1. Introduction

1.1 Purpose of Document

This document is the Software Requirements for the Food Friends App. The purpose of this document is to list all the project considerations, define the project scope, list all functional and non-functional requirements and create use diagrams with their respective descriptions. The project considerations section identifies all potential expenses, problems, ongoing questions and tools. The project scope section lists all functions that the Food Friends App must provide. This section also outlines the different roles and actions that each user has to handle. The System Architecture Diagram section is a visual representation of the flow of information inside the Food Friends App from a hardware viewpoint. Section four and five outlines all the functional and non-functional requirements. Existing functional requirements are given a category, requirement ID, description and priority level. New functional requirements, as well as changes to present requirements are given a new requirements ID or corresponding ID respectively. Non-functional requirements are given a purpose and a metric used to quantify each requirement. Section six and seven detail the use cases of the system.

1.2 High-level Product Overview

Food Friend is a product that will provide an ability for users to find their acquaintances to have wonderful meals with. If both of the users matched together, the applications will provide the location of the restaurant to the both of the users. Users can then interact with new people around them and make new friends or connections. The system will provide applications that are available through a web browser and smartphones.

The following is an overview of the product:

* Capable of creating and deleting account
* Capable of modifying displayed name, email address, password
* Capable of clicking forgot-password, inserting their corresponding data, and resetting password through the message sent to personal e-mail
* Capable of creating, updating, and deleting event
* Capable of searching preferred restaurants
* Capable of adding and removing preferences
* Capable of allowing the users to find the people that matched together
* Capable of showing the direction of restaurants
* Capable of showing promotional price of restaurants
* Capable of viewing and creating reviews
* Capable of using chat functions
* Capable of viewing explore page of events
* Capable of receiving new notifications
* Capable of storing users’ information in the database
* Capable of retrieving users’ information from the database
* Capable of showing the matched results

1.3 Acronyms & Extrapolations

Listed below are all of the acronyms used in the Software Requirement Specification. All acronyms and definitions may be used interchangeably throughout the document.

|  |  |
| --- | --- |
| Acronym | Definition |
| App | Application |
| ID | Identification |
| MCU | Ming Chuan University |
| AC | Applied Computing |
| API | Application Programming Interface |
| UI | User Interface |
| PC | Personal Computer |
| GPS | Global Positioning System |
| CPU | Central Processing Unit |
| AC | Applied Computing |
| HTML | Hypertext Markup Language |
| CSS | Cascading Style Sheets |

2. Project Consideration

2.1 Identified Resources

The following list of resources is what the MCU-AC department will be utilizing during design and development.

* International College’s Applied Computing iCube (Lab)
* 6 Philips Computer with Intel® Core™ i5-8500 CPU @ 3.00 GHz
* 1 iMac's 27-inch with 5K Retina
* 2 HP ProBook 440 G6 Notebook PC

2.2 Possible Tools

Below are all the lists of the required tools and mediums that facilitate and enhance the smoothness of Food Friend project.

* Firebase Real-Time Database
* Firebase Functions
* Firebase Storage
* Firebase Hosting:  
  - Node.js  
  - Window Command Line
* Text Editor:  
  - Atom (JavaScript, HTML, CSS)  
  - Google Editor (HTML, CSS)  
  - IDLE (Pyhton 3.6.4)  
  - App Inventor 2 (JavaScript Code Blocks)  
  - Swift (Xcode)  
  - JavaScript Blocks Editor  
  - Mu Pyhton  
  - JSON Editors  
  - Visual Studio
* Development  
  - React Native  
  - Cloud Firestone
* Communication:  
  - Line (Texting)
* Code Reposistory  
  - Github
* Video Meetings:  
  - Skype  
  - Google Hangouts
* File Sharing:  
  - Google Drive
* Media Used:  
  - YouTube  
  - Google

2.3 Open Issues and Questions

Open concerns that will continue throughout the duration of the project are in terms of scalability and structure of data processing. How much storage will be required to store user’s data? How frequently can the database be updated? How frequently can the database be updated and how long will user information be held in database? What limits are set on the internal users as to what they can and can’t see? Will those restrictions be the same for the external users as well?

Another important issue is the data loss and data security. How will security in the current version be implemented? How might future versions improve security? Will user data be safe? To what extent is hashing and encryption is used throughout? Could future versions implement local buffer servers to store and distribute data locally in case of network failure? These are some of the concerns that we should ethically consider very seriously as should future development groups. However, existing time and budget constraints are always a force for compromise on the issue. Setting an ideal compromise that ensures sufficient security while allowing the system to be feasible with our resources will be a difficult issue.

3. Project Scope

Food Friend aim to improve communication and networking in modern ways. The app will not only let users create their own profile, but also will have data available of each user and their own preferences, that could determine the match. Two or more people could meet together and build a relationship that starts with their preferences in food.

3.1 Feature Outline

To achieve this goal, the team firstly decided to develop an application that let users to create their account. Secondly, the team decided to develop basic functions, such as: Menu, Search Page, Notifications, Message Box, Create Events, etc. Thirdly, one more application would be developed for users, where users could find directions of the restaurants and detect other nearby users using Google API.

