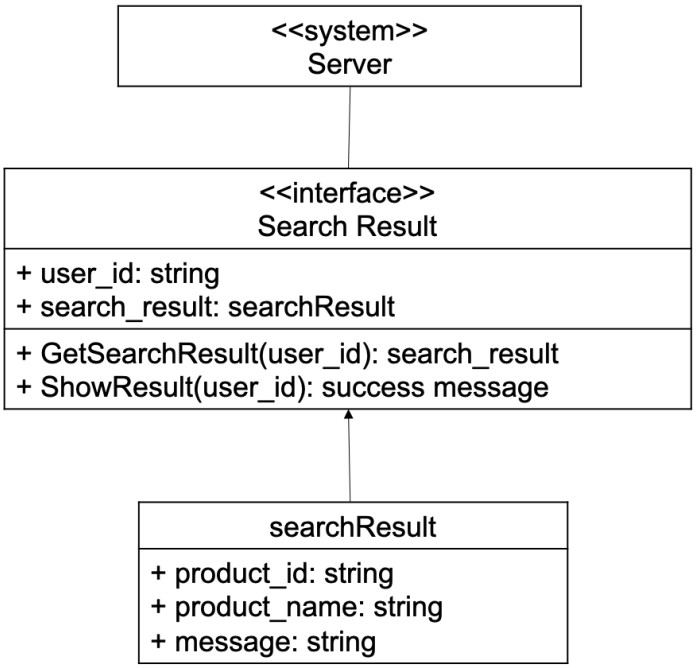
- **Product id:** product unique ID created when uploading a product to the sales list
- **Product name:** title that expresses the sales product
- **Message:** What the seller entered as a description of the product

4.2.3.2. Methods

The methods of search object are as follows:

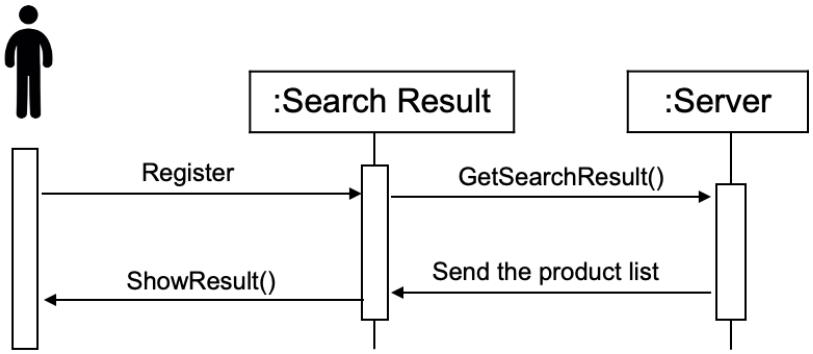
6. GetSearchResult()
7. ShowResult()

4.2.3.3. Class Diagram



[Figure 9] Class Diagram – Search Result

4.2.3.4. Sequence Diagram



[Figure 10] Sequence Diagram – Search Result

4.2.4. Promotion

Promotion class refers to a case where a user posts a product or event on a bulletin board when he or she wants to promote it more actively to the user. When a user passes a specific procedure and provides information about an item or event to be promoted to the administrator, the administrator causes it to appear on the bulletin board screen.

4.2.4.1. Attributes

The attributes of the promotion object are as follows:

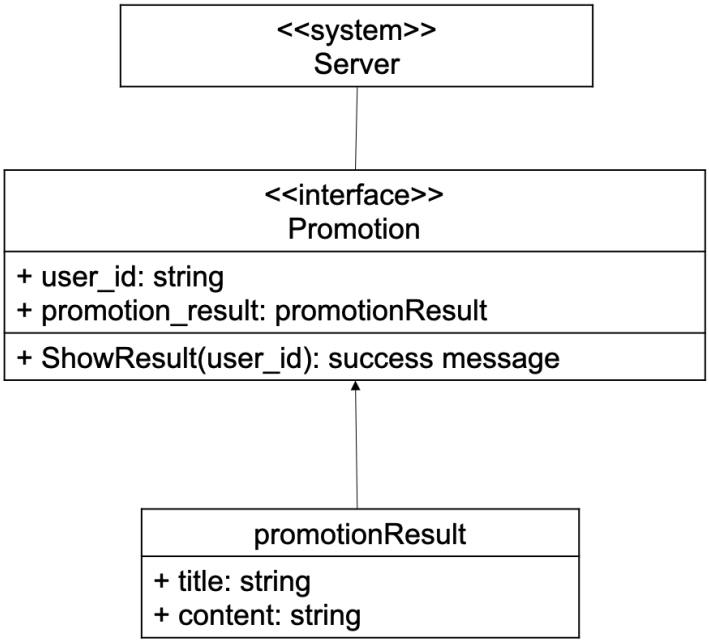
- **Title:** It can be the name of the product you sell or the title of the event
- **Contents:** It can be an introduction to a store that sells a product or an explanation of an event

4.2.4.2. Methods

The methods of promotion object are as follows:

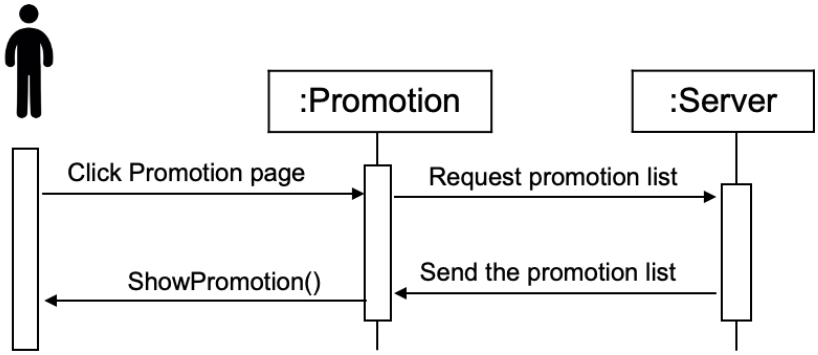
8. ShowPromotion()

4.2.4.3. Class Diagram



[Figure 11] Class Diagram - Promotion

4.2.4.4. Sequence Diagram



[Figure 12] Sequence Diagram - Promotion

4.2.5. Cart

The cart class represents a case where a user collects products that he or she wants to remember through add to cart. When the user sees the product sold and clicks add to cart, the product information is stored on the cart page and can be easily checked. If the user thinks he or she needs to remove the product from the cart, he or she can delete it within the cart page by clicking move from cart.

4.2.5.1. Attributes

The attributes of the cart object are as follows:

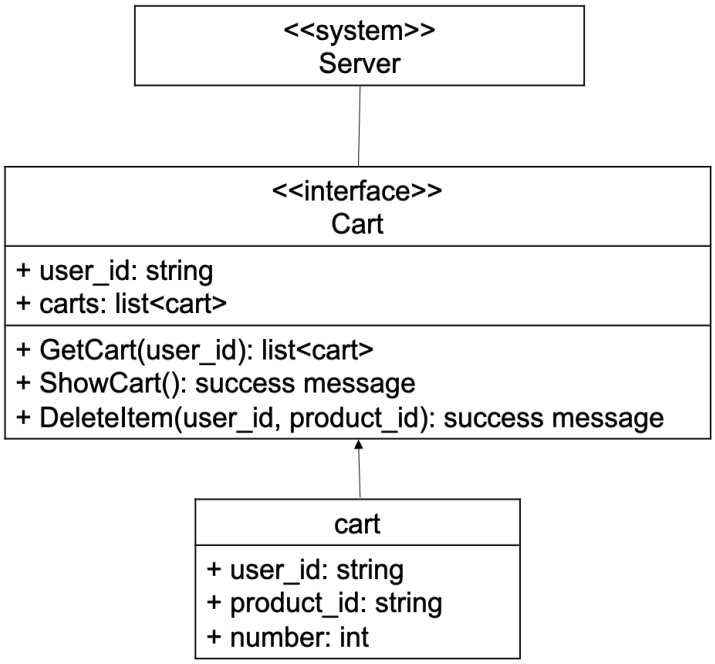
- **User id:** id of the user
- **Product id:** product unique ID created when uploading a product to the sales list
- **number:** number for the item

4.2.5.2. Methods

The methods of cart object are as follows:

