**Software Requirements Specification**

**for**

FoodEngine

**Version 1.0 approved**

**Prepared by Gupta Atharv | Lee Bo Hua | Lin Kai | Randall Cher Jin Fong | Ng Wan Jing**

**SC2006 Group 3**

**05 Nov 2022**

**Table of Contents**

[**1. Introduction**](#_heading=h.1fob9te) **4**

[1.1 Purpose](#_heading=h.3znysh7) 4

[1.2 Document Conventions](#_heading=h.d71wzsx846hk) 4

[1.3 Intended Audience and Reading Suggestions](#_heading=h.pyqzrvnwmge6) 4

[1.4 Product Scope](#_heading=h.hmjegoiqhcm) 4

[1.5 References](#_heading=h.12vvu3w0p7c4) 4

[**2. Overall Description**](#_heading=h.x79uhqh875rm) **5**

[2.1 Product Perspective](#_heading=h.2s8eyo1) 5

[2.2 Product Functions](#_heading=h.i0z5s7esix9) 5

[2.3 User Classes and Characteristics](#_heading=h.4hivlyodztq5) 5

[2.4 Operating Environment](#_heading=h.btukp6vuagew) 6

[2.5 Design and Implementation Constraints](#_heading=h.dexni8r8n2rq) 6

[2.6 User Documentation](#_heading=h.95vn1y3qxg62) 6

[2.7 Assumptions and Dependencies](#_heading=h.nr1w0yonj821) 6

[**3. External Interface Requirements**](#_heading=h.hfo21u44icj) **7**

[3.1 User Interfaces](#_heading=h.2jxsxqh) 7

[3.2 Hardware Interfaces](#_heading=h.ahqyv5g593zi) 9

[3.3 Software Interfaces](#_heading=h.bcjta6sqvy3m) 9

[3.4 Communications Interfaces](#_heading=h.yqbc3825wgj6) 10

[**4. System Features**](#_heading=h.qxft0nf85xk8) **10**

[4.1 Account Registration](#_heading=h.tfo2ugy0x0nt) 10

[4.2 Login](#_heading=h.wfuh5kr6xtu9) 11

[4.3 Search Restaurants](#_heading=h.1pc5byme7613) 11

[4.4 Create Event](#_heading=h.k6rmujqkh1ls) 12

[4.5 Join Event](#_heading=h.8honvhbgoew5) 13

[**5. Other Nonfunctional Requirements**](#_heading=h.3whwml4) **13**

[5.1 Performance Requirements](#_heading=h.2bn6wsx) 13

[5.2 Safety Requirements](#_heading=h.z58ny8j9j6m2) 13

[5.3 Security Requirements](#_heading=h.blyh1044afd9) 14

[5.4 Software Quality Attributes](#_heading=h.wxaddfsct48l) 14

[5.5 Business Rules](#_heading=h.5i0u52o6lk24) 14

[**6. Other Requirements**](#_heading=h.n94fr2b71fbn) **15**

[**Appendix A: Glossary**](#_heading=h.iupqr69y1o1a) **15**

[**Appendix B: Analysis Models**](#_heading=h.uiscxgtnea0w) **17**

[USE CASE DIAGRAM](#_heading=h.qsyldugvazgo) 17

[DIALOG MAP](#_heading=h.o8d3fq6kai8m) 18

[SYSTEM ARCHITECTURE](#_heading=h.85ktreitlau5) 19

[DESIGN CLASS DIAGRAM](#_heading=h.w3g5v26xn3qz) 20

[SEQUENCE DIAGRAM - REGISTRATION](#_heading=h.7gx9ex1y1fdt) 21

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| Lee Bo Hua | 5/11/2022 | First edit | 1.0.0 |
|  |  |  |  |

# **Introduction**

## **Purpose**

The purpose of our application, The Food Engine is to help the users of the app to browse, search restaurants, create events and a community with other users of the app.

We use the current location of the user to give them the best possible suggestion for restaurants. It also fosters an online application community so people can eat together.

## **Document Conventions**

* Titles: Times New Roman font, size 18
* Subtitles: Times New Roman font, size 14
* Text: Arial font, size 11

Apart from that, implementing natural and unambiguous language to the best of our abilities is also part of our convention.

## **Intended Audience and Reading Suggestion*s***

* The SRS is meant to be read by testers, developers and maintainers, so that they can completely understand the current working model of the application and develop it further.
* The User Interfaces portion can also be read by new users who might want to know more about navigating through the web application.

## **Product Scope**

The current scope of the application is that it is hosted on the web and is limited to returning search results of restaurants within a 1.5 KM radius around the approximated location of the user, which we hope to expand in the future.

## **References**

* <https://staff.emu.edu.tr/alexanderchefranov/Documents/CMPE412/2-Intermediate_Project_Report_Template-SRS.doc>

# **Overall Description**

## **Product Perspective**

The FoodEngine is a web application created for people who want to find restaurants around them and also invite people to eat with them.

