CSCI3100 Software Engineering

High-Level Design Documentation

Group B4

CU Simulator

Chim Ka Chun	1155126571
Ho Cheuk Hin	1155125257
Ku Nok Tik	1155143829
Au Tsz Nga	1155127047
Wong Yi Oi	1155127334

Version: 1.0 Print Date: 17/2/2022

Department of Computer Science and Engineering, The Chinese University of Hong Kong

Table of contents

1. Introduction

1.1 Project Overview

In this project, we will create a game called “CU Simulator”, which is a SLG (Strategy Game) with background based on the campus of The Chinese University of Hong Kong (CUHK). In this game, players can create a character and go through different events. User statistics are altered by different choices in the events and various kinds of endings can be reached. Real-time action functions such as friend messaging and friend aiding which helps the players’ game progress, together with a friend recommendation system, are also provided.

1.2 Objectives

1. To raise freshmen’s interest in real university life and familiarize them with CU’s atmosphere
2. To connect alumni of CUHK inside and outside the game

Our target is that our system is able to run smoothly with about 50% of Undergraduate year 1/2 students possessing a game account and finishing at least one game play, which is about 4000 people. Approximating the number of alumni by doubling the UG students, about 8000 people are to be accommodated.

1.3 System Features

There mainly 5 features in our design in order to achieve our objectives:

Interactive Friend System

Apart from usual functionalities such as messaging and viewing friends' status, we also provide a 'friend aiding' function. Users can send help to their friends every certain period of time, which can help the game process of their friends. This can improve interactions and encourage making friends with each other via the game.

Unique Gameplay Experience

Different endings can be reached based on the user statistics and choices they have made during gameplay. The allows the users to develop their unique character and reach different endings.

Informative Story Plot

In our game plot, we would cover different information and culture of CUHK in order to achieve our first objective – to raise freshmen's interest in real university life and allow them to familiarize with CUHK.

Achievements List

In order to enhance playability of the game, we would introduce an achievement list, which encourage users to have multiple game experience. Upon achieving different achievements, users can also gain a better initial statistic of the avatar, which allow the users to reach different endings easily and they could therefore explore the game more deeply.

Minigames

Different events would occur during gameplay, which would be boring if they were just plain text. We would therefore introduce some minigames. Winning and losing the minigames will affect the game process, and the users may reach a totally different ending if they win/lose the game.

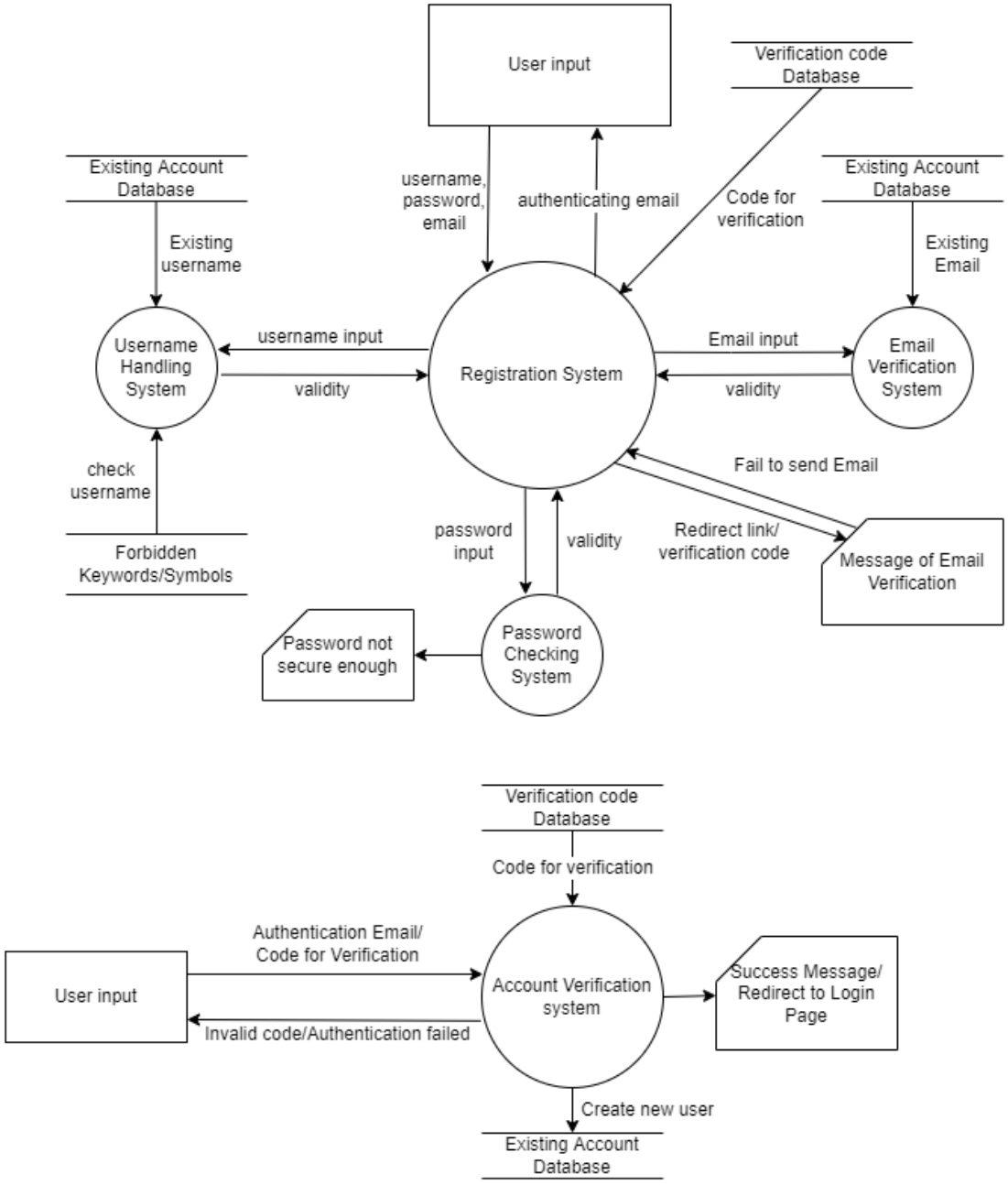
2. Background

Our team observed that due to online lessons, freshmen are not familiar with cultures, traditions, common languages or even places in CUHK. In order to raise their interest in real university life and familiarize them with the university's atmosphere, we create this education simulation game. Through traveling to different places in the university virtually and talking to NPC, or even their friends through messaging systems, we hope they can know more about their own university.