App and Browser Components:

* Create account
* Log in
* Log out
* Create event
* Join event
* Edit event
* Delete event
* Leave event
* Use map to search restaurant
* Update profile
* Receive message
* Send message
* See matched result
* Check participant location

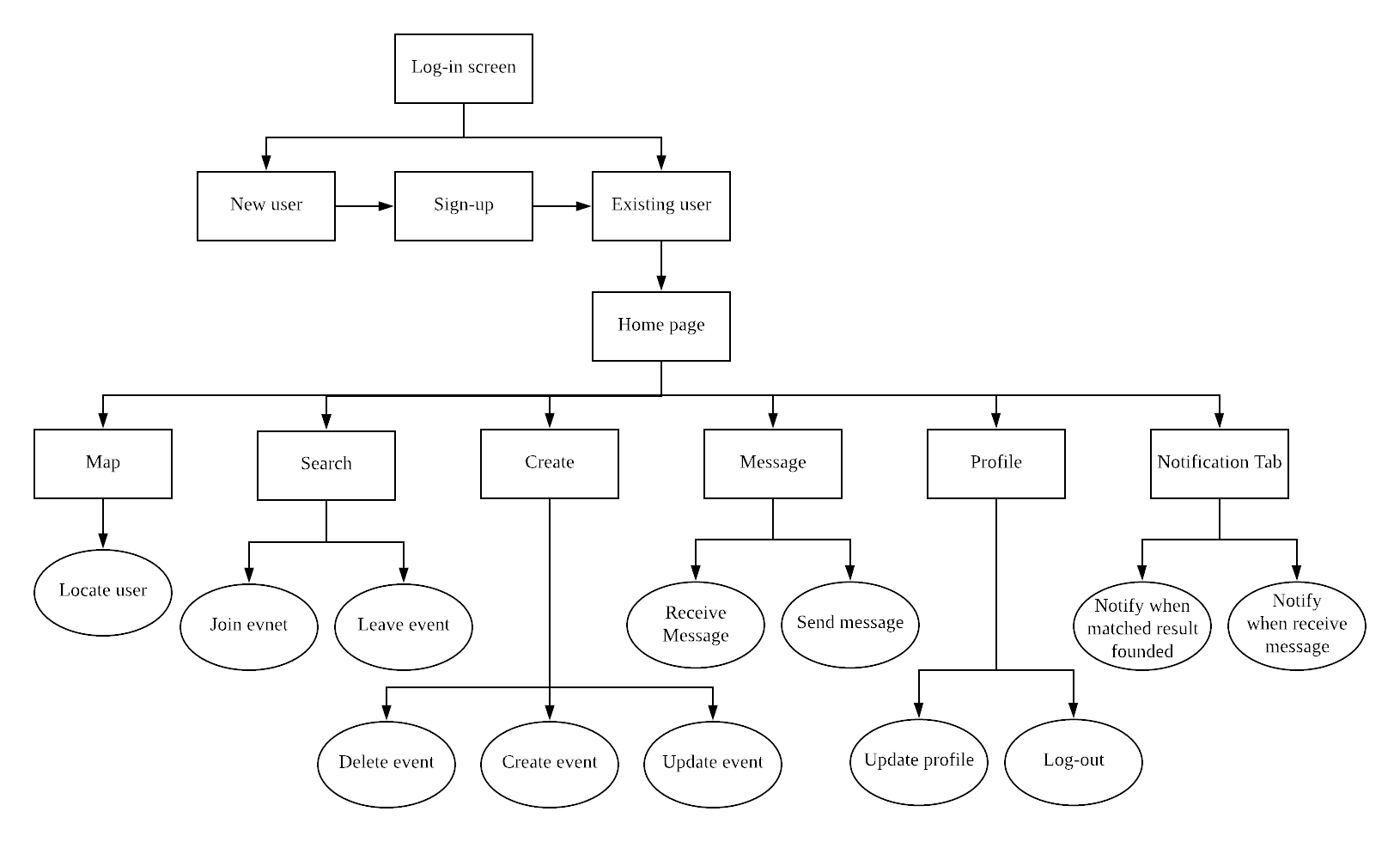
3.2 User Profile

The users of Food Friend focus on a single user itself and how to function while using the app. Users are able to create their own profile, upload food pictures, invite friends, meet-up, and also discover new restaurants. They can also use this app to find new friends, especially foreigners who wants to explore local foods and where they can find the best food near them. This app also allows user to upload photos of their foods for another users' interest. Lastly, this app will provide security for every single user’s by creating their own ID and password.

3.3 User Interface Map

User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. User Interface Design is utilized to help developers to get an overview of how the app should be implemented.

The purpose of the User Interface Design diagram is to understand the flow of users in the system to improve the quality of user’s experience and to ensure consistency. Users that try to log in from both application and browser will be specified based on their status. If users don’t have any existing account yet, users have to sign up. Once users are able to log in, users will find the main homepage that consists of an explore page, a profile page, a notification box, and a message box. Each page has its own functionality such as creating events as well as updating and deleting it, leaving and joining events, getting direction, updating profile and logging out, seeing matched results, retrieving new messages, sending and receiving messages.



3.4 System Architecture Diagram

To begin with, the tools the development team use in this system are mostly Google API. Secondly, the server and database are handled by firebase. This is because the team finds Firebase is relatively simpler and more effective to use. In addition, PC, laptop, mobile (iOS), mobile (Android), and tablet are provided for end users.

The following diagram shows the communication between the devices’ data binding with the server and database. The initial request sent from the user’s end device through web application. Server receives the request and interacts with the database. Additionally, Google API have the responsibility to locate users with other users nearby and to show directions to the restaurants. Furthermore, app version (mobile and tablet) are developed using App Inventor 2 and Swift, mostly. The database will respond to the request from the server by retrieving the proper data. To conclude, the data flow would be database to end users.