## **Product Functions**

Login Page:

* Accepts user’s email and password and displays Search Page if email and password are valid. If invalid details are entered, it will stay at the Login page.

Register Page:

* Accepts user’s email and password and adds the information into the Firebase database. If invalid details are entered, it will stay at the Register page.

Search Page:

* Accepts user’s keyword and displays relevant restaurants within a 1.5km radius of user’s location.
* Filter: Filter results according to ascending or descending levels of ratings.
* Show more information: Shows Restaurant Information
* Click on small app logo: Redirect user to Events Page
* Click on profile logo: Redirect user to Users Page
* Click on power logo: Log out and redirect user to Login Page

Restaurant Information:

* Shows the name, price level, rating, address of the restaurant and where it is on the map using embed map.
* Create Event: Shows Create Event Form only if the user is logged in.

Create Event Form:

* Accept user’s input and store the event details if valid details are entered.

Events Page:

* Shows all public events and shows private events only if users are following each other.

Users Page:

* Shows all users which other users can follow.

## **User Classes and Characteristics**

Singaporeans who want to bond with others over food, who want to use FoodEngine to find restaurants for the locations of their meetups.

## **Operating Environment**

Operating Platform for Web Application:

* Web browsers like Google Chrome

External API:

* Google Maps Static API
* Google Distance Matrix API
* Google Places API
* Geolocation API

Frontend:

* JavaScript
* React

Backend:

* NoSQL database cloud storage (Firebase)

## **Design and Implementation Constraints**

* We retrieve restaurant data from the Google Maps API, where it is quite impossible to store in the database as every call is based on the user's location and the keyword that they enter, which can change a lot of times.
* Free credits for using Google Maps API are limited, so we had to limit search results to a radius of 1.5km.
* As event details are stored using Firebase, it requires relatively good internet connection to be able to see the event details load.
* Users need to have Allow CORS: Access-Control-Allow-Origin Google Chrome extension and their location services enabled on their Google Chrome for the web application to run successfully.

## **User Documentation**

Full Source Code of application: <https://github.com/KaiKaizxc/food_engine->

## **Assumptions and Dependencies**

FoodEngine requires the location of the user, good internet connection and the Allow CORS: Access-Control-Allow-Origin Google Chrome extension. Hence, the system assumes that all the above requirements are met. If the user is not connected to the internet or does not have the Google Chrome extension, the app would not work. If the location is not known, there will not be results shown even after the user enters the keyword.

FoodEngine also depends on the Google Maps API for its data and hence if the Google Maps API becomes unavailable in some way, the web application will not be working. For the embed map in the Restaurant Information, it also depends on Google Maps. Hence, if Google Maps is not working, users would not be able to see the restaurant on the map as well.

# **External Interface Requirements**

## **User Interfaces**

|  | **Login Page**  Users will see this screen when first entering the application. They will be prompted to enter their Email and password to login. If they have yet to create an account, they can do so in the Register page. |
| --- | --- |
|  | **Register Page (optional)**  Users can enter their Email and Password to create an account to add into the Firebase Database. Following this, they can use the Login function with their Email and Password. |
|  | **Search Page**  Users can enter their Search Parameters for the application to return results of restaurants in their vicinity.  Users can filter search results according to ascending or descending levels of ratings.  Users can choose to navigate to the Events page with the small app logo, the Users page with the profile logo and the Login page with the power logo |
|  | **Restaurant Information**  Users can select the “Show more Information” after searching to see a popup with more information about the restaurant, with an option to create an event from there. |
|  | **Create Event Form**  Upon selecting the “Create event” option in the previous popup, users must fill in the blanks with valid inputs. Following which, event details are stored. |
|  | **Events Page**  Users can see all public events and show private events only if users are following each other. |
|  | **Users Page**  Shows all users which other users can follow. |

## **Hardware Interfaces**

* For network connection:
  + LAN cable or Wireless Network Interface Card (WNIC) needed.

## **Software Interfaces**

* System must have web browsers installed to support the application.

* Users must have CORS 0.1.8 extension enabled, which allows you to easily perform cross-domain Ajax requests in web applications.

* Users must have npm 8.19.2 installed to manage dependencies for javascript packages

* Software required Google Places API to get the information of restaurants in the 1.5 km radius,such as price level, rating, address location as well as the image of the restaurant. This feature must be supported by Google cloud platform.

* Software requires a Geolocation API to get information about the current location of the user

* Software requires Google Maps Static API to be able to display the map in the software .This feature must be supported by Google cloud platform.

* Software requires Google Distance Matrix API.This feature must be supported by Google cloud platform.

* Software requires Firebase, which we store information in, such as event details and user details.