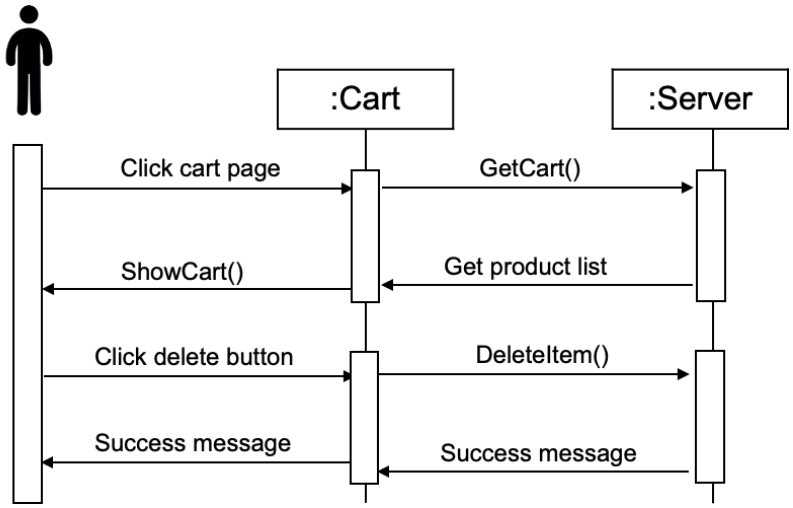
9. GetCart()
10. ShowCart()
11. DeleteItem()

4.2.5.3. Class Diagram



[Figure 13] Class Diagram - Cart

4.2.5.4. Sequence Diagram



[Figure 14] Sequence Diagram - Cart

4.2.6. Product details

The product details class indicates a case where detailed information about the product can be checked. When a user clicks on a specific product on the product list, you can check more information about the product uploaded by the seller.

4.2.6.1. Attributes

The attributes of the product details object are as follows:

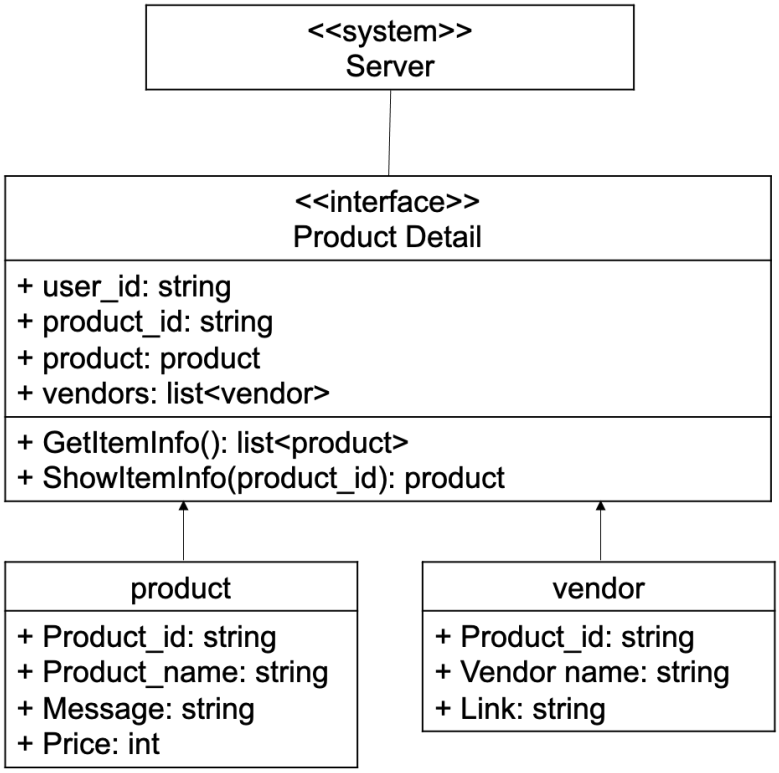
- **Product id:** product unique ID
- **Product name:** title that expresses the sales product
- **Message:** What the seller entered as a description of the product
- **Price:** price of the product

4.2.6.2. Methods

The methods of product details object are as follows:

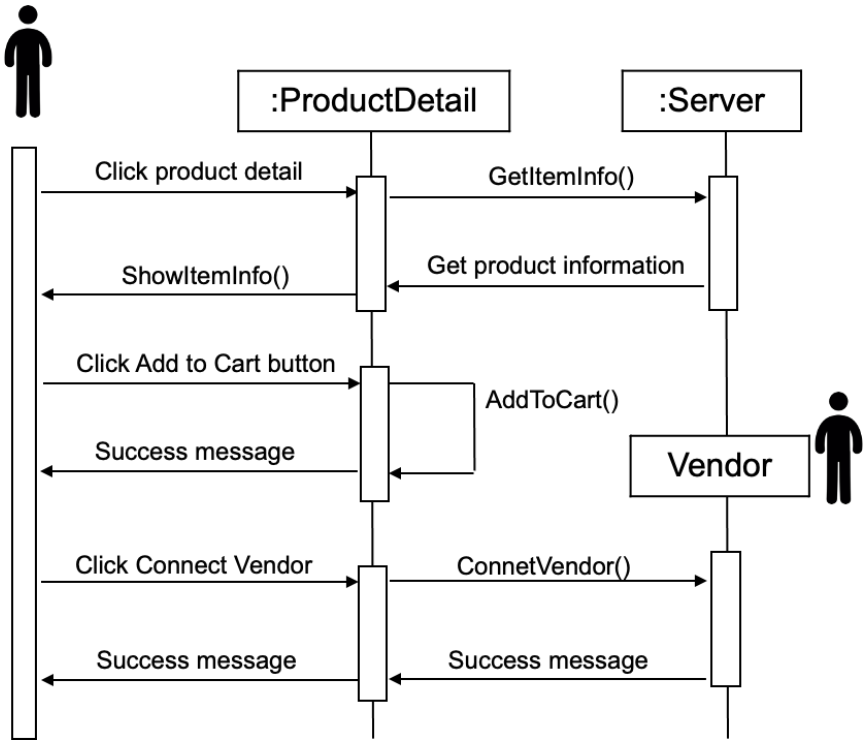
12. GetItemInfo()
13. ShowItemInfo()
14. AddToCart()
15. ConnectVendor()

4.2.6.3. Class Diagram



[Figure 15] Class Diagram – Product Details

4.2.6.4. Sequence Diagram



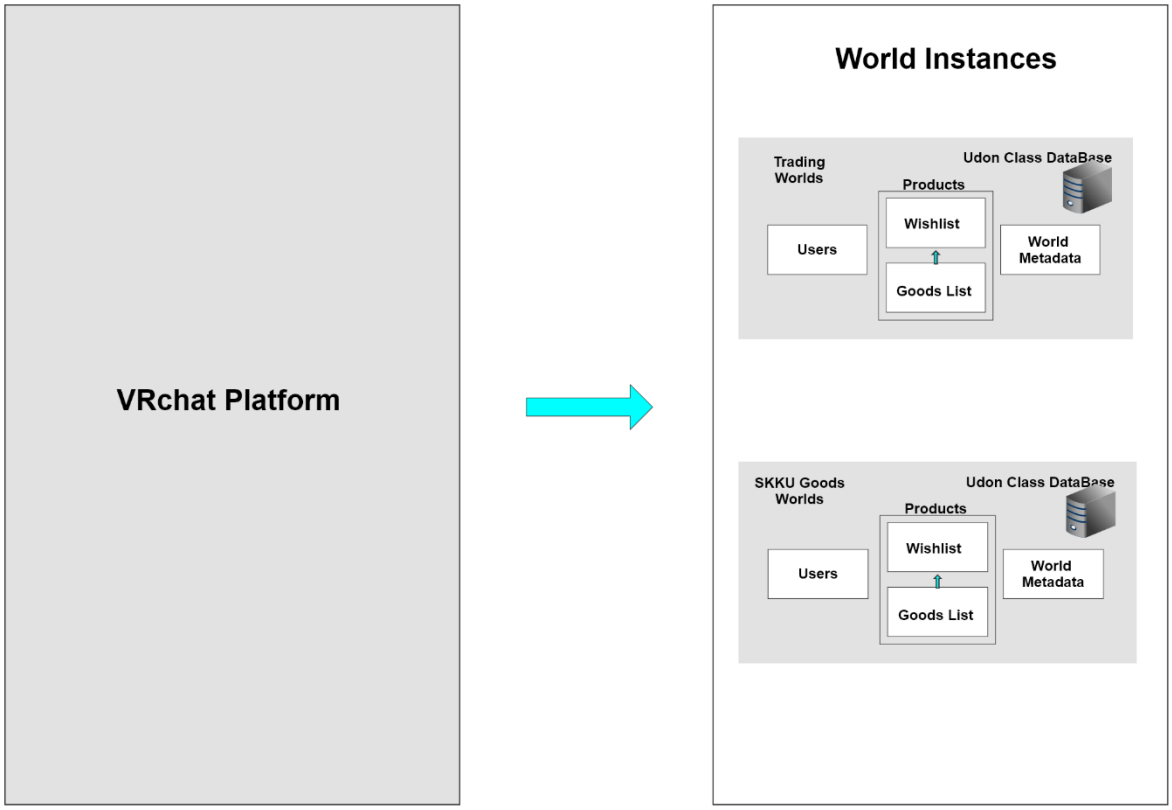
[Figure 16] Sequence Diagram – Product Details

5. System Architecture – Backend

5.1. Objectives

This chapter describes the structure of the back-end system include DB and API Cloud include. In particular, in VRchat, since it cannot communicate with an external database, a virtual DB is implemented using Udon objects in the world.