A close up of a logo

Description automatically generated

4. Functional Requirements

The following tables present listings of the functional requirements for this application. The functional requirements are listed for each version by Category, Requirement ID, Requirement, Description, and Priority (Note: “1” is the most prioritized that has to be in the Food Friend app, while “3” is not as important, but it is nice to have the function in the app).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Requirement ID** | **Requirement** | **Description** | **Priority** |
| Login | 1 | Users with an account must be able to login. | Users must be able to log in to the system with correct combination of ID and password. | 1 |
| 2 | Users must be able to reset their password when they forget. | Users with an account must be able to reset their password. | 1 |
| Account | 3 | Users must be able to create new account. | Users must be able to create an account with new ID and password. | 1 |
| 4 | Users must be able to edit an existing account. | Users must be able to edit an existing user account when they want to make changes with the account. | 1 |
| 5 | Users must be able to delete an existing account. | Users must be able to delete an existing user account when the account is no longer needed. | 1 |
| 6 | Users must be able to change password for an existing account. | Users must be able to change password for an existing account to maintain the security. | 1 |
| Logout | 7 | User must be able to logout from their account. | User must be able to logout from their account when they no longer need to use the system. | 1 |
| Events | 8 | User must be able to create new events. | User must be able to create new events when user wants to start an event. | 1 |
| 9 | User must be able to update existing events. | User must be able to update existing events when there is changes to the event. | 1 |
| 10 | User must be able to remove existing events. | User must be able to remove existing events when user no longer want to have that event. | 1 |
| 11 | User must be able to join existing events. | User must be able to join existing events when user wants to be a part of an event. | 1 |
| 12 | User must be able to leave existing events. | User must be able to leave existing events when user doesn’t want to be a part of an event anymore. | 1 |
| Notification | 13 | User must be able to see new notifications. | User must be able to see new notifications that are sent by the system. | 1 |
| 14 | User must be able to be reminded when incoming event is happening. | User must be able to be reminded when incoming event is happening in near time. | 1 |
| Event Schedule | 15 | User must be able to see existing schedule of events. | User must be able to see existing schedule of events that they created or joined. | 2 |
| Chat | 16 | User must be able to chat other event participants. | User must be able to chat other event participants to exchange information. | 1 |
| 17 | User must be able to receive other event participant’ chat. | User must be able to receive other event participant’ chat to exchange information. | 1 |
| Reviews | 18 | User must be able to view their own reviews. | User must be able to view their own reviews to see how others perceive them. | 1 |
| 19 | User must be able to view other user’s reviews. | User must be able to view other user’s reviews to see how other user is perceived. | 1 |
| 20 | User must be able to give other user’s reviews. | User must be able to give other user’s reviews to give what they think of the user. | 1 |
| 21 | User must be able to update other user’s reviews. | User must be able to update other user’s reviews when they give incorrect reviews. | 2 |
| 21 | User must be able to block and report other user’s. | User must be able to block and report other user’s with bad reputations. | 2 |
| Match | 22 | Users must be able to view existing matches. | Users must be able to view existing matches when other user join their events. | 1 |
| 23 | Users must be able to remove matches. | Users must be able to remove matches when they don’t want to be matched anymore. | 1 |
| Map | 24 | User must be able to locate themselves. | User must be able to locate themselves on the application based on their GPS. | 1 |
| 25 | User must be able to find nearby users. | User must be able to find nearby users on the application. | 1 |
| 26 | User must be able to find various direction. | User must be able to find various direction of the matching restaurant. | 1 |
| 27 | User must be able to view map with nearby restaurants. | User must be able to view map with nearby restaurants when they want to see available restaurants for match. | 1 |
| About Us | 28 | User must be able to view developer’s information. | Users must be able to view brief information about developer. | 2 |
| System Policy | 29 | User must be able to view the system policy. | The system policy is for the users to understand the rules and regulations of the system. | 2 |

4.1 Database

The table below lists the functional requirement that must be met by the database.

|  |  |  |  |
| --- | --- | --- | --- |
| **V/X** | **Requirement** | **Description** | **Implementation** |
|  | Add event | Allow user to add event. |  |
|  | Join event | Allow user to join event. |  |
|  | Remove event | Allow user to remove event. |  |
|  | Create profile | Allow user to create new profile. |  |
|  | Edit profile | Allow user to edit information from a new profile. |  |
|  | Remove profile | Allow user to remove an existing profile. |  |
|  | Create account | Allow user to create account. |  |
|  | Change password | User can change their password when they forget it or when they prefer to create better password with stronger security features. |  |
|  | Login | Allow user to login account. |  |
|  | Logout | Allow user to logout account. |  |
|  | Forgot password | User can retrieve a password if they forgot it. One will be required to provide User ID, full name, email and national ID. The password will be sent to the user’s email for retrieval. |  |
|  | Forgot user ID | User can retrieve their user ID by clicking forgot ID. This will lead the user to the forgot ID page. A user will be required to submit their phone number and birthdate to retrieve their ID and will be shown in a column below. |  |
|  | Find restaurant location | Allow user to find restaurant location. |  |
|  | View profile | Allow user to view profile. |  |
|  | Chat user | Allow user to chat with other users. |  |
|  | Send message | Allow user to send the message. |  |
|  | Reply message | Allow user to reply the message. |  |
|  | Block user | Allow user to block other users. |  |
|  | Unblock user | Allow user to unblock other users. |  |
|  | Locate user’s location | Locate user location. |  |
|  | Find nearby restaurant location | Find nearby restaurant location. |  |
|  | Check nearby user location | Check nearby user location. |  |

5. Non-functional Requirement

Browser Version

To develop the application there are various standards that must be met to judge the operation of the application, other than its general usage. Table 1 shows the Food Friend App browser version standards specified by the browser team.

|  |  |  |
| --- | --- | --- |
| Requirement | Purpose | How the requirement will be measured |
| Security | When dealing with personal data, security is the most important factor. The Food Friend App is required to properly secure users’ information. Using proper security measures to ensure users’ information is secure from unauthorized users. | The browser team will implement security measure on the application. |
| Reliability | A reliable application is mandatory for usability. The application is required to retrieve and process information as fast as possible. Wait times should be close to non-existent. | The Browser team will carry out trails and experiments on figuring out the Mean Time Between Failure. Therefore, the browser team may be required to have scheduled maintenance and/or updates. |
| Usability | Methods must be put in place on using the application to minimize time consumption on accessing the features of the application. | The Browser team will attempt to create a user-friendly interface to help user to easily read and navigate the application. |

Mobile App Version

The non-functional requirements chosen for the mobile app version of the Food Friend App were Reliability, Privacy and Compatibility. These non-functional requirements are crucial to the success of great performance and maintenance of the application. Rapid response times are important to the longevity of the Food Friend App. Additionally, must be compatible with the latest versions of Android and IOS. Table 2 below shows the non-functional requirements for the mobile app version.

