Exhibition for Graduation Project

Software Design Specification

2021.11.21

Introduction to Software Engineering
TEAM 12
Team Leader 박다영
Team Member 권윤영
Team Member 김정재
Team Member 안진형
Team Member 왕언문
Team Member 이동혁
Team Member 이하은

Table of Contents

1. Preface

1.1 Readership

This document is written for software engineers, who directly develop the system, and architecture, designing the system, and all stakeholders, who participate in system development.

1.2 Document Structure

A. Preface

Preface defines expected readers and introduces he overall structure with the purpose of each table of contents.

B. Introduction

Introduction describes all types of diagram and tools which are used to explain the system design in this document.

C. System Architecture

System Architecture describes the overall contents of the system to be developed through this project. It outlines the relationship between the system and each sub systems, and describes how they are assigned to actual hardware.

D. Protocol Design

APIs are defined through protocol design. When defining the APIs, set the appropriate method and endpoint, and define the request and response that meet the requirements. At this time, the goal is to define the value required for request and the value required for response up to variable type.

E. Database Design

Database design is the part of details about the database of our application. The relationship between each entity and entity is expressed through ER diagram, and Relational Schema is created using it.

F. Testing Plan

Testing plan includes testing policy, where how test will be performed while development and after application released will be described.

G. Development Plan

Development plan is about workflow dates and introduces tools that we will use in the development part. Also, version management tool is included.

2. Introduction

2.1 Objectives

Objectives of introduction is giving readers explanation of diagrams and tools for design system and introduce the scope of our project.

2.2 Applied Diagram

A. 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. UML provides different modeling techniques and a handful subset of diagrams, it is efficiently used to provide means of communication between developers and users as it covers wide range of symbols and definitions.

B. Class Diagram

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

C. Sequence Diagram

Sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used to understand the requirements and to document the existing process.

D. ER Diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how entities such as people, objects or concepts relate to each other within a system. These are used to design relational databases in the fields of software engineering, business information systems, education, and research.

2.3 Applied Tool

A. Draw.io

Draw.io is proprietary software for making diagrams and charts. It is web-based modeling tool providing many templates and shapes that user can draw any kind of diagram easily. With draw.io, user does not need to redraw its shape for drawing diagram, especially for UML diagrams.

2.4 Project Scope

The purpose of this project is providing graduation exhibition on a virtual environment. Since COVID 19, it became difficult to hold an exhibition of graduation works. By our project, graduation exhibition could be held online, without any time or space constraints. Our project covers several functions for the user. First of all, as the user logs in with their own ID and password, the user will be defined either visitor or exhibitor. As an exhibitor, the user can exhibit their own work on the virtual environment, and also add some extra information about their works. Also, if a visitor tries to communicate, the exhibitor can participate the communication. As a visitor, the user can view the exhibited graduation works and press like buttons if they want to express their enjoyment. Also, the visitor can ask the exhibitor to communicate. The rank of likes will be exhibited individually at the front of the exhibition.

3. System Architecture

3.1 Objectives

This section will explain the structure of this system. The relationship between the overall system and the subsystem will also be explained.