## **Communications Interfaces**

FoodEngine requires an internet connection to be able to load the Google Maps Static API, Distance Matrix API and Geolocation API, to get details of Restaurants from the Search field.as well as the Firebase Database to store information such as events and user details.

# **System Features**

We will be organising this section by use cases.

## Account Registration

**4.1.1 Description and Priority**

Within the home page, users can create an account by selecting the “Register button” they will then have to fill in the necessary “Email” and “Password” fields with valid inputs to create an account.

Priority: High

Users must create an account in order to access key features of the application, such as creating event or joining an event.

**4.1.2 Stimulus/Response Sequences**

1. User clicks on REGISTER button in the main page to direct to the registration page.
2. Users enters this information as prompted in the registration page:
   1. Email
   2. Password
3. User clicks on “Register”.

**4.1.3 Functional Requirements**

REQ-1: The System must be able to determine whether user inputs for Register are valid or invalid.

* For email addresses, it must include @ and . in it. The email address should not already be in the database.
* For passwords, it must be at least 6 characters long, include at least 1 number or special characters and it will be salted with a hashvalue in the database.
* The system must display a error message if user enters invalid details for either fields.

## Login

**4.2.1 Description and Priority**

Within the home page, users can login to their account by selecting filling in the necessary “Email” and “Password” fields with their own registered Emai and Password to login to their account

Priority: High

Users must login to their account in order to access key features of the application, such as creating event or joining an event.

**4.2.2 Stimulus/Response Sequences**

1. User Enters their Email and Password into the email and password fields.
2. User clicks on the “Log in” button to login.
3. System returns a message “Welcome back”.

**4.2.3 Functional Requirements**

REQ-2: The System must be able to determine whether user inputs for Login are valid or invalid.

* For email addresses, it must include @ and . in it. The email address field should not be empty.
* For passwords, it must be at least 6 characters long and include at least 1 number or special characters. The password field should not be empty.
* The system must display a error message if user enters invalid details for either fields.

## Search Restaurants

**4.3.1 Description and Priority**

Users are able to search relevant restaurants, with regards to the user inputted keyword, within a 1.5km radius with this function and see the results within the UI. For users to create an event, they must first select a restaurant, which can be retrieved after using the search field.

Priority: High

For users to create an event or search for restaurants, they must utilize this function.

**4.3.2 Stimulus/Response Sequences**

1. The user calls this function by:
   1. Entering the name, description, or location of their desired restaurant through the search bar.
   2. The user can then adjust the filter based on their preference, which affects search results.
2. All restaurant within the defined location range and filters, that matches the search case will be displayed, with order based on how close it matches the search parameters, and proximity to user’s location (if enabled).

**4.3.3 Functional Requirements**

REQ-3: Location services should be enabled for any devices using this function.

* The system must display a popup indicating “Unable to fetch location” if location services are not enabled.

REQ-4: The system must display the picture, name, price level, rating and address of every restaurant when display all the results in a list.

* Price level fields retrieved from Google Maps API may not be present. If so, the system must replace and show the field with a “-”.

REQ-5: The system must display everything in stated in REQ-4 and also an embed map of the restaurant location on the map when user clicks on “Show more information!”.

## Create Event

**4.4.1 Description and Priority**

Users are able to Create events for other people to join. They can create a public event where everyone can see, or create a private event where only people who follows each other can see. Users have to input information such as event name, visibility, time and duration when prompted by the system

Priority: High

Users are required to create event for others to see.

**4.4.2 Stimulus/Response Sequences**

1. The user should have already searched for a business, and clicked on it for more information.
2. User will click on the button “Create Event”.
3. User will be prompted for event details.
   1. Event Name, Visibility, time, duration
4. Event will be successfully created, and stored in the database.

**4.4.3 Functional Requirements**

REQ-6: The system must not allow the user to Create Event if they are not logged in.

REQ-7: The system must be able to determine whether user inputs for Create Event are valid or invalid.

* The System must display an error message if any of the event details such as event name, visibility, time and duration are not entered.
* The System must display an error message if event details such as time and duration are not valid. i.e., Time and duration entered by user cannot be outside of the restaurant’s opening hours.

## Join Event

**4.5.1 Description and Priority**

Users are able to Join events which have been created by other people. They can join a public event where everyone can see, or join a private event which have been created by people who selected private event when creating the event and is followed by the user.

Priority: High

Users must be able to join an event people they are following have created or others who have created public events.

**4.5.2 Stimulus/Response Sequences**

1. User goes on to the events page.
2. User click on ‘JOIN’ to join an event that is either public or created by their friend.
3. A pop up message saying “You have successfully joined {event name}, which will be held at {event location} on {event time}!” will be displayed.

**4.5.3 Functional Requirements**

REQ-8: The system must not allow the user to Join Event if they are not logged in.

REQ-9: The system must display an error message stating “Capacity already maxed, cannot join!” if user tries to join an event that is already full.