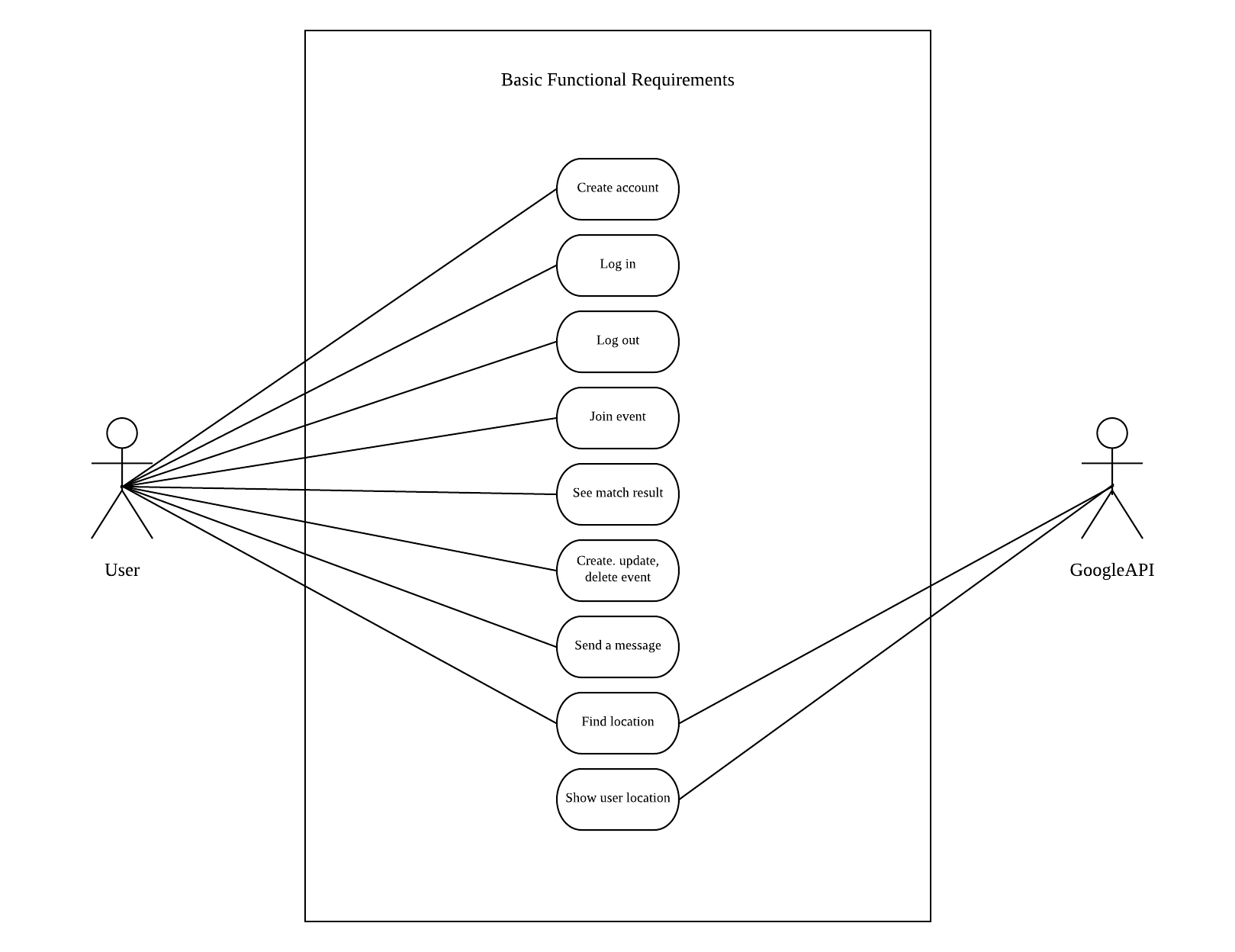
|  |  |  |
| --- | --- | --- |
| Requirement | Purpose | How the requirement will be measured |
| Reliability | The mobile app version will be able to execute functions specifically designed with minimum to no failures. | From the feedback function, the mobile team will know how reliable the application has been and will rectify any bugs or errors. |
| Privacy | Data used by the users will be stored in the cloud database (Firebase/Firestore). Protecting the users’ personal and other relevant information is a big priority. | To access the application on the mobile app version, users must input their personalized ID and password to login the application. |
| Compatibility | The mobile app will be compatible with different operating systems to meet the expectations of the end users. | The Food Friend app will be designed and developed using \*insert mobile app creator here\*. Cloud Firebase/Firestore will be implemented to allow cohesion the mobile app and the browser app in regard to data access. |

6. System Design

6.1 Use Case Diagram

Use case diagram shows an Actor and it is represented as a stick. The Use case itself will be named in an oval rectangle. The Association Line will connect the actor to the Use Case they have relationship with. The System Boundary represents all Use Cases in a specific sub-system. The sub-system is a list of major components (Use Case) for a part of a system. In this scenario, a subsystem represents functional requirement categories.

Users are able to create account and login to the account. While users are online, they could create and join events. In addition, they could see the match result of their events and send message to other participants. Also, users can find location of the restaurant and other users nearby using Google API.



6.2 Brief Use Case Description

The following table shows the brief description of use cases.

|  |  |
| --- | --- |
| Use Case | Brief Use Case Description |
| Show user location | Google API will show the location of the users. |
| Create account | Users can create an account. |
| Login | Users can sign-in to their account. |
| Logout | Users can log out of their account. |
| Join event | Users can select which events they would like to join. |
| Match | Users can match with other users. |
| Create event | Users can create event. |
| Send a message | Users can send messages to other users. |
| Find location | Users can find the location of the restaurant, matched users. |

6.3 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity.

Activity Diagram – Login

An activity diagram for logging in to the system. When user enter the username and password, the system will validate the login information. If the information is wrong; user need to enter the username and password again. If the information is correct, user can successfully log in to the system.