3.2 System Organization

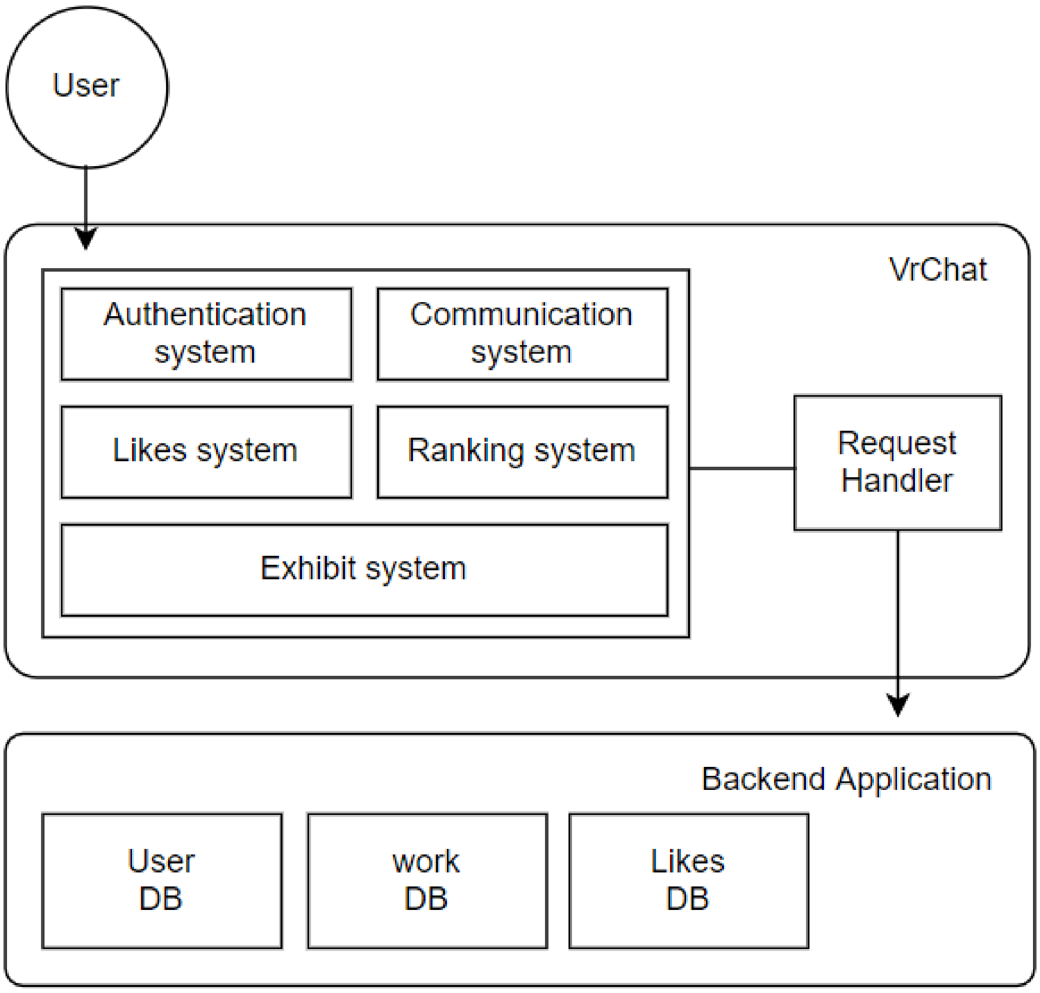


Figure 1. Overall architecture

VRChat is used frontend application framework. Each component requests data by using request handler. Request handler sends request message to backend application which contains user DB, work DB, and Likes DB.

3.3 Authentication System

3.3.1 Class Diagram

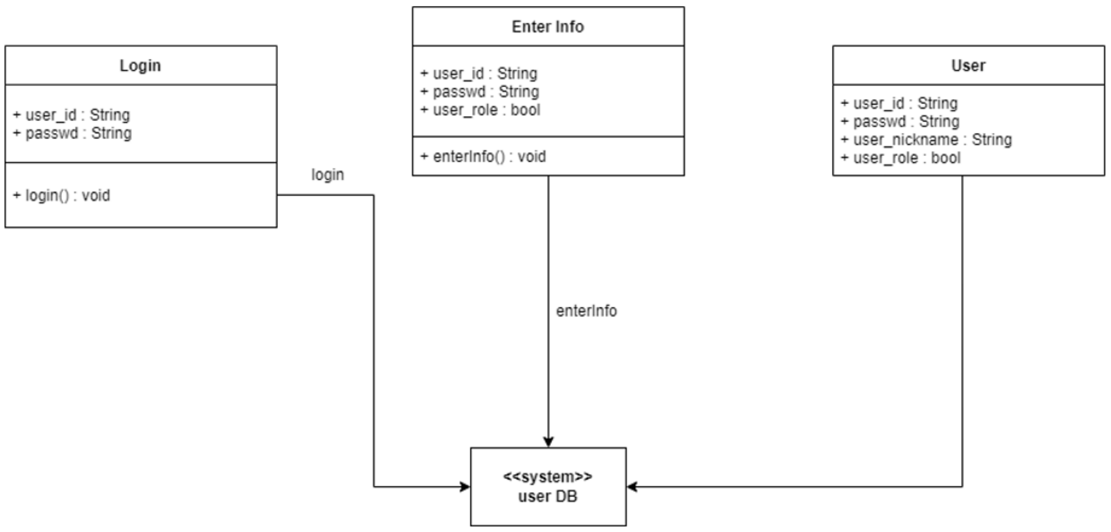


Figure 2. Authentication System – Class Diagram

A. Login

(1) Attributes

+user_id : user ID(email address)

+pw : user password

(2) Methods

+login() : check if login process is successful and finish login process. If user id doesn't exist in database, automatically sign up the user.

B. Enter information

(1) Attributes

+user_id : user ID(email address)

+user_nickname : user nickname

+user_role : exhibitor(true), visitor(false)

(2) Methods

+enterInfo() : get user's information after login and update DB

C. User

(1) Attributes

+user_id : user ID(email address)

+pw : user password

+user_nickname : user nickname

+user_role : exhibitor(true), visitor(false)