REQ-10: The system must display an error message stating “You are already part of the event” if user tries to join an event that they have already joined previously.

REQ-11: The system must not display private events for other to join if both the owner of the event and user are not following each other.

REQ-12: The system must display public events for all users to see.

REQ-13: The system must be able to update the capacity of the event whenever a user joins the event.

# **Other Nonfunctional Requirements**

## **Performance Requirements**

* The application should not take more than 5 seconds to process the login and register function.
* The application should not take more than 5 seconds to fetch information about maps, and filter according to the rating of the places.

## **Safety Requirements**

Users would have to be cautious when choosing to join public events hosted by those that they do not know.

## **Security Requirements**

* The passwords registered for users should be of a certain length and requirement
* Passwords should not be accessible or readable by others, even to admins.
* Passwords should be salted with a hash number.
* The website should not request too many requests to google API. It could use a debounce function instead.

## **Software Quality Attributes**

* **Flexibility**: The application should be flexible in the sense that it should be able to integrate third party attributes and plugins, such as data analytics and other features.
* Other plugins for data analytics could be in the sense that we can monitor user's usetime and their feedback and reviews, and create a dashboard with it to allow easy visualization about the performance of our app

* **Availability**: Our software should be highly available with high 9s. Such concerns may be addressed using where we host our application and the performance of the cloud infrastructure backend we are using.
* Google is a great choice with high 9s.

* **Maintainability**: Our files are neatly arranged with well named variables, allowing the next developer to come in and navigate the system easily. Additional help is provided with documentation.

* **Reliability:**  Our system is highly reliable, in the sense that events created can be read by other users quickly and information is quickly stored across different user's database.
* Events created are generated quickly and made available to be seen by another user

* **Partition**: In the event of partial network failure, our system has to be down to fix the failure, and no reads or writes can be done to that network currently. However, partition is unavoidable in that system, hence it cannot be avoided
* When a user's account suffers from network issues, other users may not be able to view the events created by his user profile (read) and also may not update the event created by him (write).

## **Business Rules**

* Only users with a valid account can create or join events. Those using a guest account would not be able to do so.

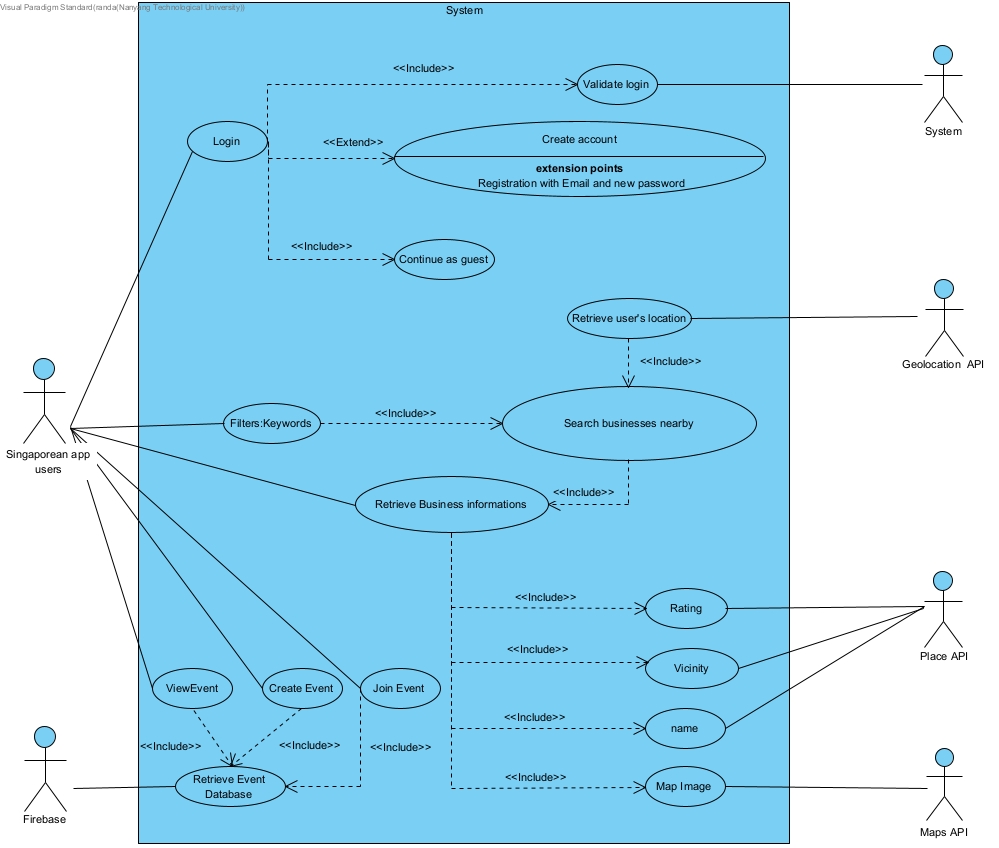
# **Other Requirements**

All the information about our software is enclosed in this SRS and the following appendices.