5.2. Overall Architecture



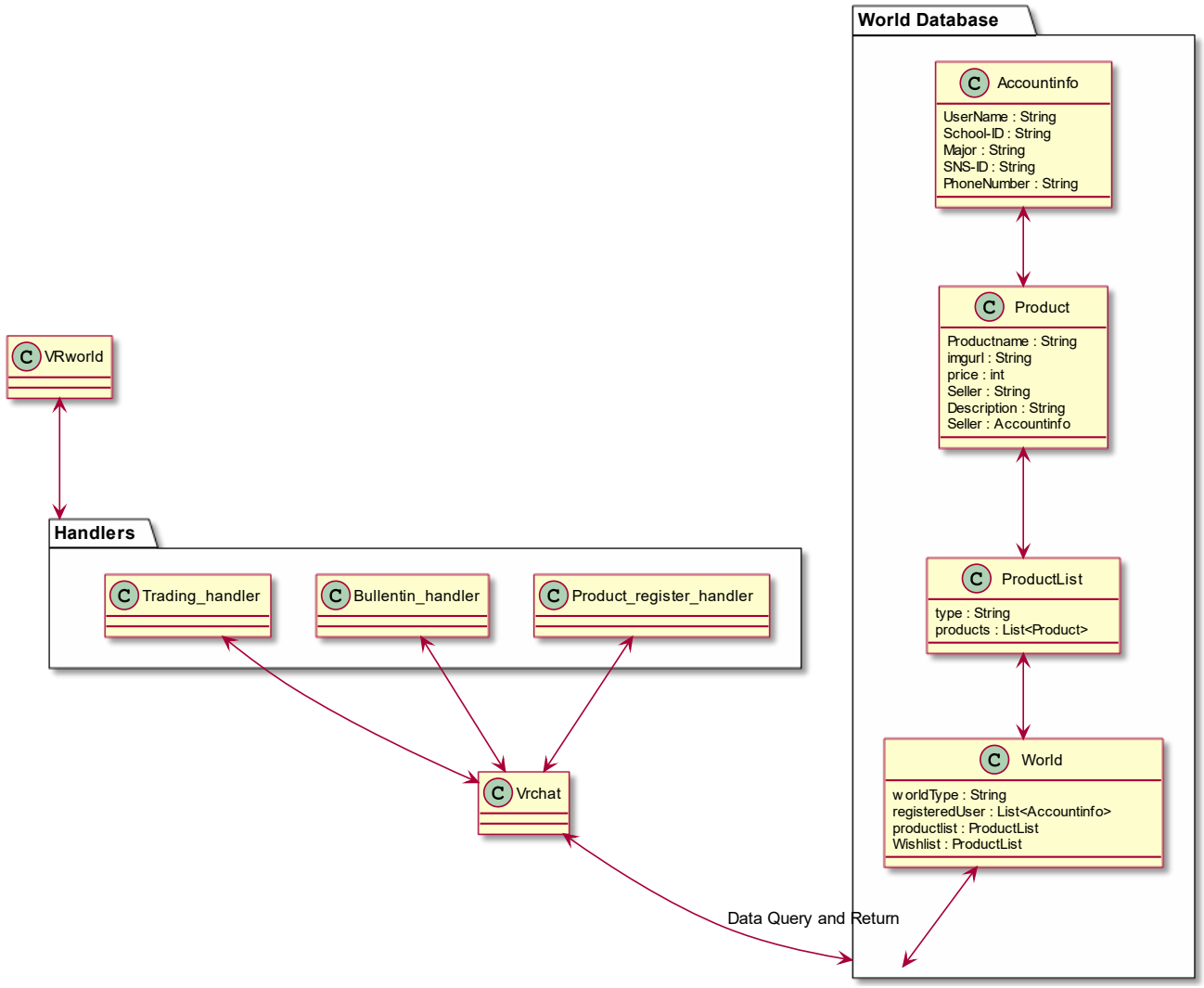
[Figure 17] Overall Architecture

The overall architecture of the SKKU flea market is as above, User accesses the VRchat through Steam. User selects and accesses one of the World instruments within the VRchat. Each World instance queries information on Products and Users, returns it to VRchat, and displays it.

Each World instance manages its DB separately in the form of Udon Class. User queries the Product List and Wish List's Product in the Udon Class Database in the World. In addition, user information accessed within the world instance is stored in the Udon Class DB in the form of a User Class.

5.3 Subcomponent

5.3.1 World instance



[Figure 18] World Class Diagram

Both Trading World and SKKU Goods market World have a common ‘World’ Class.

5.3.1.1. Account Info

Information of Account, It stores user information.

5.3.1.2. Product

It contains information about a Product. It can be listed by ProductList class

5.3.1.3. World

It has a class containing information of the entire Walls, a user accessing it, a list of goods, and a wish list as member variables.

5.3.1.4. VRWorld

VRworld refers to a VRchat instance that the actual user accesses and sees with his or her eyes.

5.3.1.5. VRchat

Software interface supported by VRchat platform

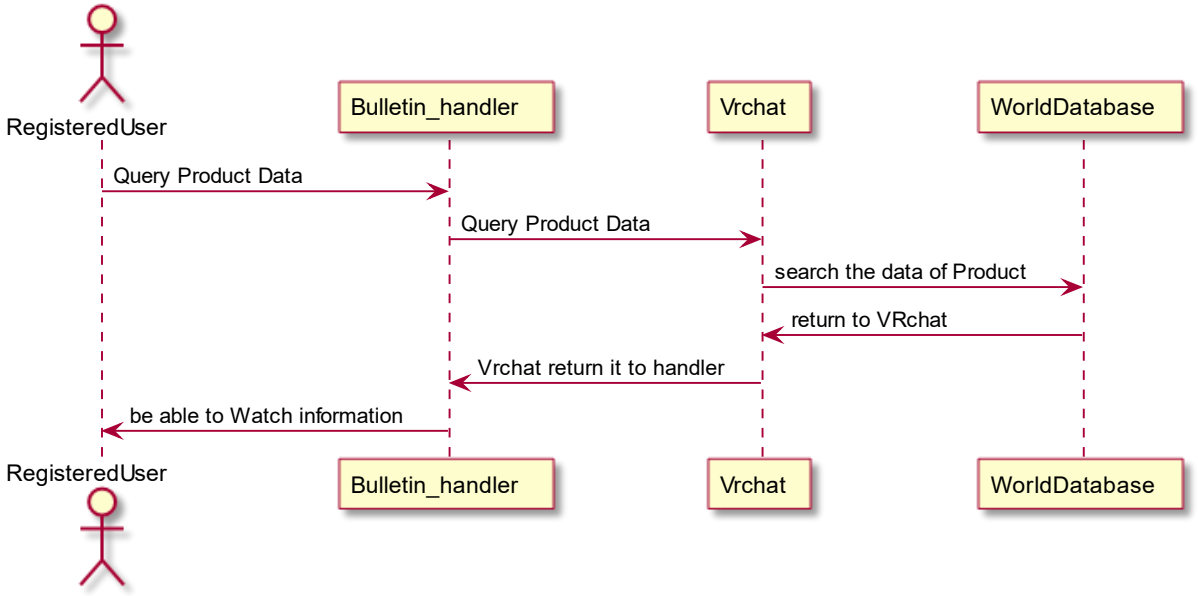
5.3.1.6. Handlers

Handlers are described in detail in 5.3.2 Subcomponents. It is a function that connects Database and VRworld. There are three types of Handlers in the SKKU flare market.