A screenshot of a cell phone

Description automatically generated

Activity Diagram – Match

An activity diagram for matching the users. User will select the desired restaurant. If two of the users match together the system will notify both of the users. If the user fails to match, the system will find other person to match.

A close up of a person

Description automatically generated

Activity Diagram – User Profile

An activity diagram for checking user’s profile. User request the system to check other user profile. The system will fetch data and show the profile to the user.

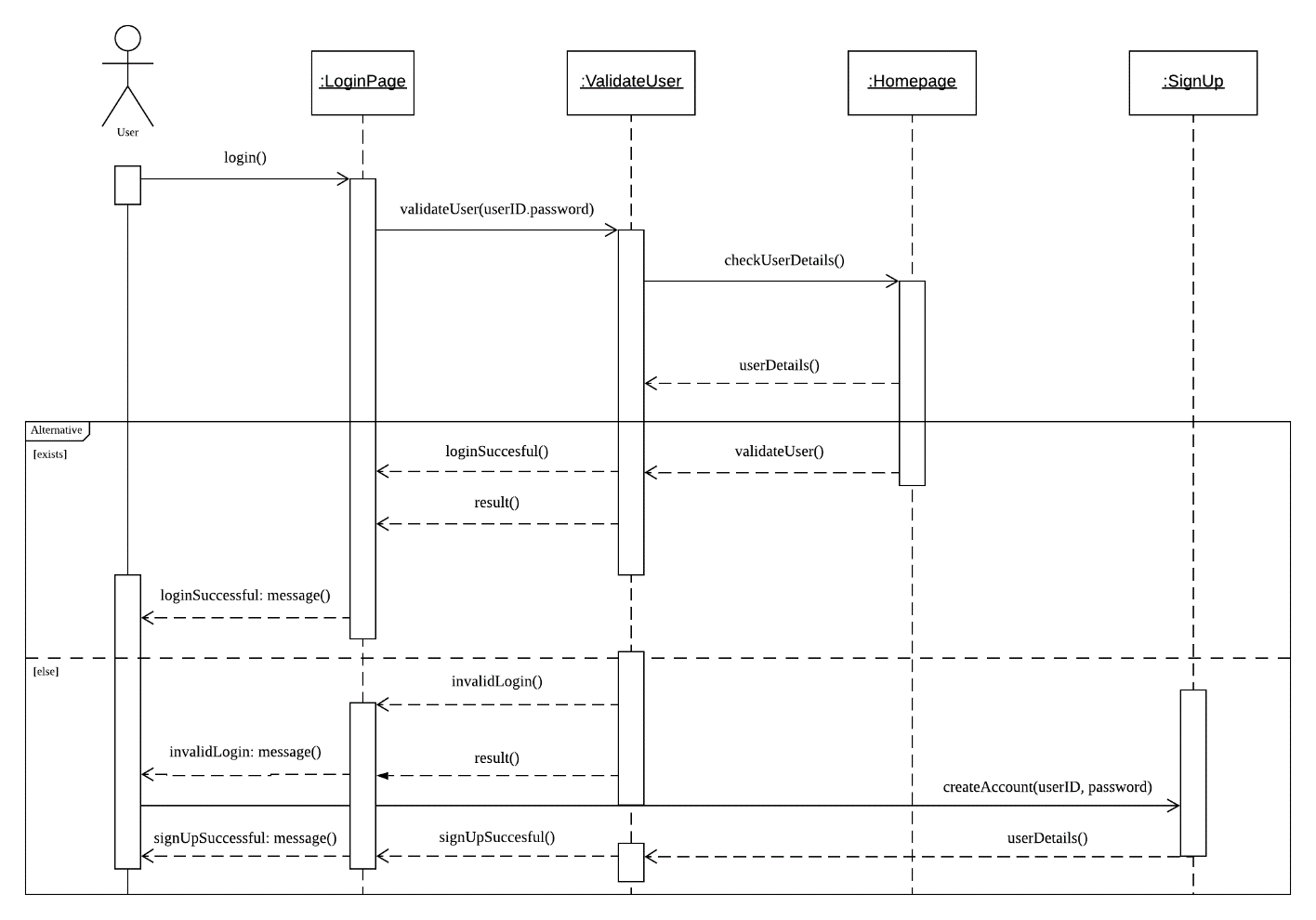
A screenshot of a cell phone

Description automatically generated

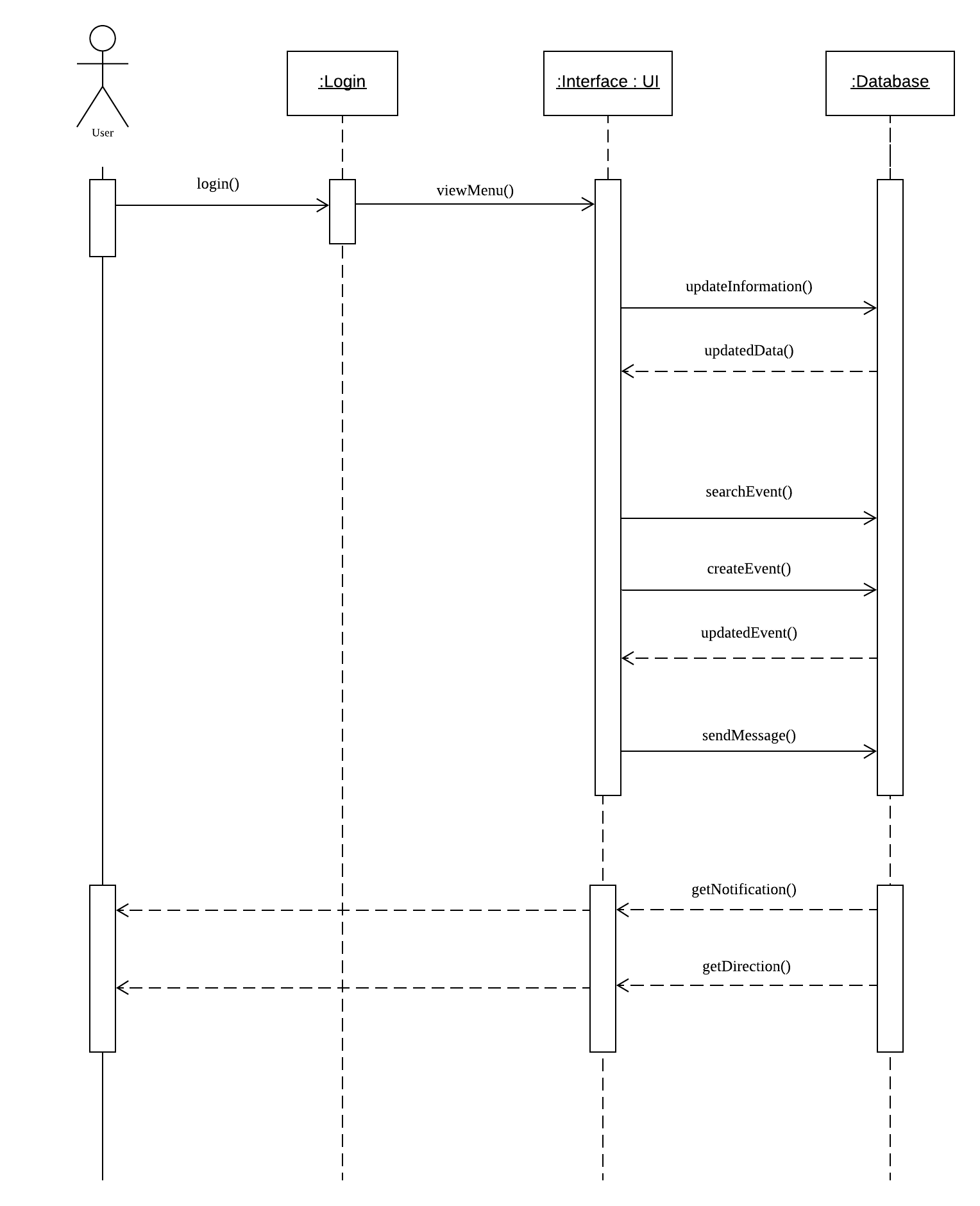
6.4 System Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

In the System Sequence Diagram, user have tried to login on the Login Page. The system will the validate user information using the user ID and password. If the login is