# Appendix A: Glossary

| **Terms** | **Description** |
| --- | --- |
| **Register** | The process of making a new account on the app. |
| **Guest Login** | If a user doesn’t intend to create an account, they can still access the app via guest login. However, they are unable to create or join events. |
| **Login** | The process of logging in to an existing account. This can be done by entering your email and password. |
| **User** **Account** | An account meant for normal users of the app, who want to find a restaurant nearby and create or join events. |
| **Current Location** | The live location of the user, determined by Geolocation API. |
| **Location Refresh** | The function refreshes the app, to indicate the accurate location of the user. |
| **Business Information** | All the relevant information of a certain business, which would include, its name, location, cuisines, timings, reviews. |
| **Range** | The radius in kilometer (KM), up to which the application searches and recommends businesses to the user. |
| **Ratings** | Information is taken from Google Maps API. |
| **Friends** | Users can make friends or follow their real-life friends on the app. That way they can see where their friends are eating and can join them too.  A user can only be friends with another user if they follow each other. |
| **Event** | A registered user can create an event at a certain restaurant, for their friends to join. |
| **Event Details** | Event Details would contain the location, capacity, date and time and whether the event is public or private. |
| **Join Event** | Join Event is the way a user can join a private or public event. |
| **Create Event** | This is used for creating an event at a restaurant |
| **Search Bar** | Search bar can be used by the user to search and filter to find the restaurants they prefer. |
| **User** | The person who uses the app. They must be within Singapore and can either register and login or use it as a guest. |
| **Validate** | The user can validate their login credentials using Google or Facebook authentication. On top of that they can also login by creating a separate account on the app |
| **Public Event** | These are the events that can be joined by any logged in user |
| **Private Event** | These are the event that can only be joined by a user if they are friends with the creator of the event |