5.3.2. Subcomponents

All subcomponent systems operate on the World Database system. Each World instance manages its own World Database object.

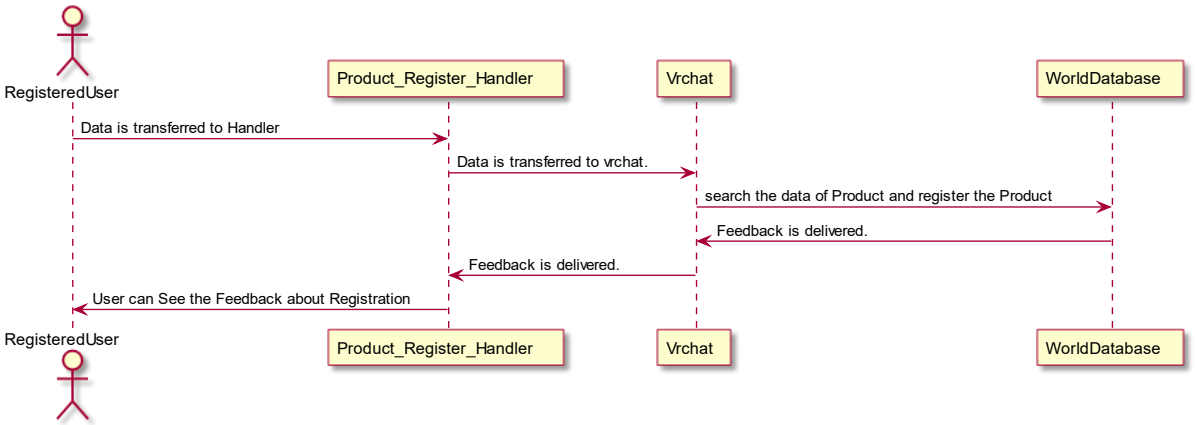
5.3.2.1 Bulletin Board System



[Figure 19] Sequence Diagram – Bulletin Board

The system should be able to display information on the product stored in the DB to the world using the bulletin board. To implement this function, the flare market queries data through Bulletin_Handler in the World System. The process of querying data is as shown in the sequence diagram above.

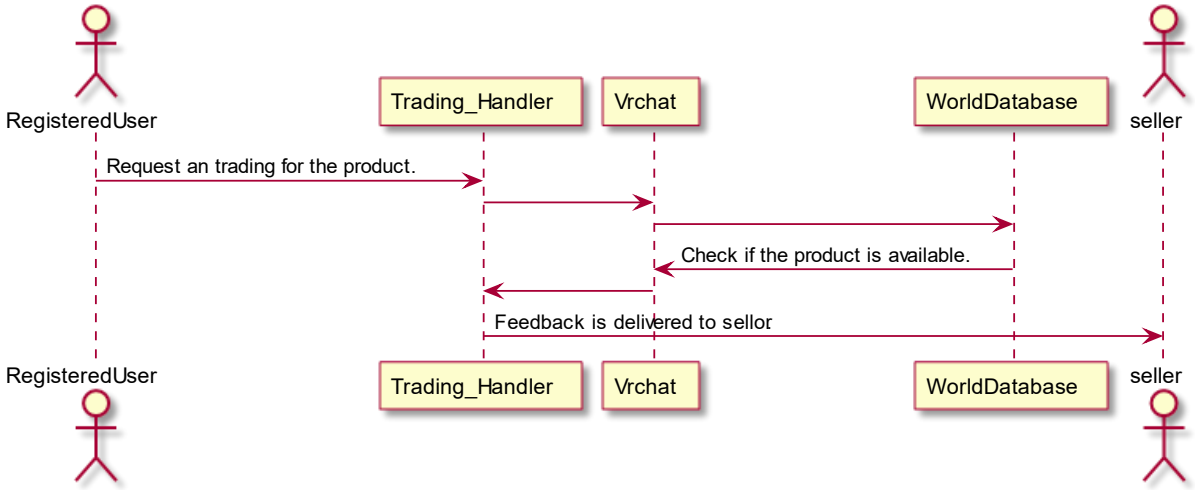
5.3.2.2 Product registration system



[Figure 20] Sequence Diagram – Product Registration

Users should be able to register their products in the World. When their information and data about Product are entered on the bulletin board, Product_Register_Handler delivers the data to the World Database. After that, the feedback is delivered to the user according to the success of product registration. The entire process is the same as the sequence diagram above.

5.3.2.3 Trading System



[Figure 21] Sequence Diagram – Trading System

Regardless of whether it is SKKU flea market or goods market, User and Seller communicate through Trading_Handler. The user requests the system to buy the product on the Bulletin Board. After inquiring the availability of the product in WorldDB, Trading Handler delivers feedback to Seller. Seller receives information about the purchase requester, or feedback from the system.

6. Design

6.1. Objectives

This chapter describes the overall structures which are used for the protocols which are used for interaction between each subsystem, especially SKKU Flea Market system and the Udon. Also, this chapter describes how each interface is defined.

6.2. VR Chat Udon

VR Chat Udon is a node format programming language, which is developed by VR Chat development team. This programming language uses Udon Node Graph, which is based on a visible interface through connecting nodes and wires like Blueprint from Unrealengine.

VR Chat Udon not only allows users to replicate all actions of triggers and actions, but also allows you to interact with your own actions, scenes, or players. Additionally, Udon works on both VRChat clients and Unity editors, making it easier to debug and see if it works properly.

6.3. Authentication

6.3.1 Log In

- Request

[Table 1] Table of login request

Attribute	Detail	
Method	VR Chat Udon trigger	
Node Body	Get localplayer	get local player’s api
	isValid	check the local player is valid

- Response

[Table 2] Table of login response

Attribute	Detail	
Success	follow the IsValid: true flow	
Failure	follow the IsValid: false flow	
	Node loop	Keep check localplayer is valid
Success Node Body	Message	Success log message
Failure Body	Message	Failure log message

6.4. Product of Users

6.4.1. Register product

- Request

[Table 3] Table of register product request

Attribute	Detail	
Method	VR Chat Udon trigger	
Node body	UserID	user id
	Index	product registering index
	Title	input title of product from user

	Content	input content of product from user
	Price	input price of product from user
	Image	send image to admin if user wants to register
	IsValid	check all content been written

- Response

[Table 4] Table of register product response

Attribute	Detail	
Success	follow isRegistered: true flow	
Failure	follow isRegistered: false flow	
Success Node Body	Message	Success log message
	new_product	Register a new product
Failure Node Body	Message	Failure log message

6.4.2. Delete product

- Request

[Table 5] Table of delete product request

Attribute	Detail

Method	VR Chat Udon trigger	
Node body	UserID	user id
	Index	product registering index
	IsValid	check user id & product index

- Response

[Table 6] Table of delete product response

Attribute	Detail	
Success	follow isValid: true flow	
Failure	follow isValid: false flow	
Success Node Body	Message	Success log message
	delete_product	Delete the selected product
Failure Node Body	Message	Failure log message

6.4.3. Buy product

- Request

[Table 7] Table of buy product request

Attribute	Detail
Method	VR Chat Udon trigger

Node body	UserID	user id
	Index	product index
	onClick	get button click input
	IsValid	check user id & onClick: true

- Response

[Table 8] Table of buy product response

Attribute	Detail	
Success	follow isReserved: true flow	
Failure	follow isReserved: false flow	
Success Node Body	Message	Success log message
	button_color_change	Make button color blue to red to notify the reservation is made
Failure Node Body	Message	Failure log message

6.4.4. Search product

- Request

[Table 9] Table of search product request

Attribute	Detail
Method	VR Chat Udon trigger

Node body	Title	input title
	onClick	check search button clicked
	IsValid	check title & onClick: true

- Response

[Table 10] Table of search product response

Attribute	Detail	
Success	follow isReserved: true flow	
Failure	follow isReserved: false flow	
Success Node Body	Message	Success log message
	button_color_change	Make button color blue to red to notify the reservation is made
Failure Node Body	Message	Failure log message