3.3.2 Sequence Diagram

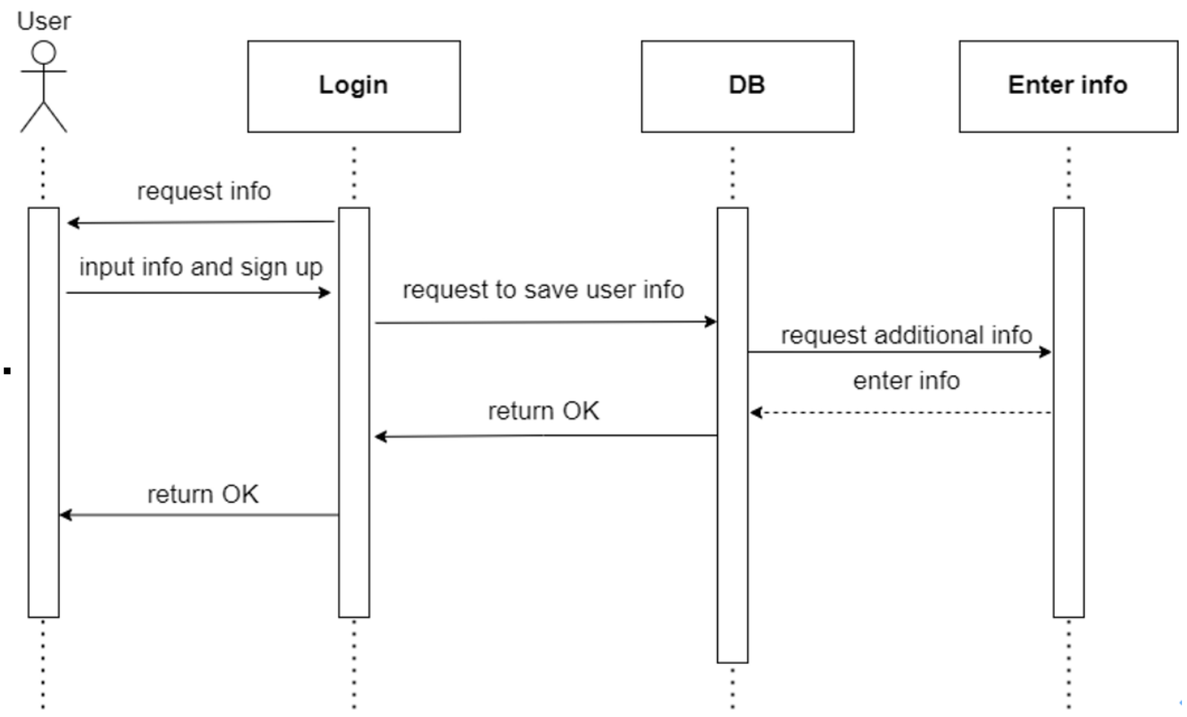


Figure 3. Authentication System – Sequence Diagram

3.4 Communication System

3.4.1 Sequence Diagram

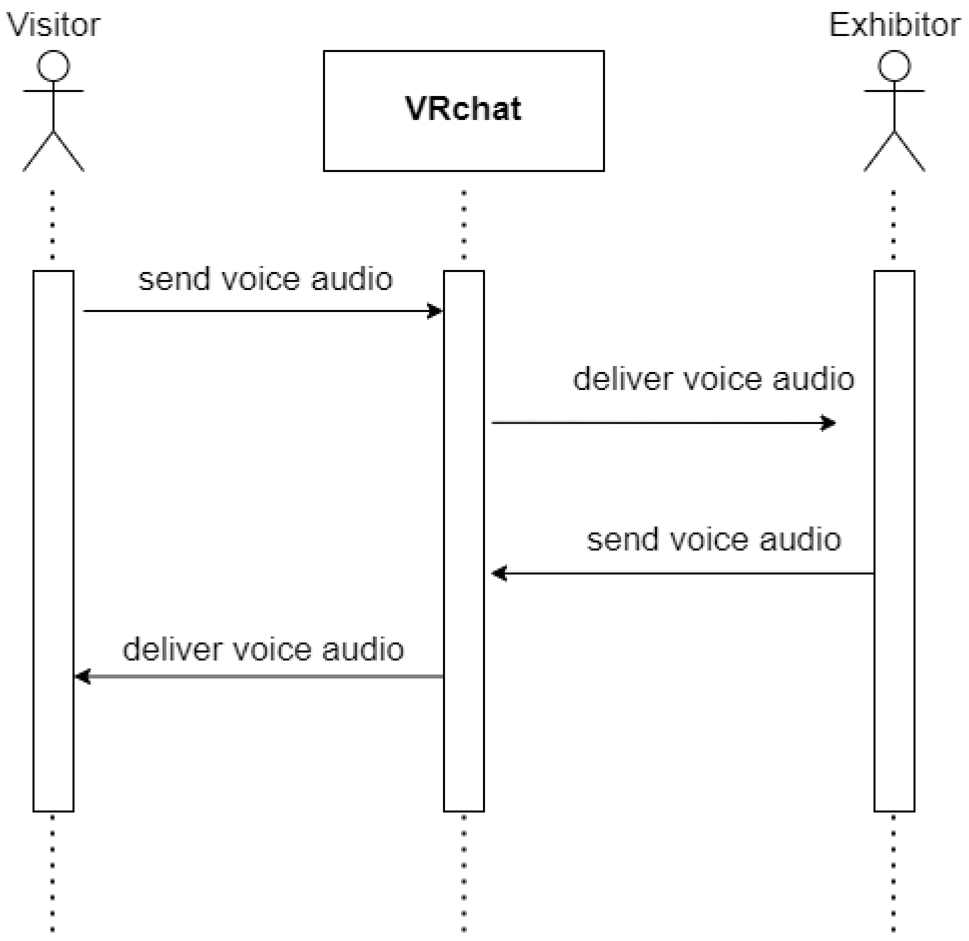


Figure 4. Communication System – Sequence Diagram

3.5 Exhibit System

3.5.1 Class Diagram

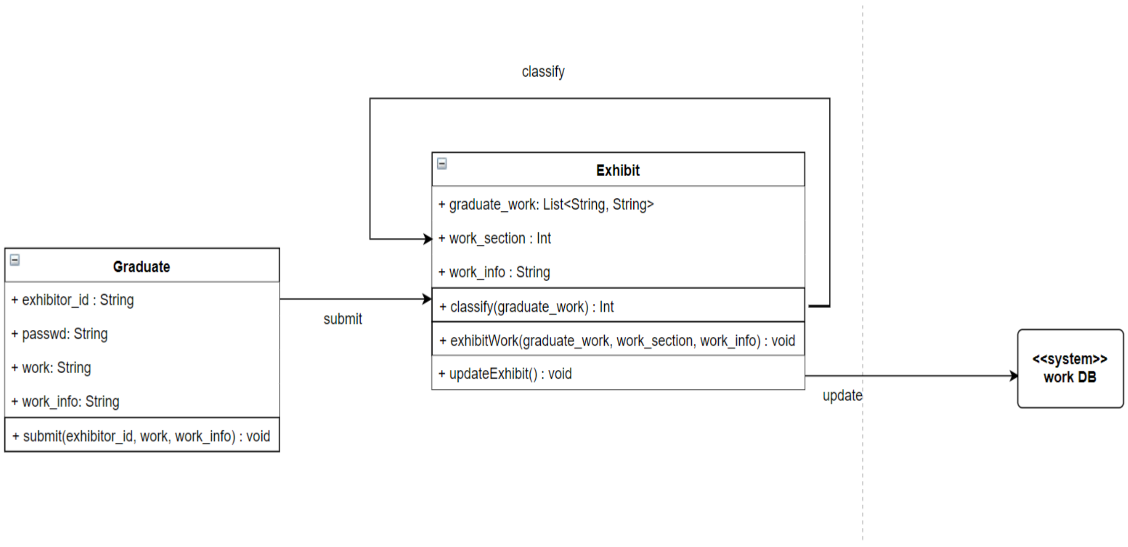


Figure 5. Exhibit System – Class Diagram

A. Graduate

(1) Attributes

+exhibitor_id : user(graduate) ID(email address)

+pw : user(graduate) password

+work : senior project

+work_info : information for senior project

(2) Methods

+submit(exhibitor_id, work, work_info) : graduates press the 'Submit' button to submit their graduation work and related explanations.

B. Exhibit

(1) Attributes

+graduation_work : variables with the ID of the graduate and the name of the graduation work