# Appendix B: Analysis Models



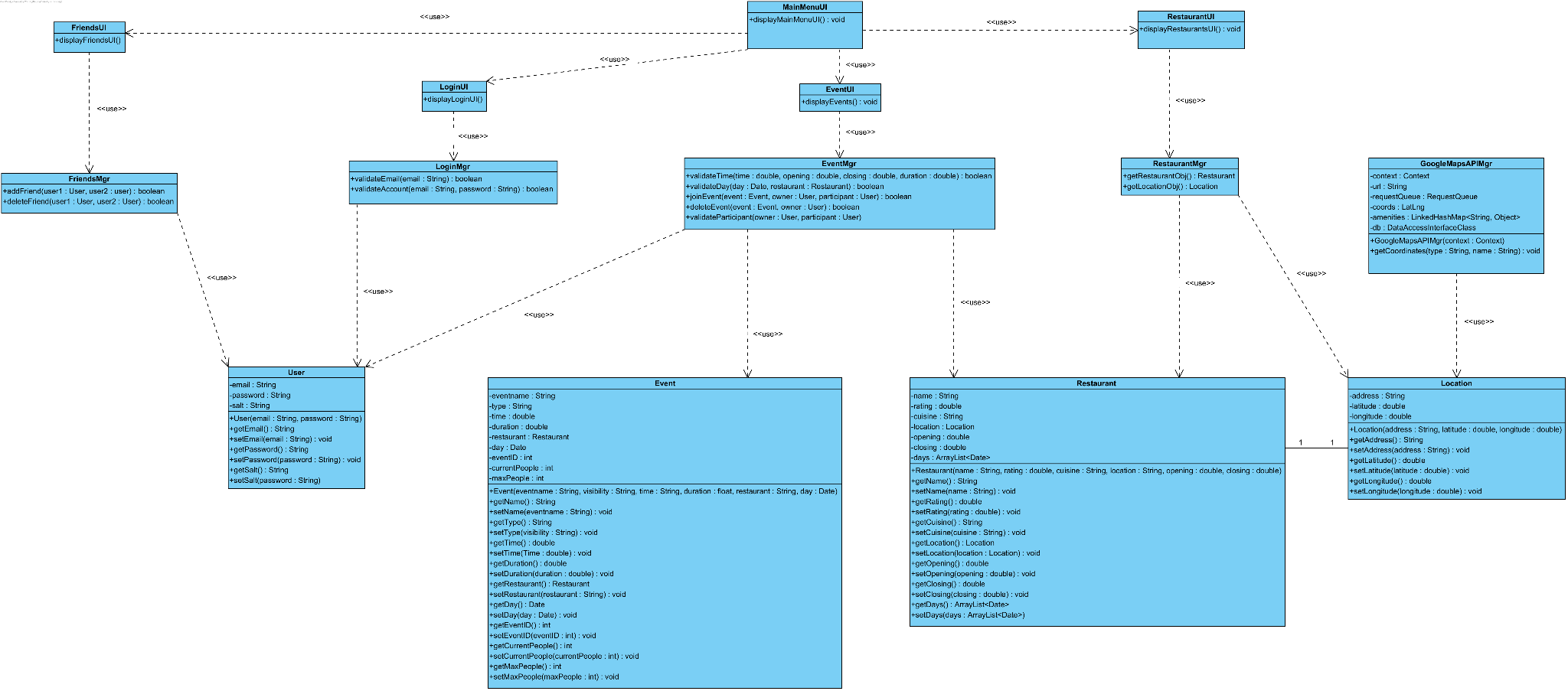
## USE CASE DIAGRAM

## 

## DIALOG MAP

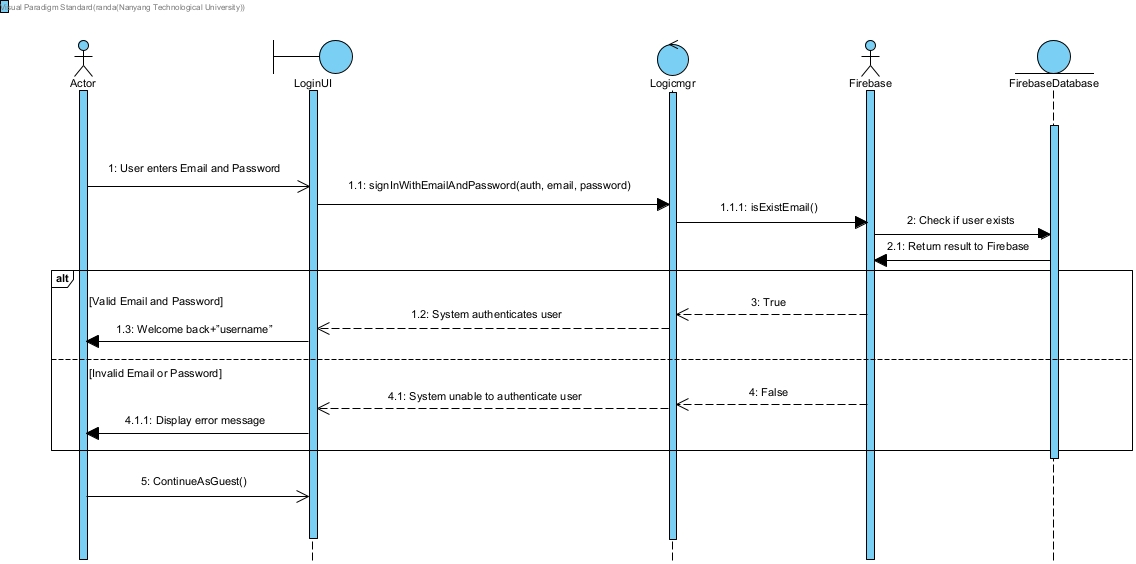
## 

## SYSTEM ARCHITECTURE

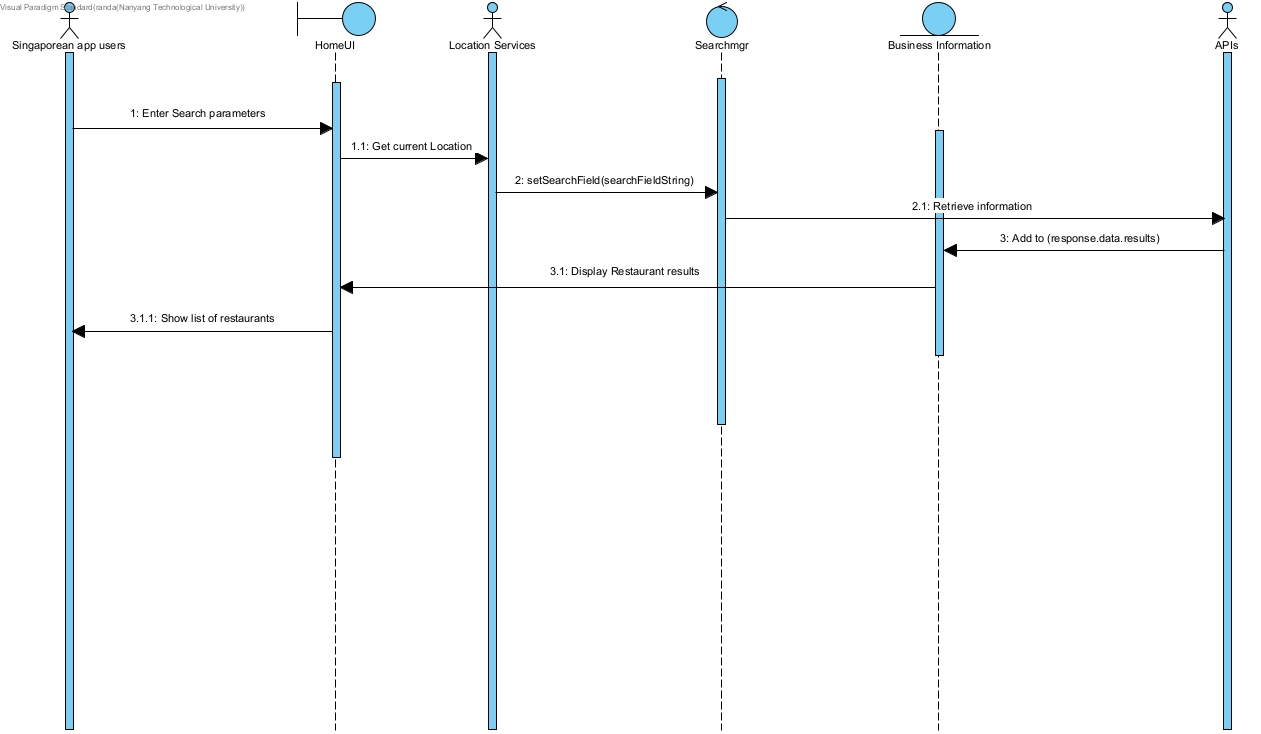