6.5. Product of school goods

6.5.1 Buy product

- Request

[Table 11] Table of buy product of school goods request

Attribute	Detail	
Method	VR Chat Udon trigger	
Node body	Index	product index
	onClick	check the buy button clicked
	IsValid	check the product is valid & onClick : true

- Response

[Table 12] Table of buy product of school goods response

Attribute	Detail	
Success	follow IsValid: true flow	
Failure	follow IsValid: false flow	
Success Node Body	Message	Success log message
	show_QR	show qr code of the product
Failure Node Body	Message	Failure log message

6.6. Bulletin Board

6.6.1. Upload Promotion

- Request

[Table 13] Table of upload promotion request

Attribute	Detail	
Method	VR Chat Udon trigger	
Node body	UserID	user id
	Index	promotion index
	Content	input contents for promotion
	onClick	check the upload button clicked
	IsValid	check id, index, content & onClick : true

- Response

[Table 14] Table of upload promotion response

Attribute	Detail	
Success	follow IsValid: true flow	
Failure	follow IsValid: false flow	
Success Node Body	Message	Success log message
	new_promotion	Register a new promotion
Failure Node Body	Message	Failure log message

6.6.2. Delete promotion

- Request

[Table 15] Table of delete promotion request

Attribute	Detail	
Method	VR Chat Udon trigger	
Node body	UserID	user id
	Index	promotion index
	IsValid	check user id & product index

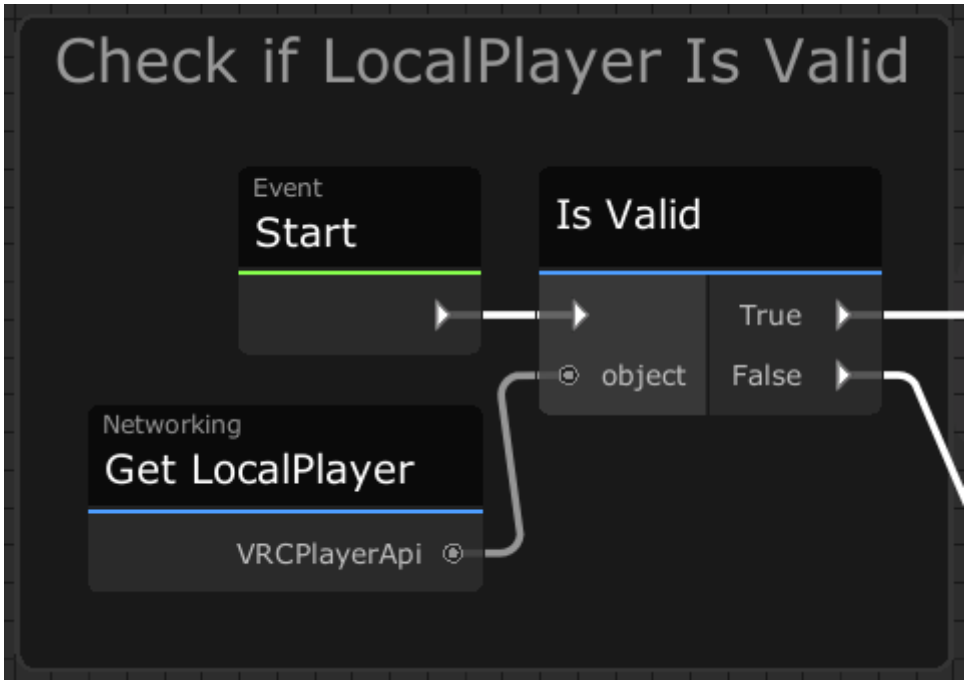
- Response

[Table 16] Table of delete promotion response

Attribute	Detail	
Success	follow IsValid: true flow	
Failure	follow IsValid: false flow	
Success Node Body	Message	Success log message
	delete_promotion	Delete the selected promotion
Failure Node Body	Message	Failure log message

7. Entities

7.1. Log In

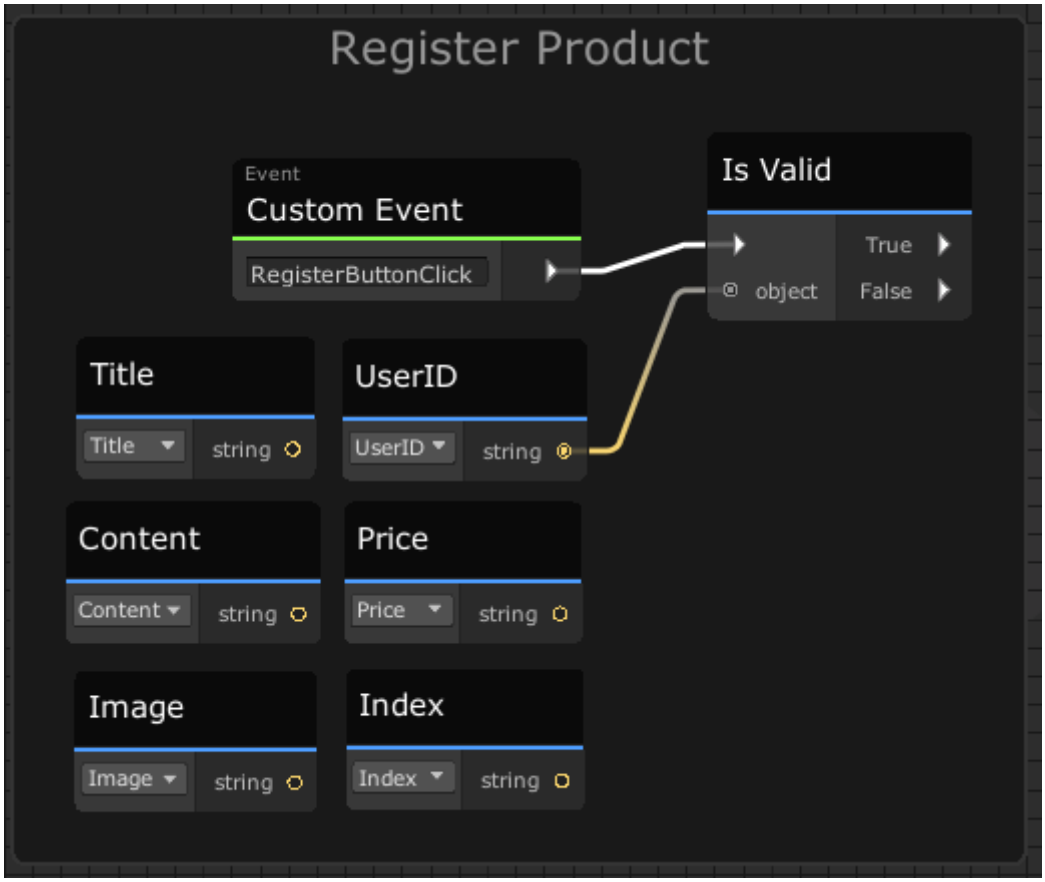


[Figure 22] Entity - Login

Event starting point is activated when the user comes into the SKKU Flea Market VRChat world. From VRChat, Get LocalPlayer trigger gets the VRChat player's API from the VRChat, and checks if the user with the api is a valid user in this 'Check if LocalPlayer Is Valid' part. If from Is Valid: True, the user can get the authentication to get in the SKKU Flea Market VRChat world. If Is Valid: False, users need to check their own ID from the VRChat system and retry the log in session.