In addition, through this game, we hope to get CUHK alumni connected. By the real-time action functions provided in the game, alumni can meet friends and recall their precious memories in CUHK. They can also try to discover different endings or do things that they haven't done while they were university students. Some questions require them to go back to CUHK directly in order to get the correct answer, which encourages offline interactions of alumni.

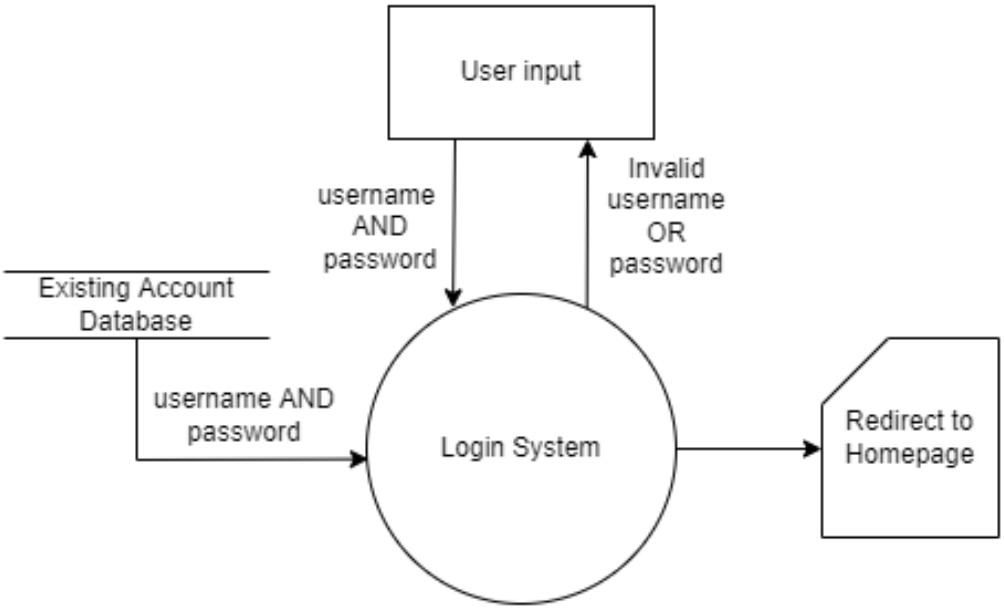
3. Specification

3.1 Registration and Verification System



A new user can register by clicking the registration button in the login page. They are required to input email, username and password (which need to be validated). Then a verification email will be sent to the user (which includes a random generated code) and the user needs to input the code to the website in order to finish registration.

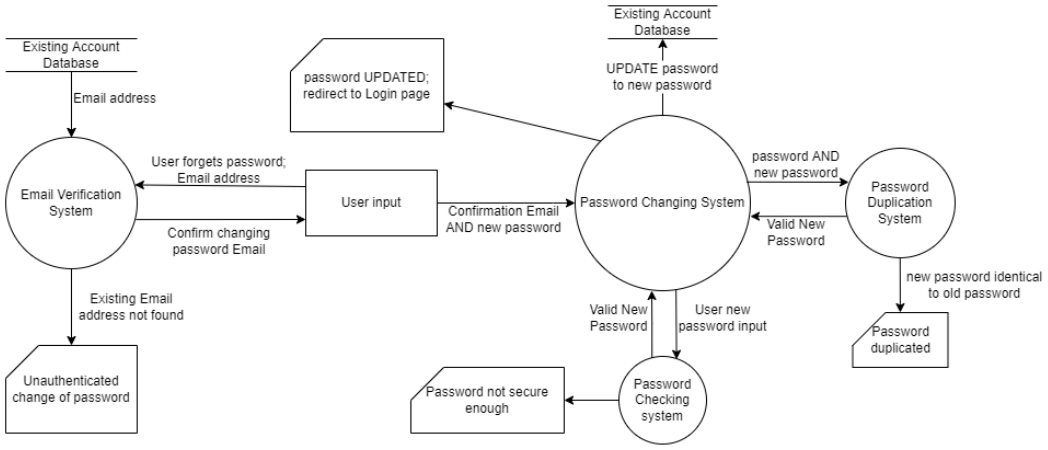
3.2 Login System



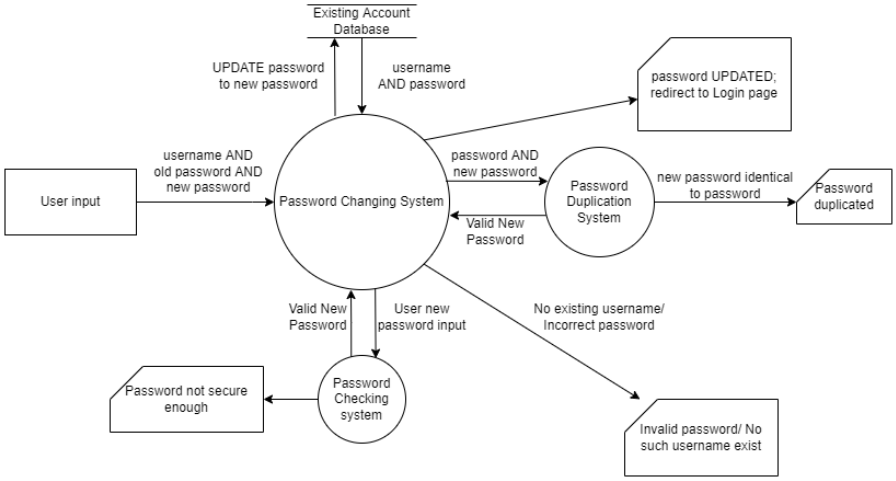
After registration, the user can login using username and password. If the user login successfully, the system will show the homepage to the user. Otherwise, the system will show a notification of wrong username or password.