+work_section : the field related to graduation work

+work_info : information for senior project

(2) Methods

+classify(graduate_work) : the field related to the work is determined and stored in work_section as an int-type variable

+exhibitWork(graduate_work, work_section, work_info) : the graduation work is displayed using the x and y coordinates of the empty space in a place suitable for the work section

+updateExhibit() : update the work on workDB

3.5.2 Sequence Diagram

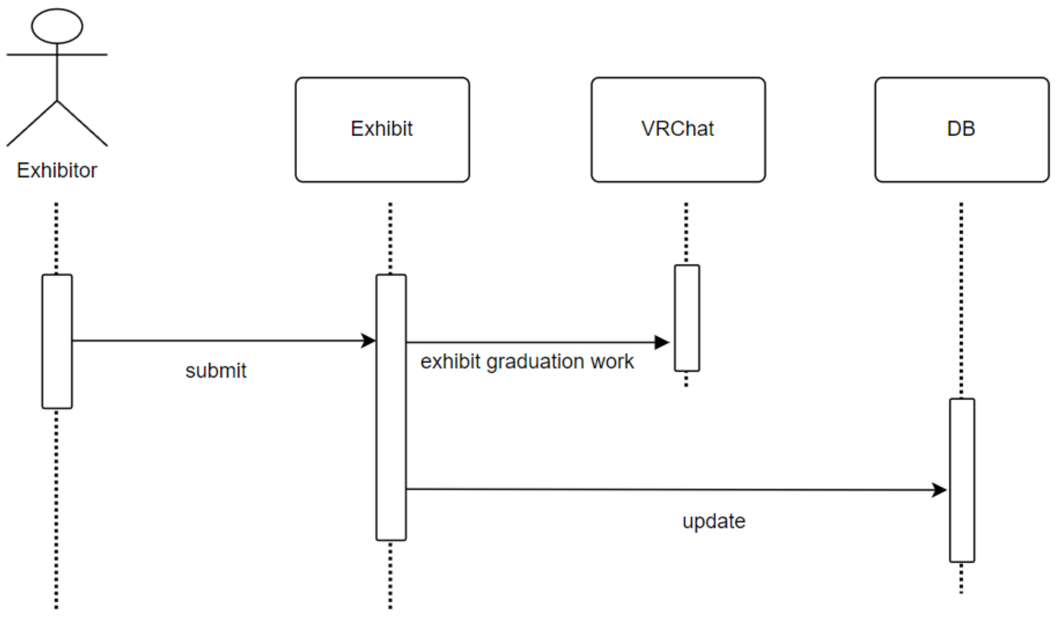


Figure 6. Exhibit System – Sequence Diagram

3.6 Express Likes & Ranking System

3.6.1 Class Diagram

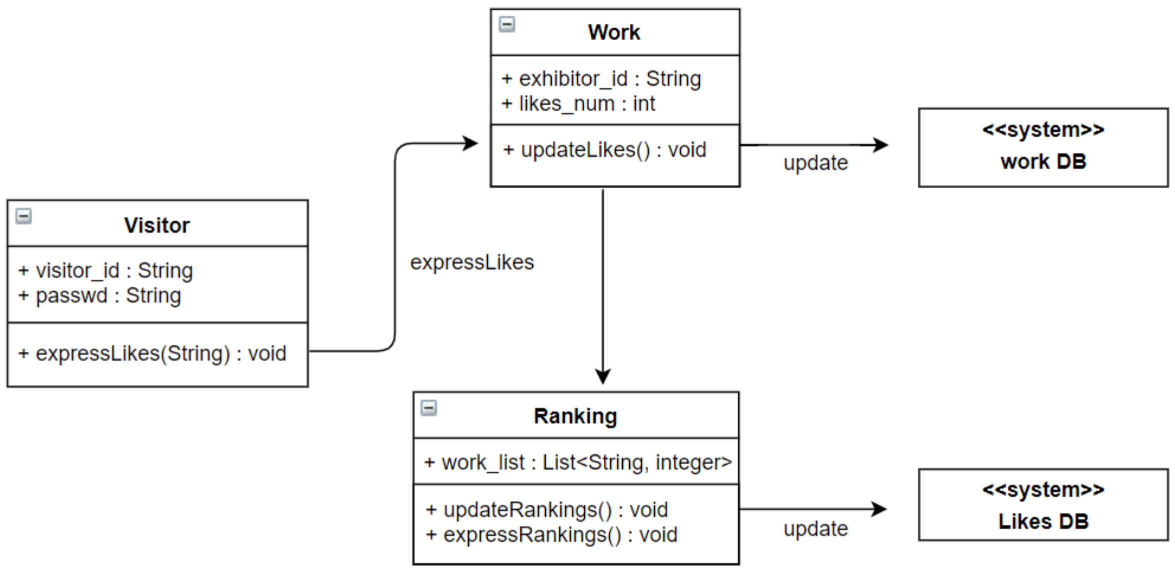


Figure 7. Express Likes/Ranking System – Class Diagram

A. Visitor

(1) Attributes

+visitor_id : user ID(email address)