7.2. Product of users

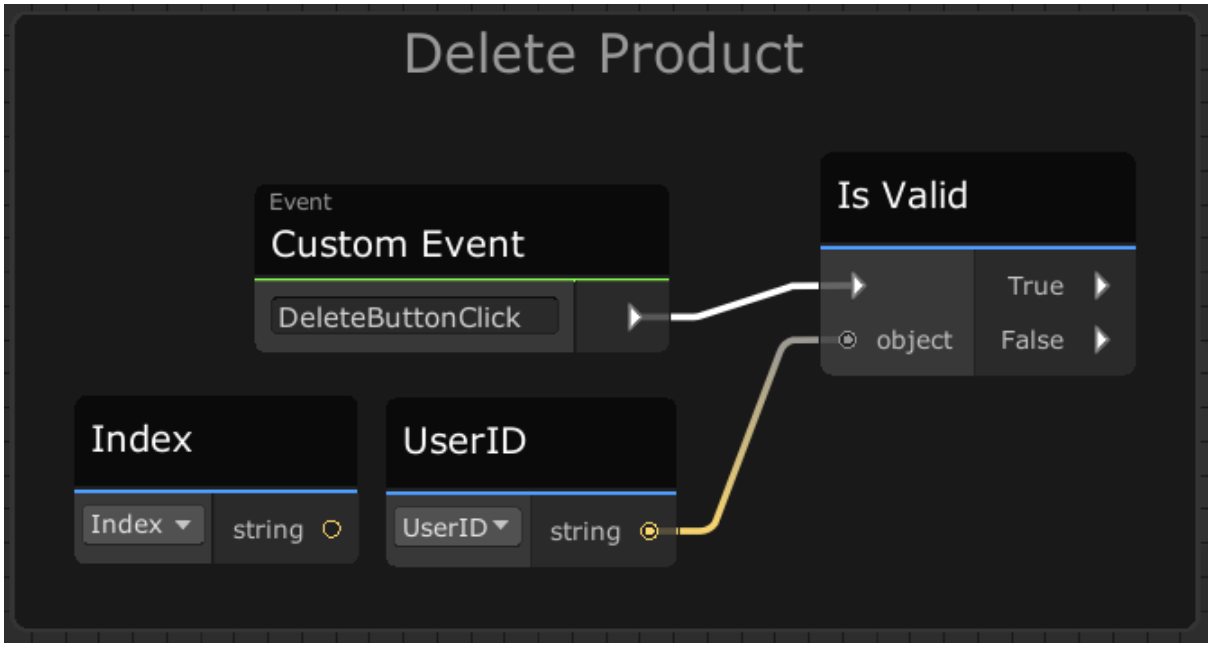
7.2.1 Register product



[Figure 23] Entity – Register Product

When users come into the SKKU Flea Market VRChat world, users can register their product on this SKKU Flea Market VRChat world. To upload their products, users need to fill up the title, content, price, index and if there is image, image as well. When users finish filling up the product contents, users click the upload button to register the products. If users didn't fill up the content, price, or title, IsValid trigger would be false, and this led to users to fill up the blanks again. If IsValid: true, the button activates, and the product will be uploaded in the SKKU Flea Market VRChat world marketplaces where the user clicked the button. So the other users can check the products.

7.2.2. Delete product



[Figure 24] Entity – Delete Product

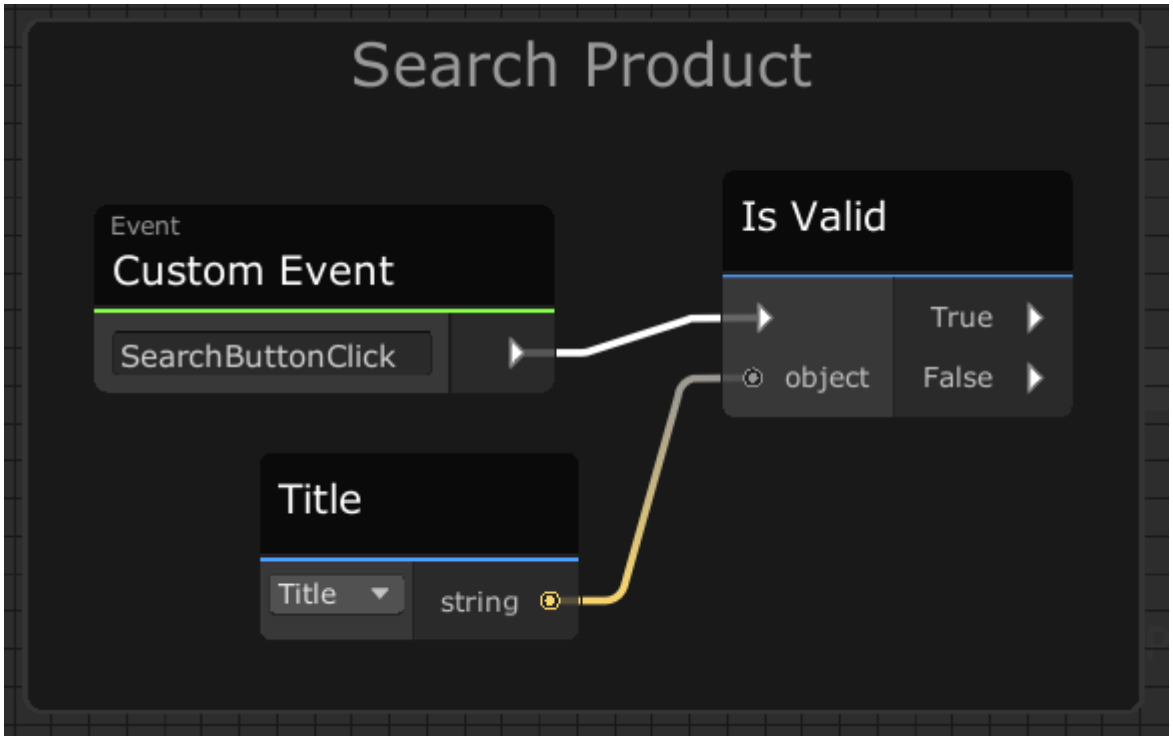
When users register the product, if the users want to change the price of product, or the content of the product, and want to delete the product if the product has been sold, users can use this 'Delete Product'. If users clicked the delete button, the 'Delete Product' will check the UserId and the product Index that the product is from the same user, and if the user is who registered the product, they can now delete the product. If the user is not the user who registered the product will get IsValid: false, so the abuse of deleting product can be managed.

7.2.3. Buy product

[Figure 25] Entity – Buy Product

The users can be separated with two groups, and can be compatible, which are the buyers and sellers. This is for the users who want to buy some products. Users can click the product which is needed to buy, so the user clicks on the button, button color will be changed, and the seller will let the product that users uploaded to sell, have a buyer. After IsValid: true, the match is made.

7.2.4. Search product

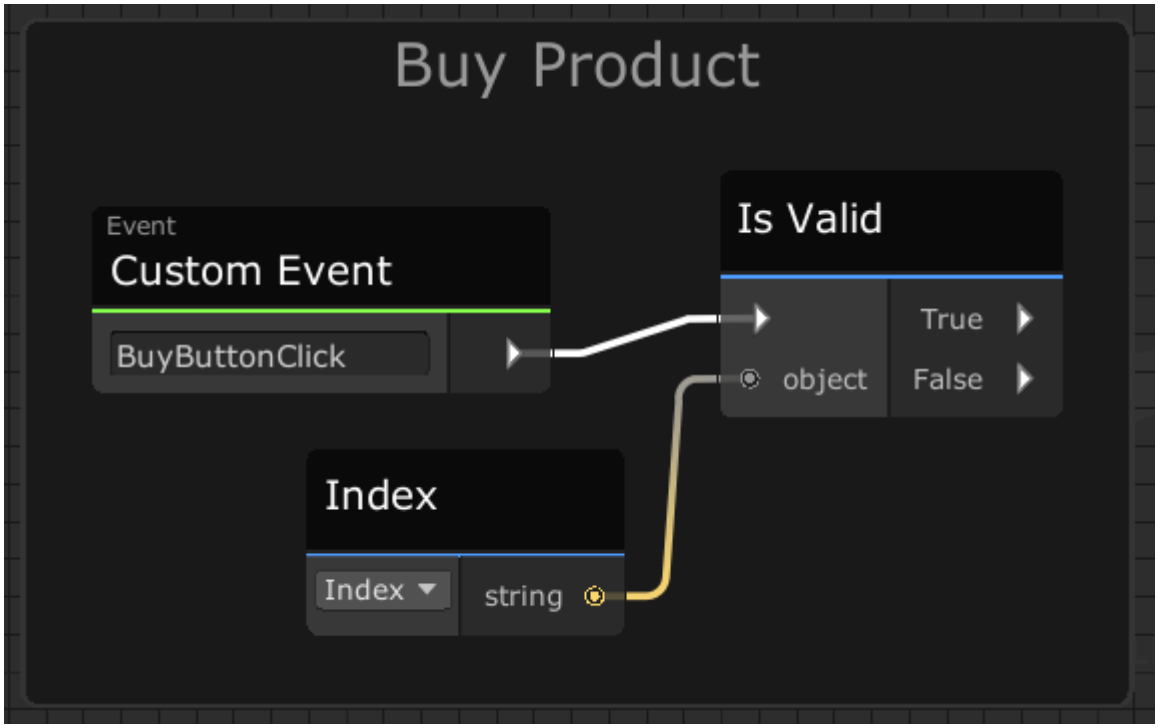


[Figure 26] Entity – Search Product

Users can search for the product by title. They can fill out the blanks for the product. If the blank is empty, the IsValid will return false, and the search function will not operate.