3.3 Password Changing System

User forgets password:



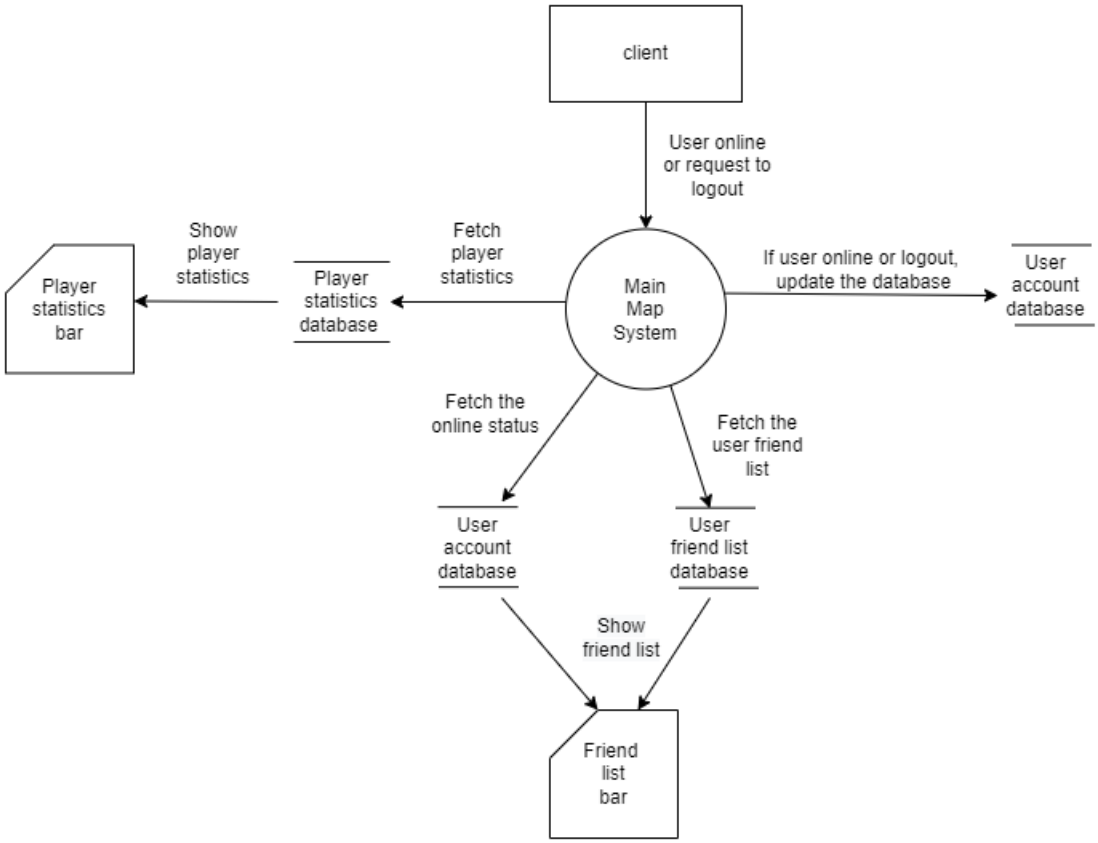
User requires password changing:



For a logged in account, the user can change the password only if they can enter their username and original password correctly after validation. They can then enter their new password.

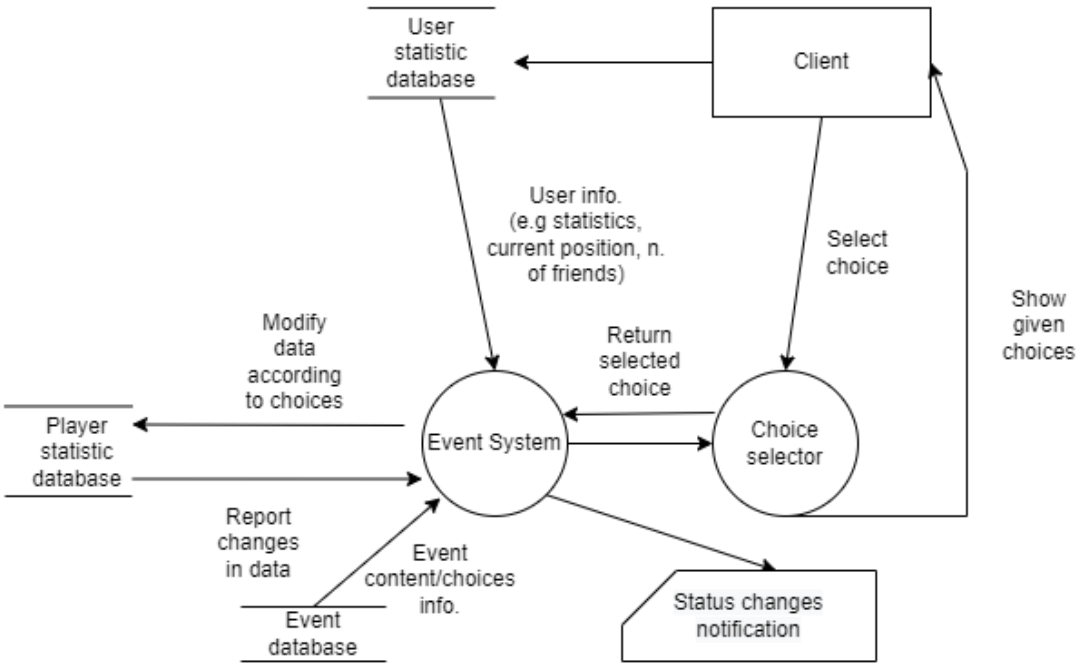
If the user forgets the password, they may press the “Forget password?” button. The user is required to enter their registered email. If the email was registered, a verification email will be sent and inquires the user to activate the password changing progress. The user will then run through the same process as the change password option.