successful, user will be able to receive message showing that the login has been successful. If user fails to login, user will be able to receive message showing that the login has been invalid and encourage user to sign up by creating new account (ID and password). The new detail will be then stored to the server and validate user’s login. Furthermore, user will be able to use the application.



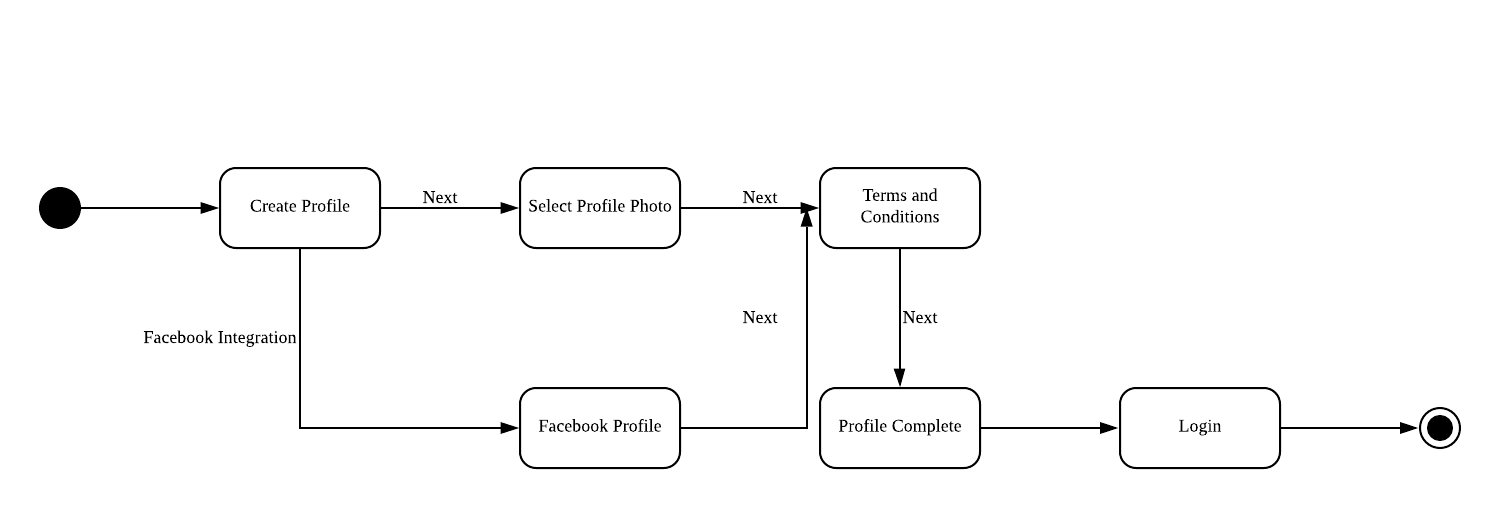
In addition, the System Sequence Diagram below visualizes how user access the database by logging into their account. Once user is able to login successfully, they will be taken to the interface of the application and be able to view menu with all the functions. In the application, user is able to update information, such as: events, profile, etc. When user store new information on the interface, it will be directly stored to the database. Likewise, when user update existing information, it will be stored at the database.