****

## DESIGN CLASS DIAGRAM

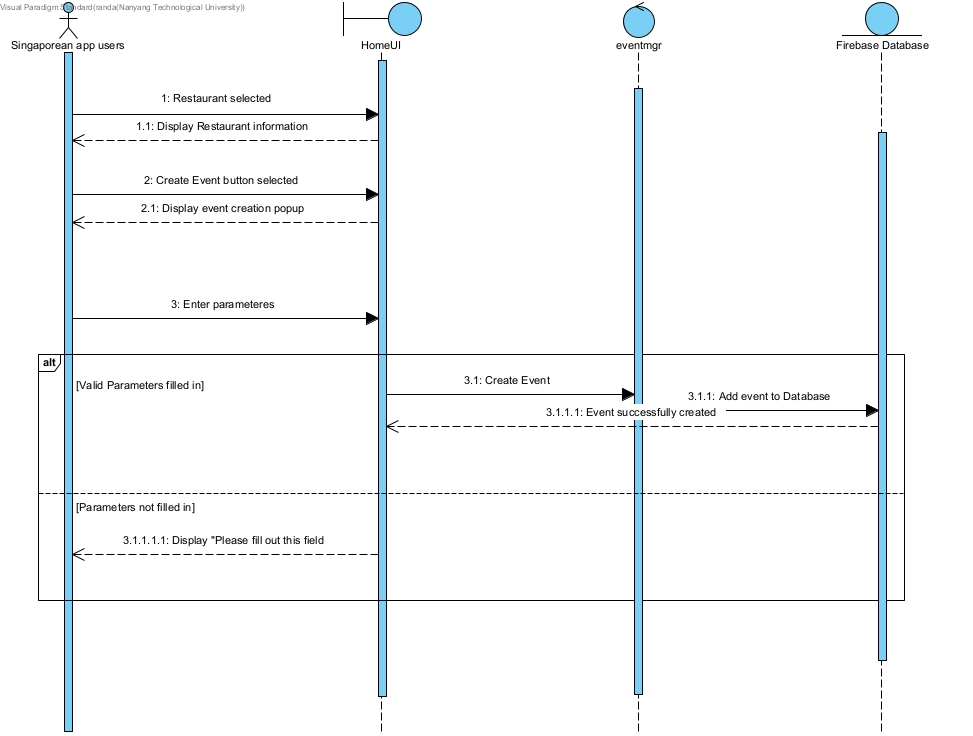
## SEQUENCE DIAGRAM - REGISTRATION

****

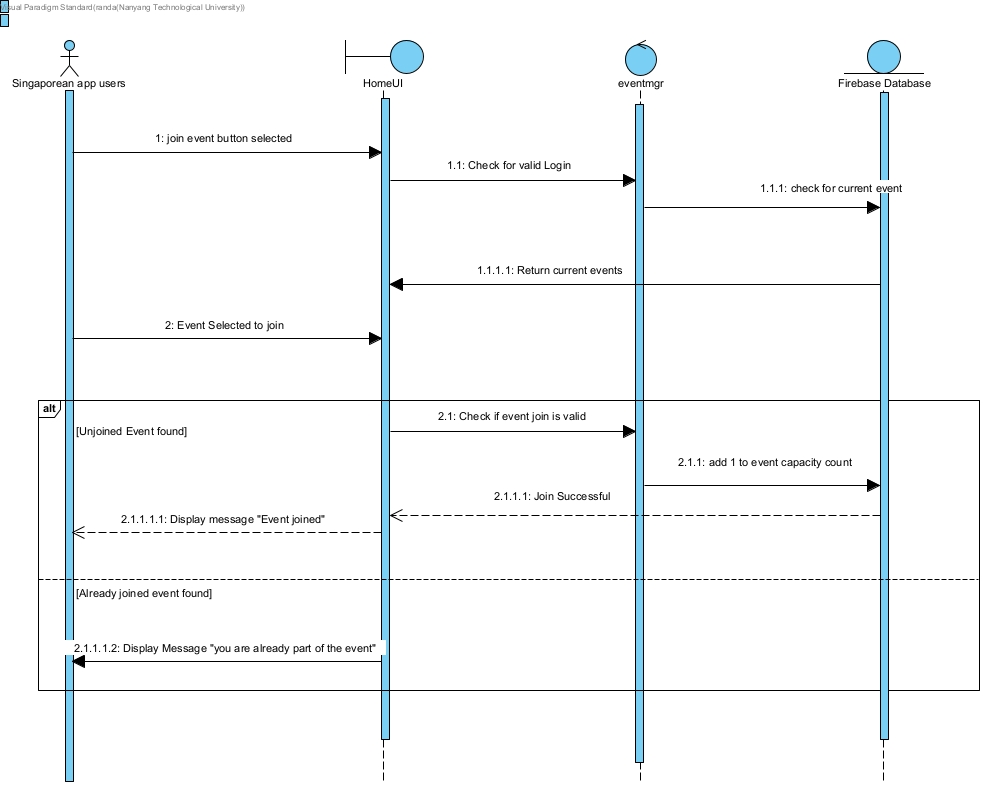
**SEQUENCE DIAGRAM - LOGIN**

****