3.4 Main Map (Homepage)



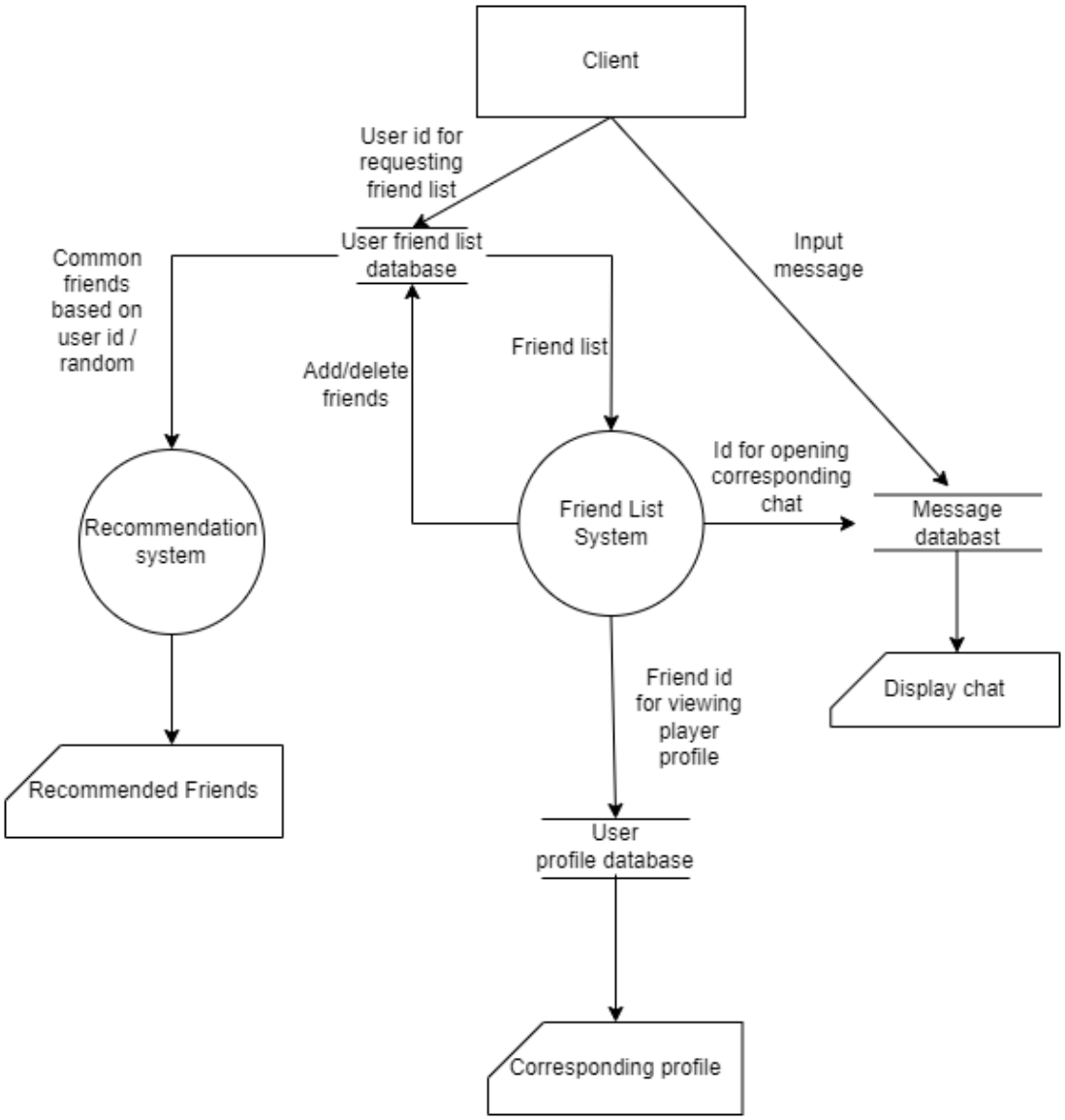
The main map is the main interface of the game. It serves as a bridge between most of the functions. The interface consists of several buttons to trigger all systems in the game, such as the statistics system, event system, friend list system and profile system. Therefore, through the main map, users can trigger the functions such as show profile, show friend list, and start the event. Also, the statistics and friend list of players are shown on the main map, players can view their statistics and the online status of their friends.