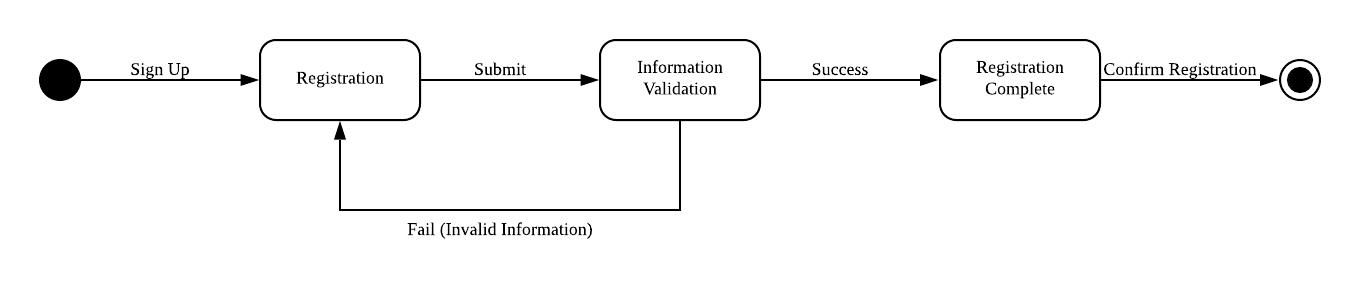
User also will be able to send message to another user. When that happens, it’s actually another information stored in the database. Moreover, users will be able to get notification and direction that are provided by the database.

6.5 State Machine Diagram

6.5.1 Create Profile

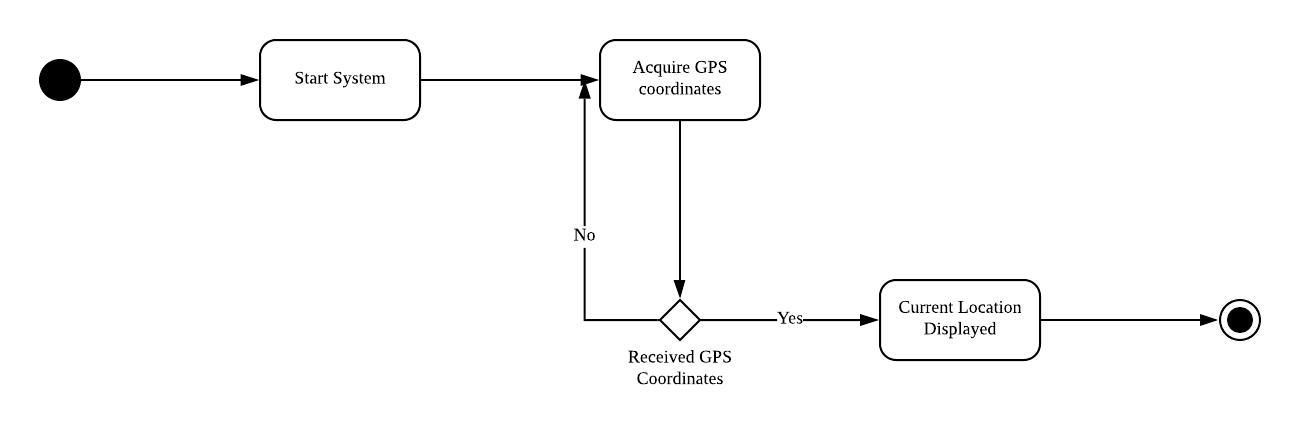
A state machine diagram models the behavior of a single object, specifying the sequence of events that an object goes through during its lifetime in response to events. As an example, the following state machine diagram shows the states these two objects go through during their lifetime.  Figure 1 and 2 shows the states in which a user may go about creating a profile/ signing up, as well as some encounters.





6.5.2 Map API/GPS Location

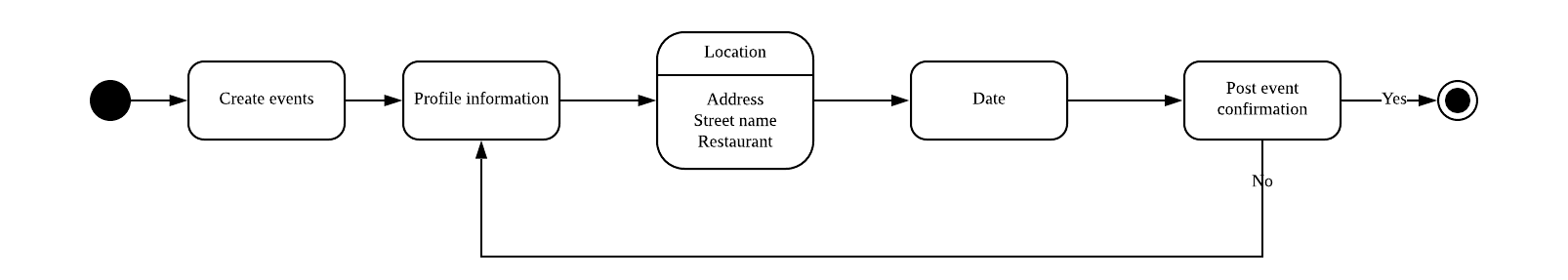
Figure 3 show the system acquiring a GPS coordinates in order to show current location.  The system begins then tries to get coordinates of current location.  If no coordinates are received, the system goes back and try acquiring the coordinates.  Once coordinates have been received that the current location is displayed.



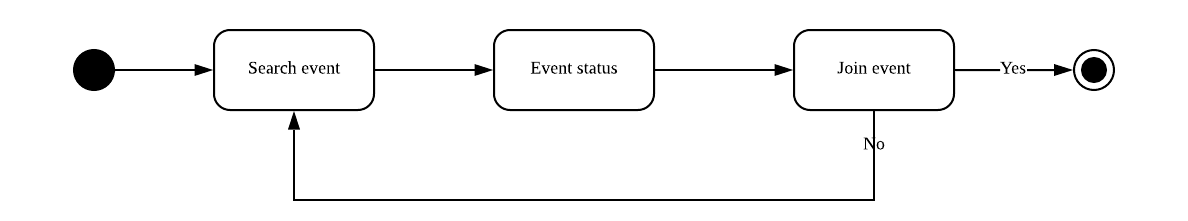
6.5.3 Event

The event for state machine diagram is used for the user to either participate or create an event for others that can join.