7.3. Product of school goods

7.3.1. Buy product

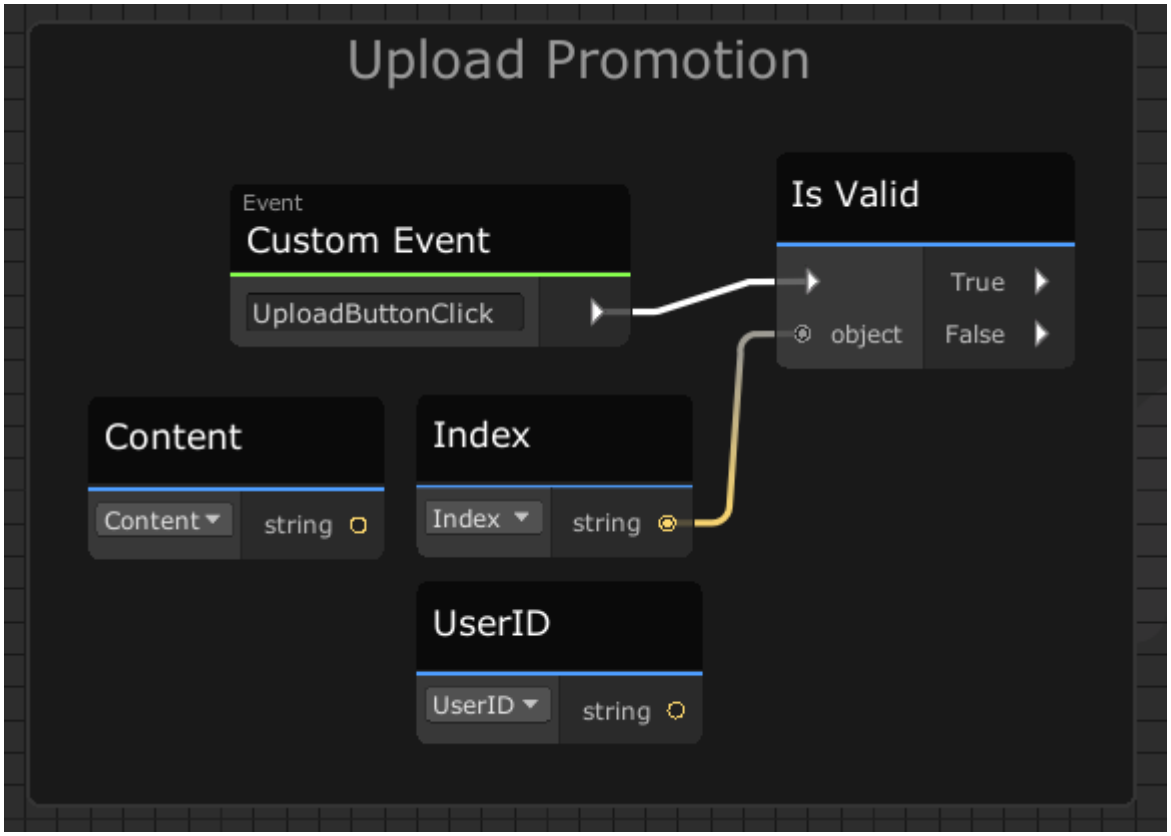


[Figure 27] Entity – Buy Product

There are many items which are closely related to SKKU or represent SKKU. If users want to buy or see those items, they can click the buy button. After doing that, a QR code of the selected items will appear and the website which is selling that item will be linked in a new web page.

7.4. Bulletin Board

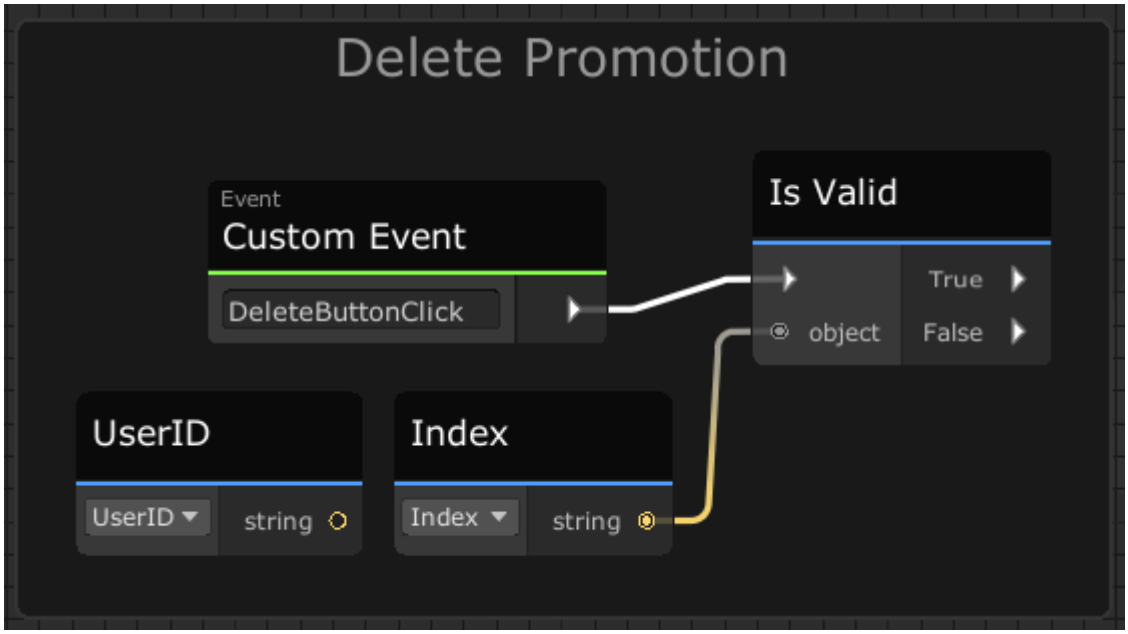
7.4.1. Upload Promotion



[Figure 28] Entity – Upload Promotion

When users want to promote their products or something else, they can use a bulletin board for promotion. By pressing the uploading button after filling the contents of what they promote, they can upload their promotion on bulletin board in SKKU Flea Market VRchat world.

7.4.2. Delete Promotion



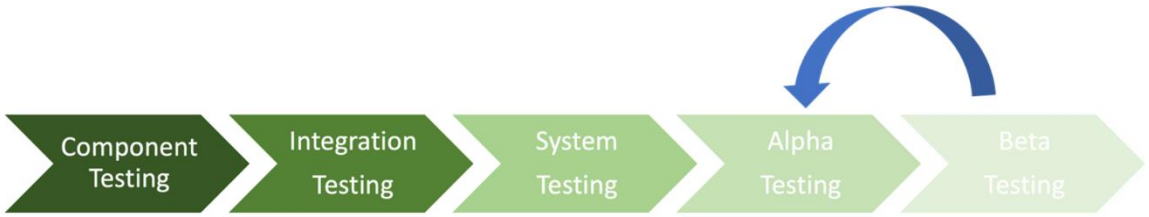
[Figure 29] Entity – Delete Promotion

When users who uploaded their promotions on bulletin boards in SKKU Flea Market VRchat world want to delete the promotions, they can press the delete button. In this way, the user id and index of the promotions go through the process which checks that it is valid. After the function 'isValid' returns true value, the selected promotion is deleted from the bulletin board.

8. Testing Plan

8.1. Objectives

In this chapter, it is largely composed of three aspects : development testing, release testing, and user testing. The goal of testing is to evaluate and identify errors, gaps or missing requirements in contrast to actual requirements. Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss.



[Figure 30] Testing Plan

8.2. Testing Policy

8.2.1. Development Testing

It is a method of applying testing practices consistently throughout the software development life cycle process. This testing ensures the detection of bugs or errors at the right time which further ensures delay of any kind of risk in terms of time and cost. Development Testing aims to establish a framework to verify whether the requirements of a given project are met in accordance with the rules of the mission to be accomplished. This testing is performed by the software developers or other engineers during the construction phase of the software development lifecycle (SDLC). In Development Testing, the phases are more tightly integrated so that code that is being written and checked in is automatically tested. In this way, the problems can be more quickly discovered and can be addressed.