3.5 Events System



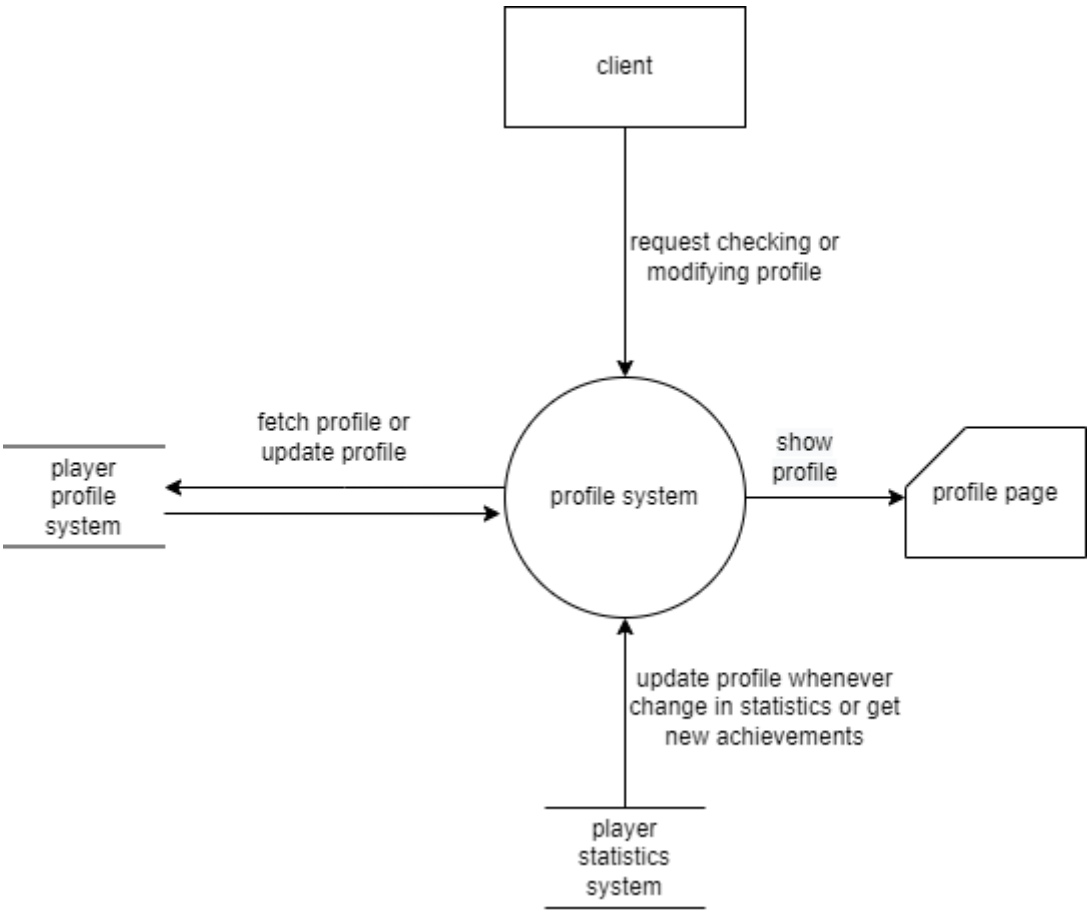
The player needs to trigger events to progress in the game. In general, the event will take input from the statistics part to determine the possible outcome. For example, the outcome may depend on the income of the player. Also, the result may alter the statistics of a player, such as gaining income or getting items. However, some events may also take input from the real-time system, such as the friend support function.

3.6 Friend List system



The players can search friends by using ID or player name. They can add or delete friends to the friend list. Also, the system can recommend some other players to the player.

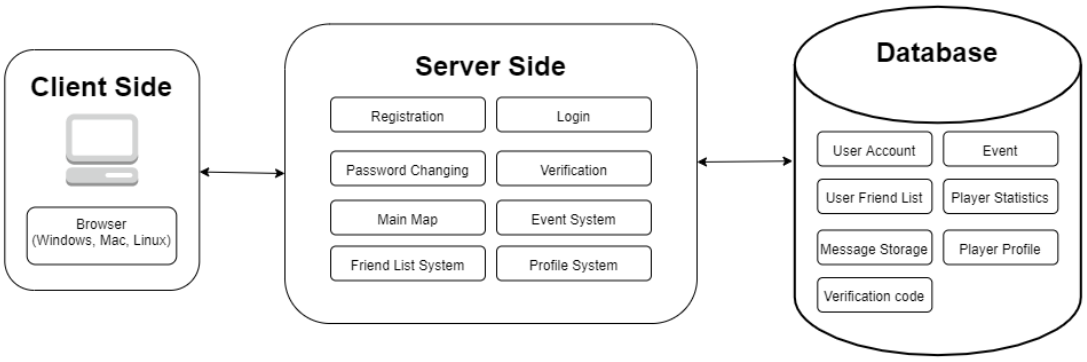
3.7 Profile System



The profile system is used to store the player's own information. The profile shows the profile photo, statistics and achievement of the player, etc. The player can also modify part of his own profile, including adding personal messages or choosing to show or hide their profile photos.

4. System Architecture Design

4.1 Architecture Diagram



This diagram shows the overall architecture of our software application. For the client side, users can request content from the server. Our system sends the request and displays the data to the client on the browser. For the server side, it supports several functions for the users. Users can register and verify their accounts first. After registering successfully, users can login their account and start playing the game. Users can complete their event tasks, view the statistics and profile, modify the friend list, change the password, help their friend's game progress and message others. At the same time, our system stores the user data such as user account, user-friend list, message, event progress, player statistics and profile into the database.

4.2 System Components

In our software, we plan to use JavaScript as our primary programming language and MERN stack as our web development framework, MERN stands for MongoDB, ExpressJS and ReactJS and NodeJS.