In Figure 3, the user creates an event that includes the name, location and the date. The user is free to check the event it created at the “confirm event post”. If the user is not satisfied with what he/she wrote, the user rewrites the event. If the user is satisfied with it, then the event is posted to the general public which will be seen at another user’s notification or at the main menu.



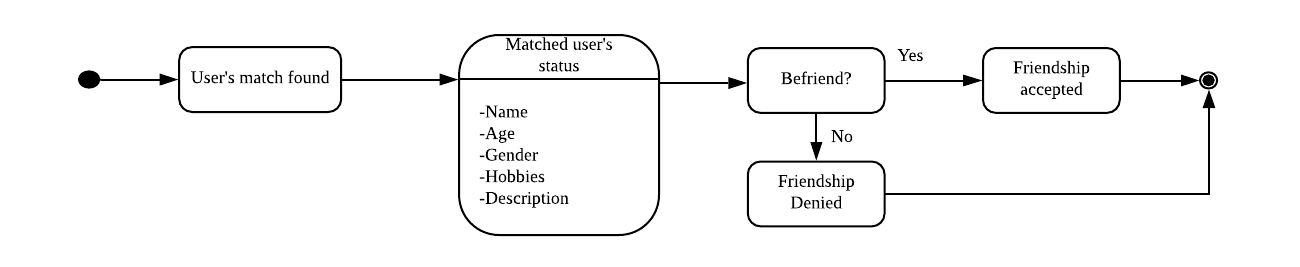
In Figure 4, the user wants to participate in an event that the other user has created. The user searches the names of events. Once found, the event shows its status to the user including its date and location. If the user wants to join, the user is automatically registered into the chosen event. If the user refuses to join, he/she goes back to the search event.



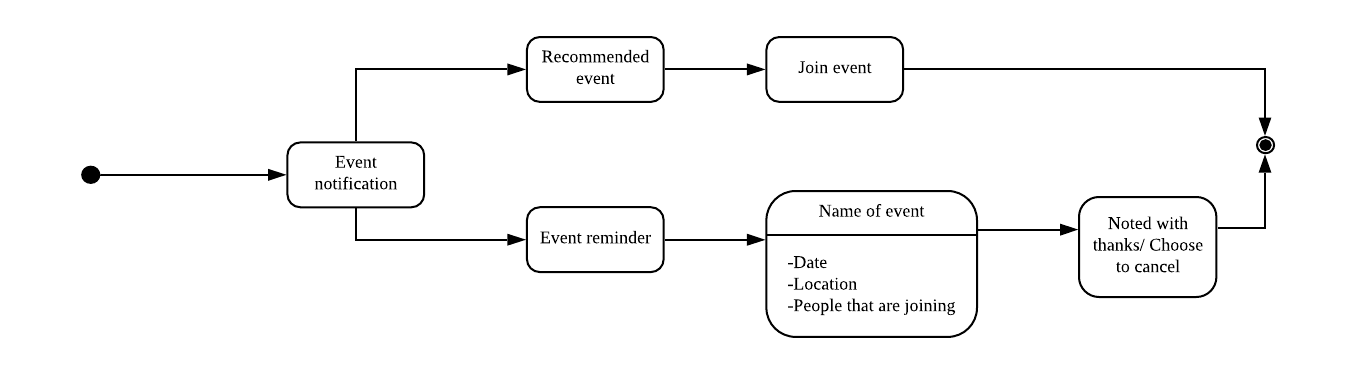
6.5.4 Notifications

The Notifications in state machine diagram is used for notifying the user about the event it joined or recommended, a match of another user found, and a message received from a friend.

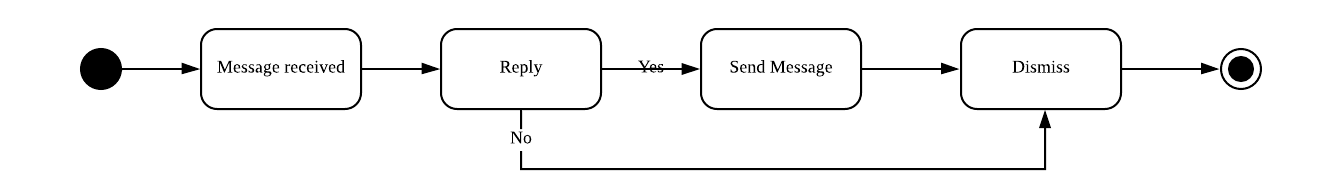
In Figure 5, the app notifies the user via “user’s match found”. The user looks into the matched user by seeing the user’s status (Name, Age, Gender, Hobbies, Description, etc.) If the user wants to accept the matched user as a friend, user and matched user become friends. If the user refuses to become friends, then friendship is denied. This means the user fails to have a match.



In Figure 6, the app notifies the user via “Event notification”. The event notification can be identified as a reminder or a recommended event. For recommended events, the user can look into the status of the event. If the user wants to join the event, automatically registers into the event. If the user does not want, then notification is dismissed. As for the event reminder, it notifies a reminder of the event the user participated by showing the name of the event and its status. The user dismisses the reminder by choosing “Noted with thanks”, and the user can also choose to cancel the event by choosing “Choose to cancel.”



In Figure 7, the user app notifies a user via “message received by (Username)”. The user can choose to either reply or not to the username or not. If the user chooses to reply, dismisses after the user replies with a message. The user can also dismiss if the user refuses to reply with the username



6.6 Class Diagram

Figure 8 illustrates the class diagram of application. Class diagram represent the static

view of an application. It is not only used for visualizing, describing, and documenting

different aspects of a system but also for constructing executable code of the software

application. For each entity, there are two types for variable; The public information that will

flow through the system and the private information that will be execute separately. They are

represented by plus sign and minus sign respectively. Moreover, lines and arrows

demonstrate how the entity connects and affects other entities.

A screenshot of a cell phone

Description automatically generated