+pw : user password

(2) Methods

+expressLikes(String) : user express likes to their favorite work (increase likes_num of that work) by clicking 'Like' button on that work

B. Work

(1) Attributes

+exhibitor_id : exhibitor ID(email address)

+likes_num : number of likes of that work

(2) Methods

+updateLikes() : update likes_num to work DB

C. Ranking

(1) Attributes

+work_list : list of exhibition works sorted by the number of likes (String : exhibitor_id, integer : likes_num)

(2) Methods

+updateRankings() : when new likes are included, update ranking which is sorted by the number of likes, and update new ranking to Likes DB

+expressRanking() : A board screen shows 'Like' ranking and scores of graduation works at the entrance of the exhibition

3.6.2 Sequence Diagram

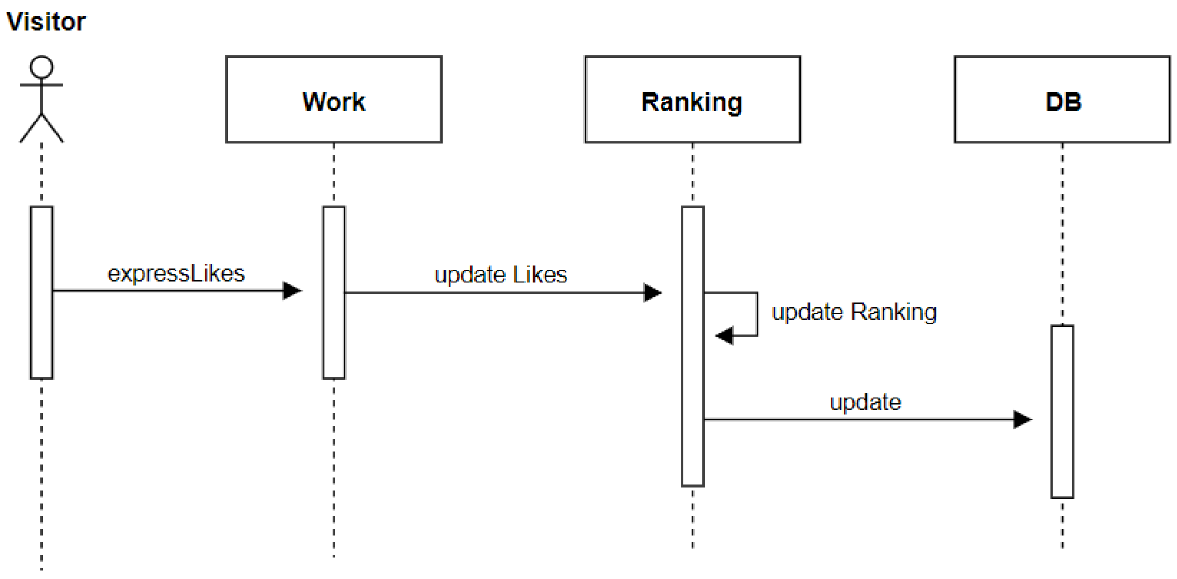


Figure 8. Express Likes/Ranking System – Sequence Diagram

4. Protocol Design

4.1 Objectives

At protocol design, there's description how communication is performed between subsystems. Protocol's basic format is JSON. Also, consider each function as an API and try to make it RESTful.

4.2 JSON

JavaScript Object Notation (JSON) is a human-readable text format to convey data objects that consist of attribute-value pairs and array data types (or any other serializable value) or "key-value pairs. It is an open standard format used.

Originally derived from the JavaScript language, it follows the syntax format of JavaScript, but is a language-independent data format, in other words, since it is independent of the programming language or platform, the code for parsing and generating JSON data can be easily used in numerous programming languages such as C, C++, C#, Java, JavaScript, Perl, and Python.

4.3 RESTful API

An API, or application programming interface, is a set of rules that define how applications or devices can connect and communicate with each other. REST APIs are APIs that adhere to the design principles of the REST (Representational State Transfer) architectural style. For this reason, REST APIs are often referred to as RESTful APIs.

4.4 Protocol Description

4.4.1 User Authentication (Sign-up)

Method : POST

Endpoint : /user

-Request

Attribute	Type	Value
u_id	String	Student ID
email	String	User’s e-mail
password	String	User’s password

Table 1. Sign-up-request

-Response

Attribute	Type	Value
Status	Boolean	Success or fail

Table 2. Sign-up-response

4.4.2 User Authentication (Log-in)

Method : POST

Endpoint : /user

-Request

Attribute	Type	Value
email	String	User’s e-mail
password	String	User’s password

Table 3. Log-in-request

-Response

Attribute	**Type**	**Value**
status	Boolean	Success or fail
user	Json	User’s information

Attribute	Type	Value
u_id	String	Student ID
email	String	User’s e-mail
role	Boolean	True: exhibitor, False: visitor
name	String	User’s name