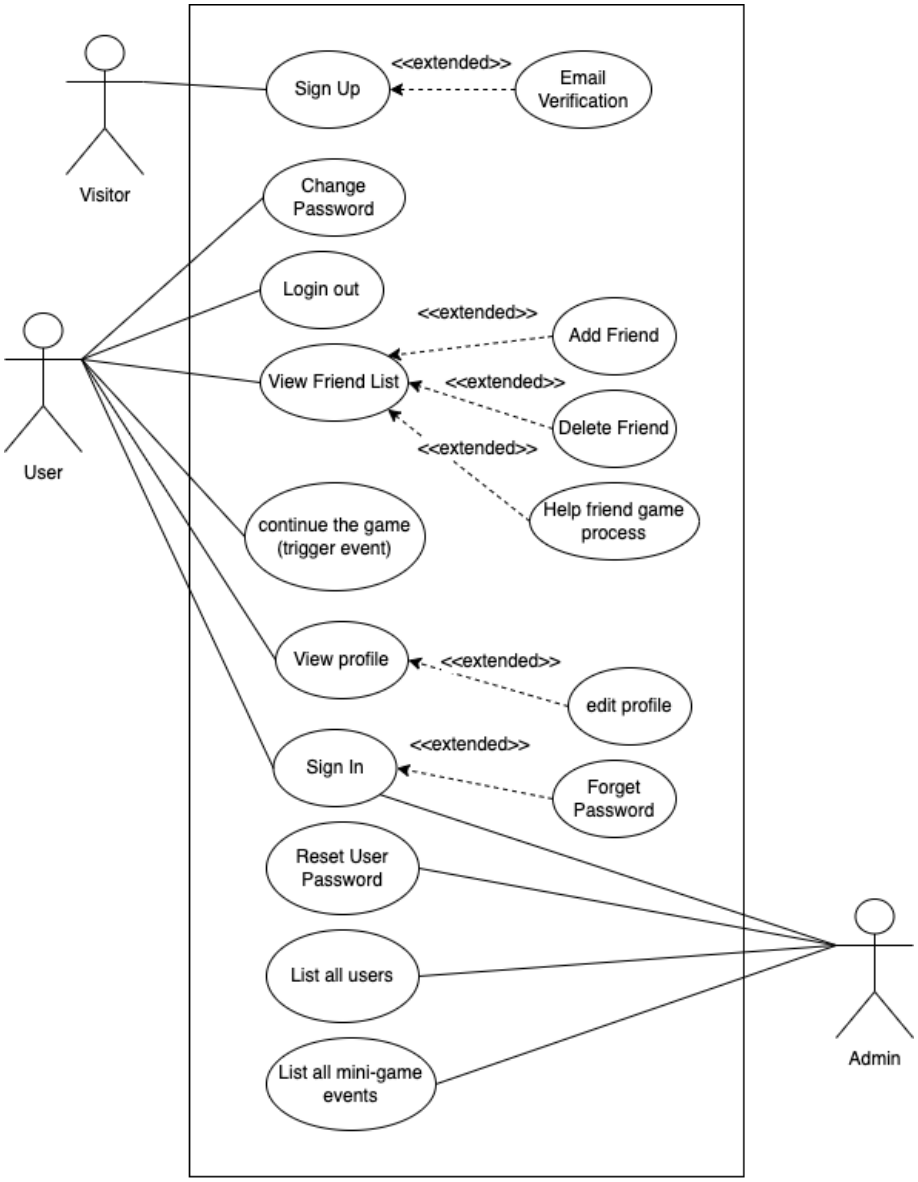
With regards to the frontend, we will use ReactJS, bootstrap. ReactJS is used to build the user interface of our website. It can create reusable UI components and help us to modularize our frontend code. Moreover, React uses virtual DOM to update our website. It can rerender the updated part of our website. Also, we will use bootstrap to design and decorate our user interface. For the graphics of the game, we will use React Konva and ThreeJS to create and display some 2D and 3D graphics respectively.

For middleware, we will use NodeJS and ExpressJS to set up middleware and respond to HTTP requests. The middleware can help us to fetch data from the database and send the user input data to the database.

In terms of the backend, we will use MongoDB to store data. The data in MongoDB is stored in JSON format which is easy for us to manipulate using JavaScript. Also, no redefined schema is needed for MongoDB. Therefore, we can expand and modify our tables and data easily.

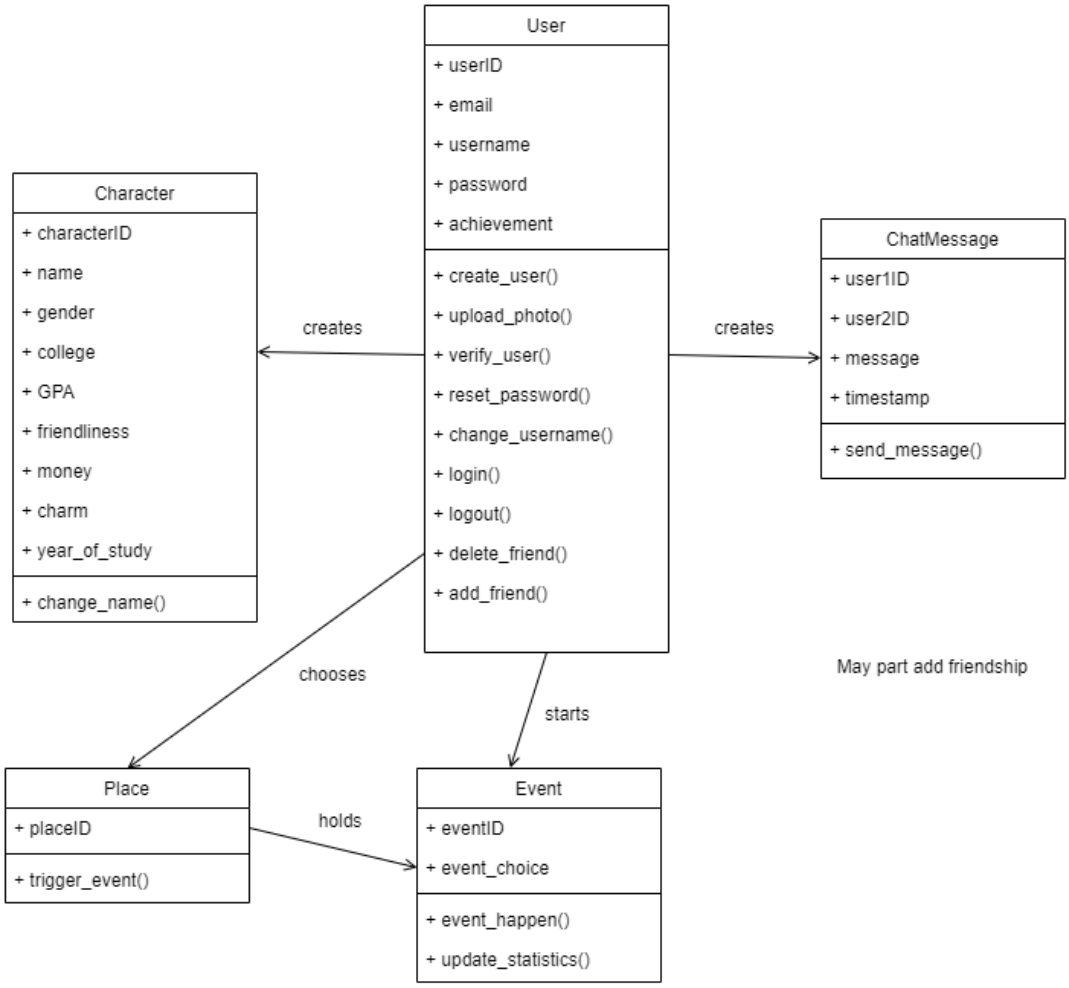
4.2.1 UMLs of Major System Components

4.2.1.1 User Case Diagram



Visitor will become user after registration.