8.2.1.1. Performance

Our system stores most of the data on a server and obtains and renews data through communication with the server, so the latency of this communication is a large part of the performance. Therefore, it is necessary to test the configuration and communication of efficient databases to ensure an appropriate level of stable latency.

In addition, one of the most important function of our system is chatting system. Therefore, the speed and efficiency of chatting system will be tested. For characteristics such as reliability, security, and maintainability in a trade-off relationship with performance, it is necessary to test whether performance has been sacrificed excessively and to improve if excessive performance degradation occurs.

8.2.1.2. Acceptability

Acceptability is the characteristic of a thing being subject to acceptance for some purpose. It is a formal testing according to user needs, requirements and business processes conducted to determine whether a system satisfies the acceptance criteria or not and to enable the users, customers or other authorized entities to determine whether to accept the system or not. Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

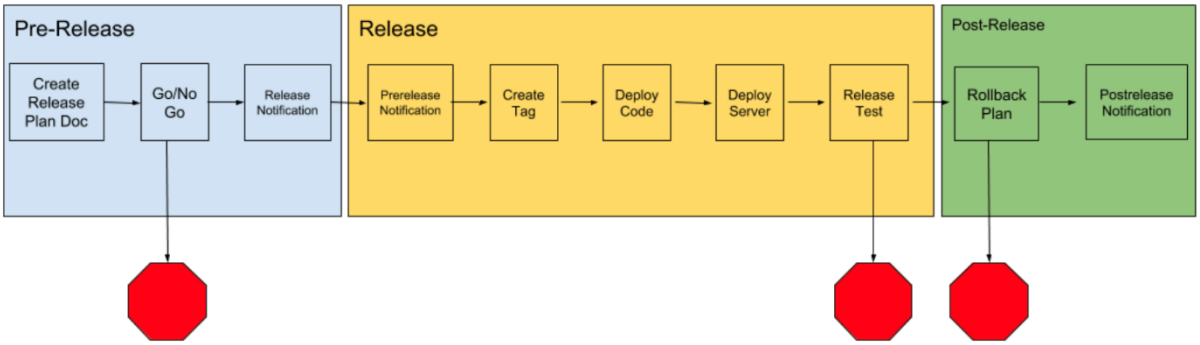
8.2.1.3. Security

Software security is the application of techniques that assess, mitigate, and protect software systems from vulnerabilities. These techniques ensure that software continues to function and are safe from attacks. Developing secure software involves considering security at every stage of the life cycle. The major goal is to identify flaws and defects as early as possible. It should also be tested to ensure that the personal information of users stored in the server's database is well encrypted and protected. Additionally, it should be tested to ensure that there are no loopholes in communication between the payment API and the system and that there are no security issues.

8.2.1.4. Maintainability and Sustainability

The system utilizes already commercialized cloud services to maximize the ease of server management and maintenance. In addition, the system should be organized to make it easier to reflect various changes, such as adding new items, changing status, and changing prices. Therefore, we check that the server is designed to be easy to maintain and fluctuate.

8.2.2. Release Testing



[Figure 31] Release Testing

Release testing refers to coding practices and test strategies that give teams confidence that a software release candidate is ready for users. Release testing aims to find and eliminate errors and bugs from a software release so that it can be released to users. The objective of release testing is to build confidence into a release candidate. Release testing is a testing approach or strategy rather than one single grand testing method.

Scenario testing is used to validate and demonstrate the system. We devise

general usage scenarios and use them to conduct release testing. Scenarios should be realistic and in practice like users using the system. Scenarios in the requirements engineering process can also be utilized for scenario testing. Release testing begins with the alpha version, where the basic implementation of the software has been completed.

8.2.3. User Testing

Usability is one of the key factors in this system, so it should be easy for anyone to use and intuitive to instinctively use all functions without a system manual. Thus, in user testing, Usability, as well as characteristics such as Performance and Reliability, is an important test perspective. We release the Beta version to a restricted group of users to test based on the user's actual usage data and also receive feedback.

8.2.4. TEST CASES

Test cases are generated according to the characteristics of performance, acceptability, security,

maintenance and sustainability. Additionally, the test is conducted by generating a test case to verify the system's fulfillment of the required functions. Prepare an evaluation sheet to verify the results of the test.

9. Development Plan

9.1. Objectives

In this chapter, we introduce the technologies, language and environment for the development of the application and necessary for the development of the system.

9.2. Environment

9.2.1. Unity

[Figure 32] Unity

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, architectural visualization, and virtual reality (VR). One of Unity's advantages is the low license cost. This does not charge a separate license fee even if the game is released.

9.2.2. VRchat

[Figure 33] VRchat

VRChat is a large-scale multi-user online virtual reality social service. In this game, players can interact with other players implemented as 3D character models. The biggest feature of VRChat is that you can register your own avatar or world directly through Unity Engine.

9.2.3. Adobe Photoshop

[Figure 34] Adobe Photoshop

Adobe Photoshop is a graphical tool developed and published by Adobe that enables image synthesis, editing, and drawing. It is a raster graphics editor developed and published by Adobe Inc. for Windows and macOS. This program would provide aesthetical layout and icons for improved user experience during our project.

9.3. Version Control System

9.3.1. Git

[Figure 36] Git

Git is a version control system for tracking file changes and coordinating the operations of these files among multiple users. It features very fast speed and distributed storage support. It has advantages such as offline work through local storage, fast speed, easy and stable branch and merge, and free use. In our system, version management is done using Github server.

9.4. Constraints

The system is designed and implemented based on the content of the document. Other details shall be designed and implemented in the direction preferred by the developer, and the following restrictions shall be observed:

- Use technology that has already been widely demonstrated.
- Avoid using technology or software that requires a license or financial costs.
- Unauthorized parts should not be used arbitrarily because information on facilities in schools is used.

- When using external APIs, the relevant constraints and regulations should be observed.
- It should be designed and implemented considering the characteristics of the system's Performance, Acceptability, Security, Maintenance, and Sustainability.
- Refactoring source code should be executed frequently.
- Develop with Windows 10 environment.
- Optimize the source code to prevent waste of system resources.
- Decide in a more user-friendly and convenient direction.
- Use opensource software whenever possible.

9.5. Assumptions and Dependencies

All systems in this document are written on the assumption that they are designed and implemented based on Unity, VRchat and open source. Therefore, all contents are written based on the Unity with minimum API version 2020 and may not be applied to other operating systems or versions.

10.Supporting Information

10.1. 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).

10.2. Document History

Data	Version	Description	Writer
2021/10/27	0.1	Style and overview	Munsuk Jang
2021/10/28	1.0	Addition of 1, 2	Sanghun Kim
2021/10/29	2.0	Addition of 3	JiYoung Kim
2021/11/1	3.0	Addition of 4.1, 4.2	Jihye Park
2021/11/3	4.0	Addition of 5.1, 5.2	Seongbin Yoon
2021/11/4	5.0	Addition of 5.3	SeHyun Choi
2021/11/5	6.0	Addition of 6.1, 6.2, 6.3	Munsuk Jang
2021/11/7	7.0	Addition of 6.4, 6.5, 6.6	Jihye Park

2021/11/8	8.0	Addition of 6.7, 6.8, 6.9	Seongbin Yoon
2021/11/9	9.0	Addition of 7.1, 7.2	Ji Young Kim
2021/11/11	10.0	Addition of 7.3, 7.4	Sanghun Kim
2021/11/12	11.0	Addition of 8.1, 8.2	HaJeong Lim
2021/11/13	12.0	Addition of 9.1, 9.2	SeHyun Choi
2021/11/14	13.0	Addition of 9.3, 9.4	Munsuk Jang
2021/11/16	14.0	Addition of 9.5	Sanghun Kim
2021/11/17	15.0	Addition of 10	Seongbin Yoon
2021/11/18	16.0	Revision of 1, 2	Hajeong Lim
2021/11/19	17.0	Revision of 3, 4	JiHye Park
2021/11/20	18.0	Revision of 5, 6	Ji Young Kim
2021/11/21	19.0	Revision of 7, 8, 9	SeHyun Choi
2021/11/21	20.0	Revision of Structure	HaJeong Lim

10.3. List of Tables

10.4. List of Figures