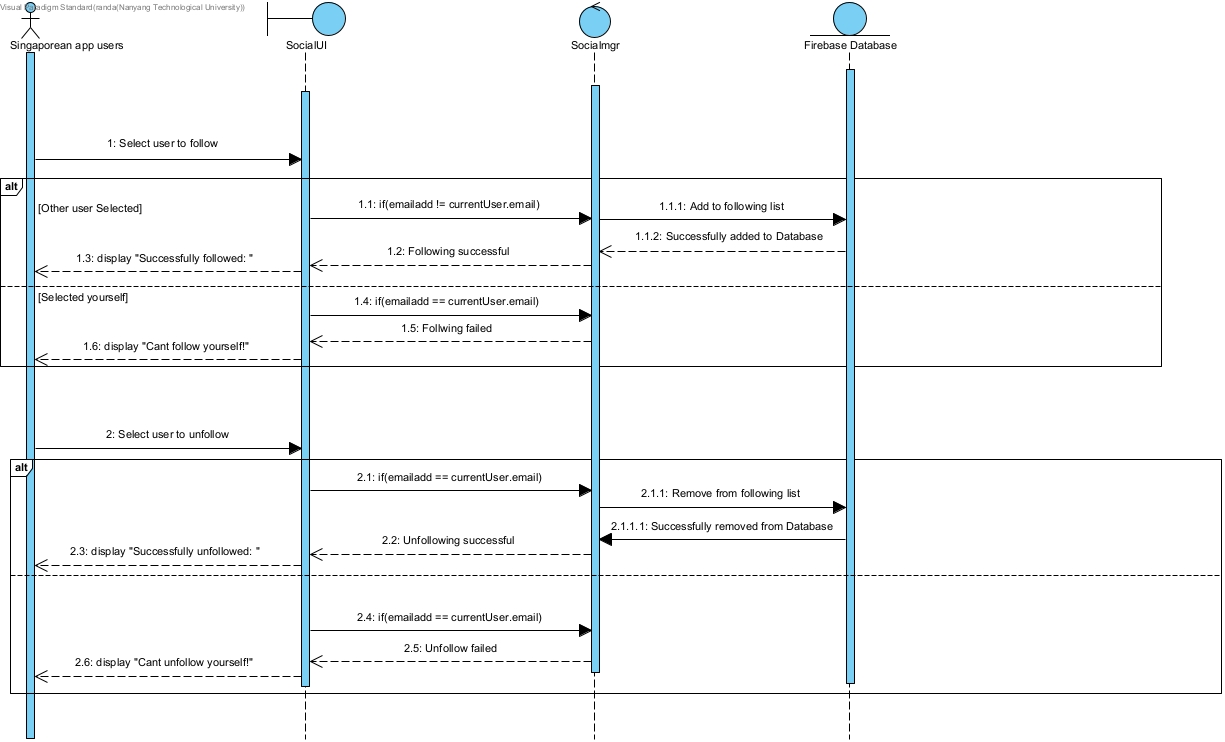
**SEQUENCE DIAGRAM - SEARCH**

****

**SEQUENCE DIAGRAM - CREATE EVENT**

****

**SEQUENCE DIAGRAM - JOIN EVENT**

****

**SEQUENCE DIAGRAM - FOLLOWING AND UNFOLLOWING OTHER USERS**