Table 4. Log-in-response

4.4.3 User Authentication (Log-out)

Method : GET

Endpoint : /user/logout

-Request

Attribute	**Type**	**Value**
e-mail	String	Student ID
password	String	User’s password

Table 5. Log-out-request

-Response

Attribute	**Type**	**Value**
status	Boolean	Success or fail

Table 6. Log-out-response

4.4.4 Enter the user’s information

Method : POST

Endpoint : /user/detail

-Request

Attribute	Type	Value
u_id	String	Student ID
role	Boolean	True: exhibitor, False: visitor
name	String	User's name

Table 7. Enter the user's information-request

-Response

Attribute	Type	Value
status	Boolean	Success or fail

Table 8. Enter the user's information-response

4.4.5 Enter the graduation work's information

Method : POST

Endpoint : /work

-Request

Attribute	Type	Value
u_id	String	Student ID
position_x	Float	Position value (x-axis)
position_y	Float	Position value (y-axis)
file_addr	Float	the graduation work file's address
title	String	Title of the graduation work
description	String	Description of the graduation work

Table 9. Enter the graduation work's information-request

-Response

Attribute	Type	Value
status	Boolean	Success or fail

Table 10. Enter the graduation work's information-response

4.4.6 Get the graduation work's information

Method : GET

Endpoing : /work

-Request

Attribute	Type	Value
u_id	String	Student ID

Table 11. Get the graduation work's information - request

-Response

Attribute	Type	Value
w_id	Int	Work's ID (idx)
u_id	String	Student ID
position_x	Float	Position value (x-axis)
position_y	Float	Position value (y-axis)
file_address	Float	the graduation work file's address
title	String	Title of the graduation work
description	String	Description of the graduation work
like_score	Int	"Like" score of the graduation work

Table 12. Get the graduation work's information - response

4.4.7 Update "Like"

Method : PUT

Endpoint : /like

-Request

Attribute	Type	Value
u_id	String	Student ID

Table 13. Update "Like" - request

-Response

Attribute	**Type**	**Value**
u_id	String	Student ID
like_score	Int	Updated “Like” score of the graduation work

Table 14. Update “Like” - response

4.4.8 Get “Like” scores

Method : GET

Endpoint : /like/list

-Request

None

-Response

Attribute	**Type**	**Value**		
List	List<>	List of ID, title and “Like” scores		
		Attribute	Type	Value
		w_id	Int	Work ID (idx)
		u_id	String	Student ID
		name	String	User’s name
		title	string	Title of the graduation work
		like_score	Int	“Like” score of the graduation work

Table 15. Get “Like” scores response

5. Database Design

5.1 Objectives

Database design was created based on the system architecture described in requirement specification. The relationship between each entity and entity is expressed through the ER Diagram, and a Relational Schema is created using it.

5.2 ER Diagram

There are three Entities in the system : Account, Work, and Like. Each Entity is expressed in the form of a rectangular box, and the relationship between Entities is expressed in a diamond shape. The attributes of each entity are expressed in an oval shape, and the primary key is underlined on the label.

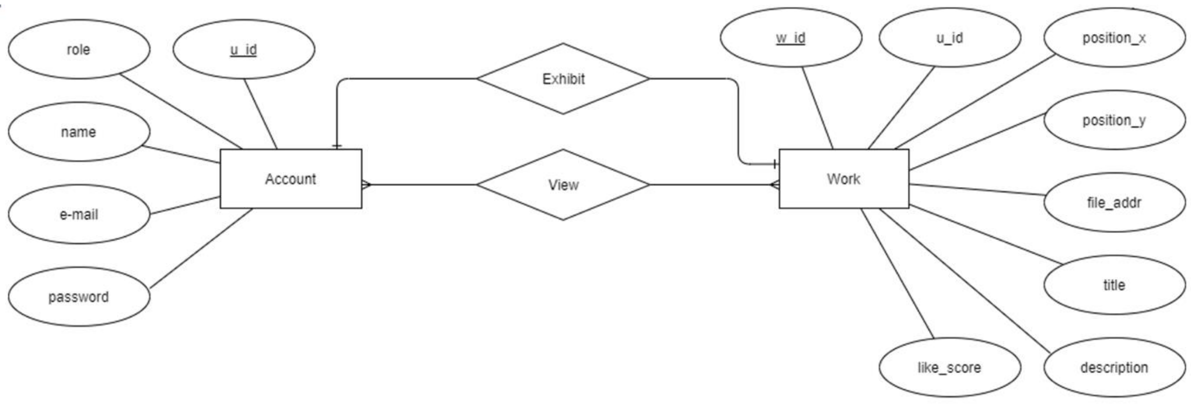


Figure 9. Database Design – ER Diagram

5.3 Entities

5.3.1 Account

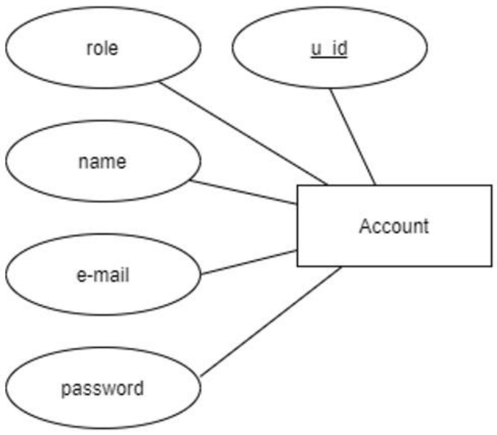


Figure 10. ER Diagram – Account

Account entity includes the information about user such as role, name, e-mail, password, and primary key is u_id.