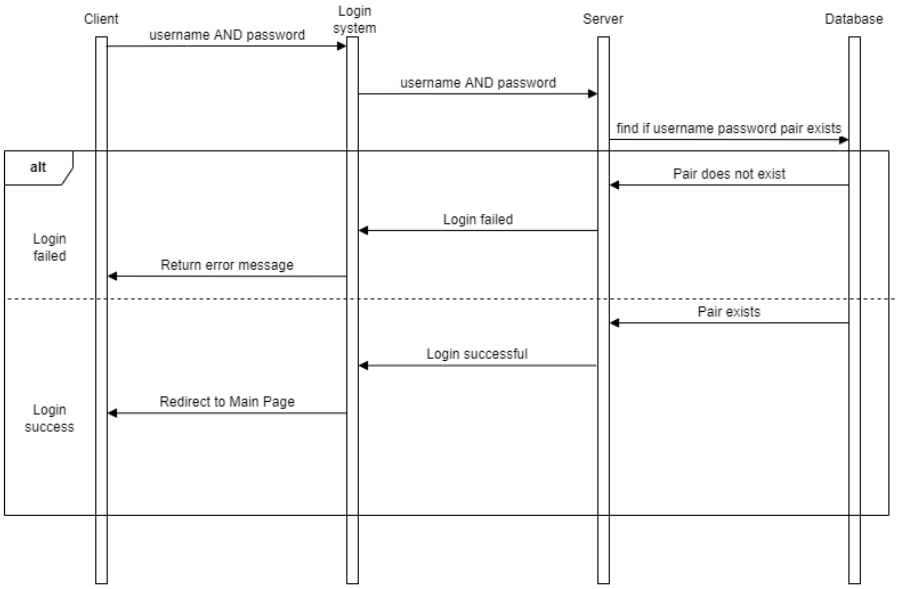
4.2.1.2 UML Class Diagram



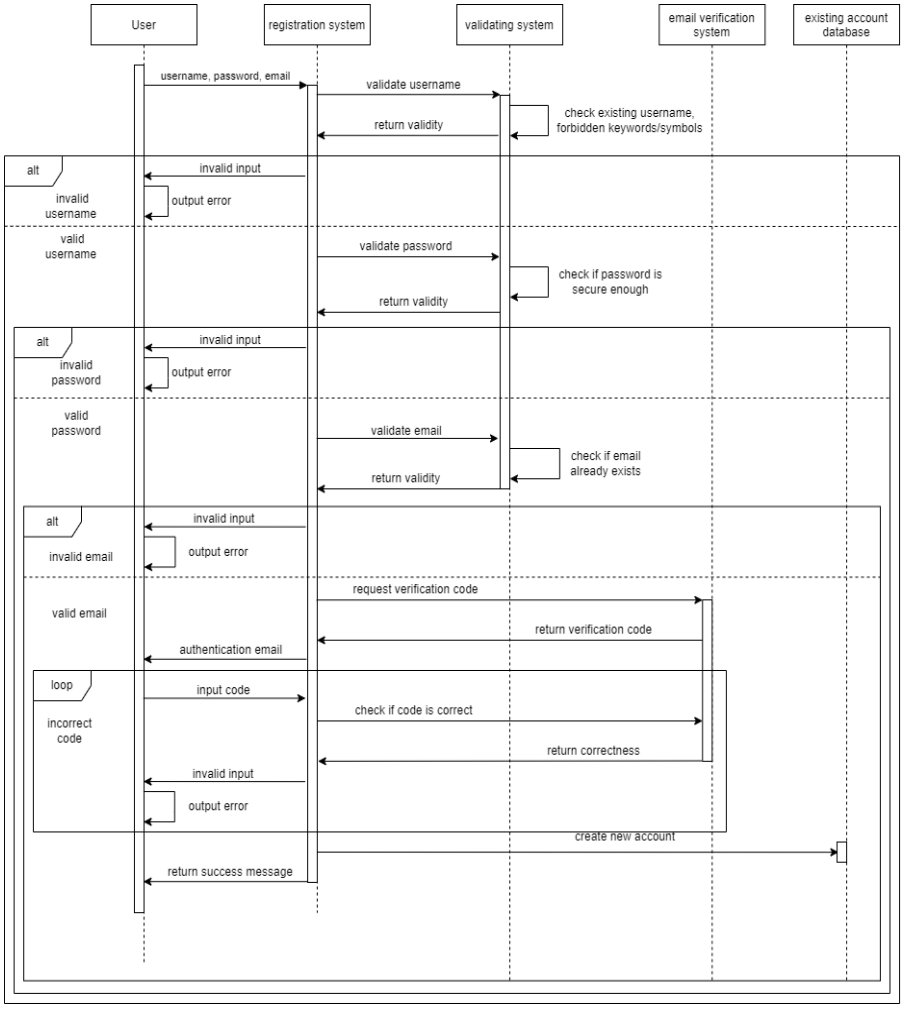
This diagram describes the classes of our software, including their attributes and methods. User is a class to store and manipulate user information. An object in class User represents a user registered in our software. In our game, the user can create his/her character. The characters belong to class Character and the class stores the statistics of characters. Also, the user can send messages to other users and the messages are stored in Class ChatMessage. During the game process, users can choose different places in class Place to go. In different places, an event in class Event will be triggered based on the place, game process and statistics of users. According to the result of the events, the statistics of users would be updated.