5.3.2 Work

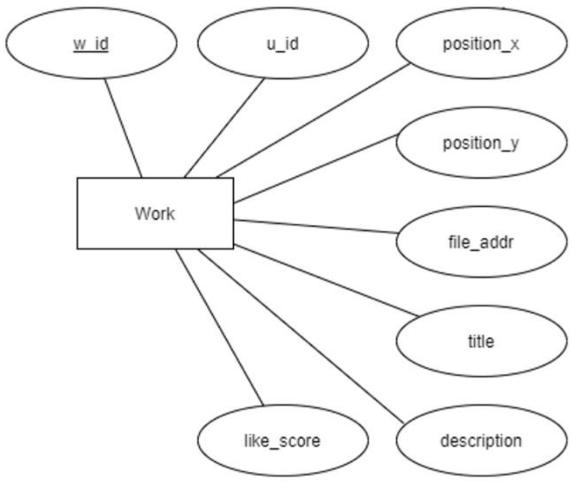


Figure 11. ER Diagram – Work

Work entity includes the information about user's graduation work such as position, address of work file, title, description, score of like and primary key is w_id. u_id is foreign key referring to account entity.

5.4 Relational Schema

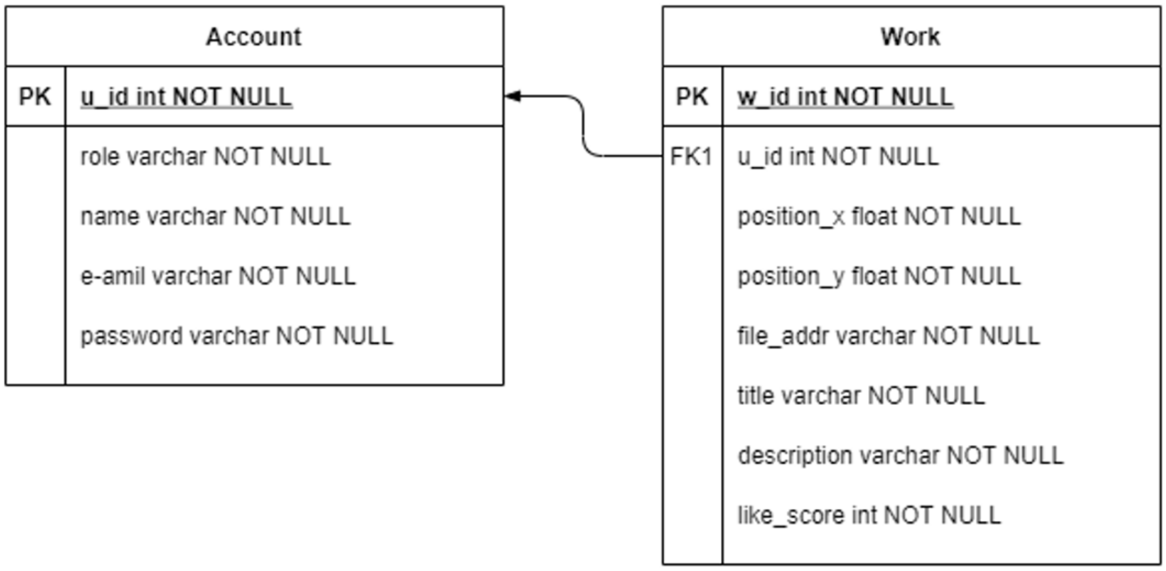


Figure 12. Database Design – Relational Schema

6. Testing Plan

6.1 Objectives

This chapter describes how to test and check the system. To determine whether the system is performing as expected so that defects can be identified and analyzed after the system is completed.

6.2 Testing Policy

6.2.1 Develop Testing

The purpose of development testing is to prevent defects and detect strategies to reduce software development risk.

6.2.1.1 Development test includes 4 testing processes. Component testing, integration testing, system testing, and acceptance testing.

(1) Component Testing

Component testing independently verifies the functionality of each component. Component may be functions, objects or coherent groupings of these entities.

(2) Integration Testing

Integration testing is performed to ensure that the system is functioning properly while integrating the subsystems.

(3) System Testing

System testing tests the system as a whole to check whether the entire system is functioning properly.

(4) Acceptance Testing

Acceptance testing users customer's data to check whether the system meets the customer's needs.

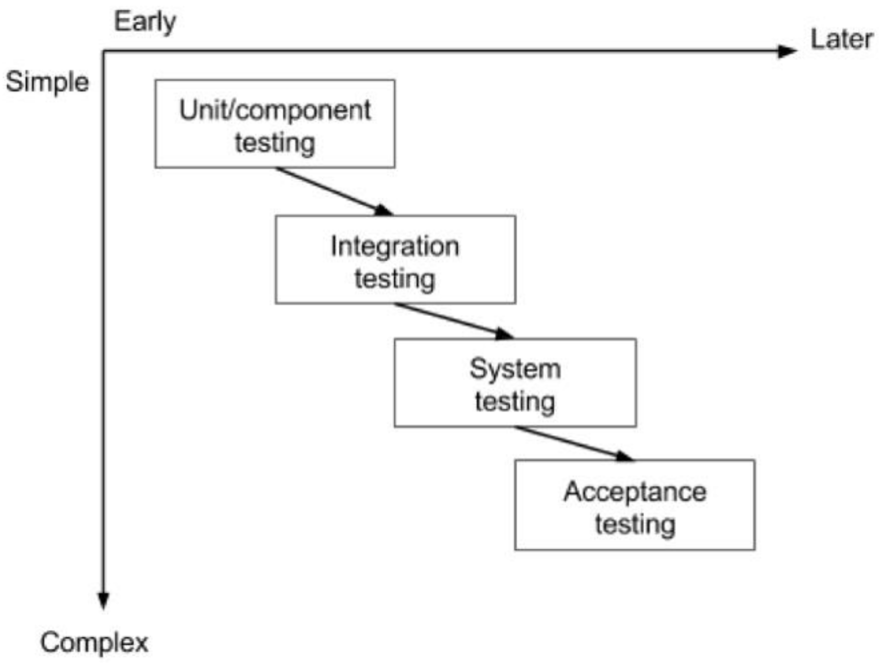


Figure 13. Develop Testing

6.2.1.2 Security

Protection of SQL injection

SQL injection is when an attacker executes arbitrary SQL code in the primary database. Such as attack could reveal users' personal information or d elete records from a database.

During testing, we should protect the generated SQL from potential da tabase attackers. However, when using raw queries or private SQL, we should test for the presence of user-accessible parameters.

6.2.2 Release Testing

Release testing the latest version to test the full functionality of the software. Check for any problems or errors to make sure they are perfect.

⚠ : Must be completed before release.

6.2.3 User Testing

(1) Usability Testing

Usability tests check whether the user and participant interfaces are easy to use and understand.

(2) Closed Testing

During the beta phase we only released it to a limited number of people. We need users to give us feedback on improvements and deficiencies through real use.

6.2.4 Test cases

Use holistic testing from a basic user management and service usage perspective.

7. Development Plan

7.1 Objectives

This section describes the programming language and the IDE that is going to be used.

7.2 Front/back-end Programming Language & IDE

7.2.1 Front-end

Front-end and 3D space that users will use is developed by Unity.

A. Tool: Unity

B. Sdk : VR CHAT

VR CHAT is a free massively multiplayer online virtual reality. Through this we can make an online 3D graduation exhibition platform. People in front of the computer screen can be immersive feeling.

C. C#

C# is a precise, simple, type-safe, object-oriented language.

C# features: 1. Fully object-oriented 2. Distributed support 3. Portability 4. High performance, etc. The features of C# fit our project perfectly.

D. 3D Object Source

3D Object Source and UI Source are self-made or purchased from Unity Asset Store, an open market provided by Unity.

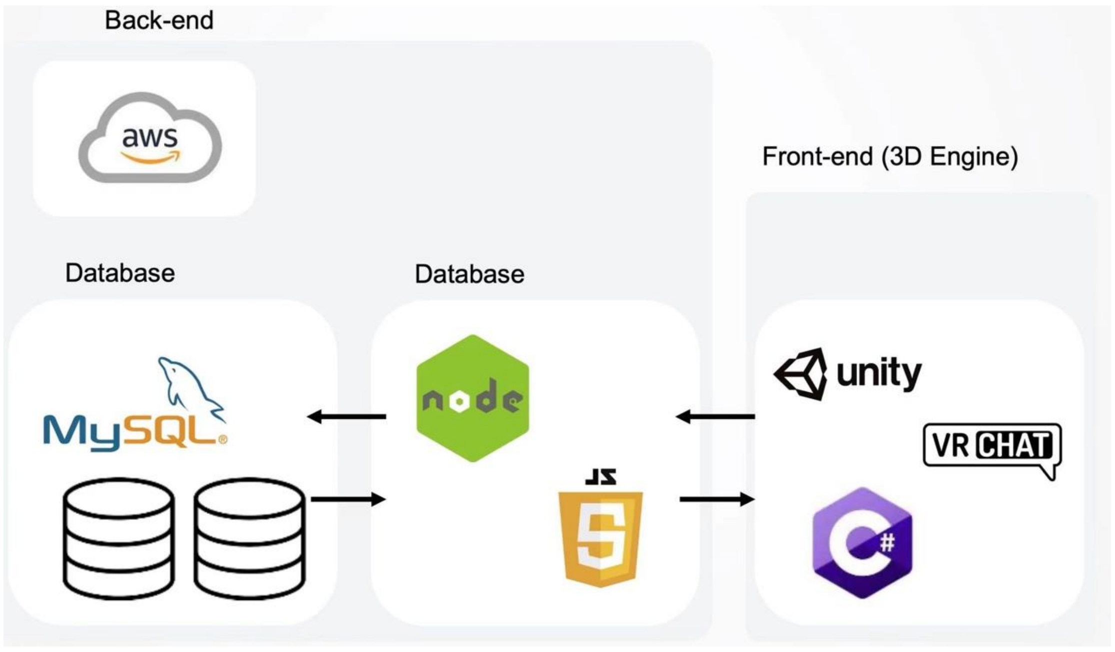
7.2.2 Back-end

We use AWS (Amazon Web Service) provided by Amazon as a server that communicates with the frontend and stores the DB.

Because it is a small project, DB server and web server are created on one server. DB is made of mysql and App server is made of Node.js Framework.

A. Server : AWS
B. Environment : Node.js
C. Language : Javascript
D. Database : MySQL

7.3 Architecture



The user needs to register for the first time. Fill in the personal registration i nformation in the front end, and the front end will send the user's personal informatio n to the server database. After confirming the information, the server will send it to t he MySQL database for saving. Convenient for future use.

The front-end page collects the personal information entered by the user and s ends it to the server. The server compares the personal information of the user throug h the MySQL database. If it is already in use, the user information will be returned l ayer by layer, inform the front end, users quickly log in.

7.4 Workflow Dates

Proposal 9/26

Requirement specification 10/31

Design specification 11/21

Implementation 12/10

8. Index

8.1 Figure

8.2 Table