4.2.2 Sequence Diagram

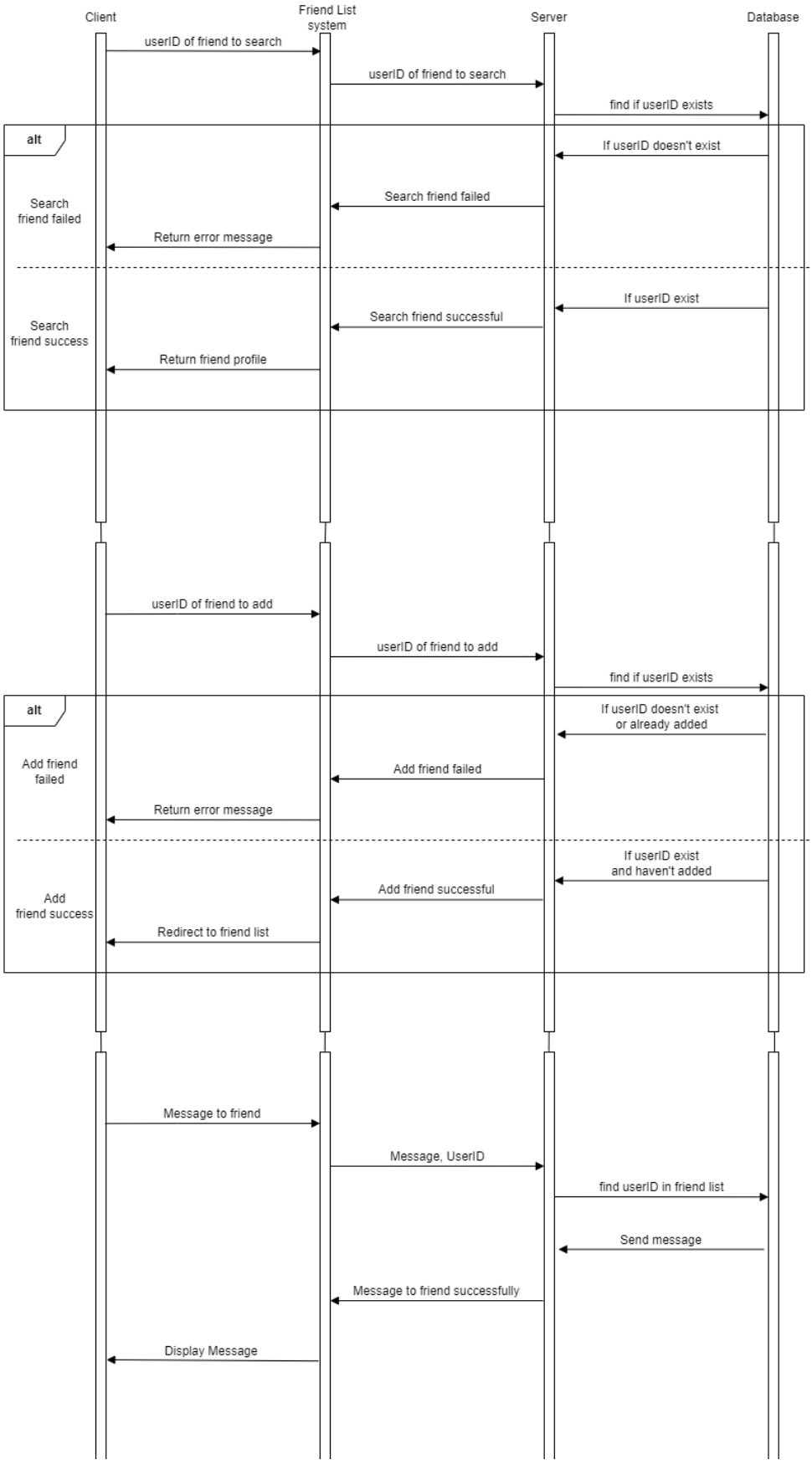
Login System Sequence Diagram

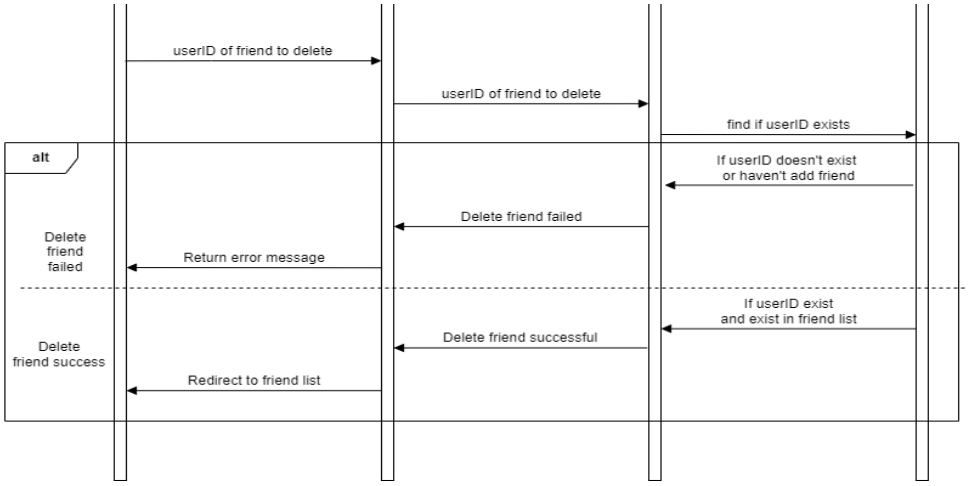


Registration and Verification System Sequence Diagram



Friend List System Sequence Diagram




userID of friend to delete
userID of friend to delete
find if userID exists
alt
If userID doesn't exist
or haven't add friend
Delete friend failed
Delete
friend
failed
Return error message
If userID exist
and exist in friend list
Delete friend successful
Delete
friend success
Redirect to friend list

4.2.3 Activity Diagram

Event UML Activity Diagram