Software Requirements Specification

for

Jiak!

Version 5.0

Prepared by

Lim Xin Wei (U2240093B),
Seah Kah Yen (U2240401D),
Le Yanzhi (U2240481K),
Chia Jia En (U2221939E),
Tay Wei Yang (U2221303F),

Nanyang Technological University, MACS2 (group 3) The Buggers

Table of content

Revision History

Name	Date	Reason For Changes	Version
Lim Xin Wei	30/1/2024	Initial documentation specifying software requirements	1.0
Lim Xin Wei	25/2/2024	Refinements for lab 2 submission	2.0
Lim Xin Wei	1/4/2024	Refinements for lab 3 submission	3.0
Lim Xin Wei	15/4/2024	Refinements for lab 4 submission	4.0
Lim Xin Wei	20/4/2024	Refinements for lab 5 submission	5.0

1. Introduction

1.1. Purpose

Our mobile application, Jiak!, has the purpose of revolutionising the way food enthusiasts discover dining experiences. This will be achieved by seamlessly integrating cutting-edge geolocation technology and proximity detection with an emphasis on a user-centric design into the app.

With a focus on personalised discovery, Jiak! seeks to uncover and highlight local, highly rated merchants, creating a platform for diners with adventurous palates. Coupled with our friends and party feature, we encourage users to connect with their friends and discover new food experiences together.

Jiak aims to empower both diners and merchants, all while prioritising values of privacy, togetherness, and user preferences. Thus, enriching the dining landscape and fostering a community of explorers who take delight in finding and sharing hidden gems. The overarching goal is to enable diners to embark spontaneously on culinary adventures, turning everyday meals into extraordinary experiences.

1.2. Document Conventions

Software Requirement Specification Format: This document follows IEEE standard[1]. Priorities of higher-level requirements are inherited by detailed level requirements.

Level 1 Heading: Font Family: Times, Font Size: 18, Font Weight: Bold
Level 2 Heading: Font Family: Times, Font Size: 14, Font Weight: Bold
Level 3 Heading: Font Family: Times, Font Size: 12
Content: Font Family: Times, Font Size: 11

Further conventions on the terms used could be found at Appendix A – Data Dictionary Section

1.3. Intended Audience and Reading Suggestions

This document is intended for all stakeholders including users of the mobile application, software developers of the application, project managers, testers and collaborative partners involved.

This document will also contain the application's functions and reasoning for design purposes including examples of different pages. A detailed description of each feature will be provided.

[1] 829-2008 - IEEE Standard for Software and System Test Documentation. (18 July 2008). Electronic ISBN:978-0-7381-5746-7. https://ieeexplore.ieee.org/document/4578383

This document is to be read sequentially by testers, users and collaborative partners. For developers and project managers, a focus on the application features and its functional and non-functional requirements should be prioritized to meet user needs.

1.4. Product Scope

Our mobile application can be downloaded and users on both the Android and IOS platforms. The application will be user-friendly and provide a smooth experience for users to discover and rate food when they are travelling outside alone or with friends.

1.5. References

1) React Native Framework - https://reactnative.dev/
2) React Native Expo - https://expo.dev/
3) Google Maps API - https://developers.google.com/maps
4) RESTful API - https://restfulapi.net/
5) Firebase - https://rnfirebase.io/

2. Overall Description

2.1. Product Perspective

Jiak! is a novel mobile application that allows users to discover new restaurants involuntarily while they are travelling alone or with their friends. If the user has enabled notifications and is within proximity of a restaurant that caters to the user's preferences — which includes their favourite types of cuisine and ratings of the restaurants; a notification would be sent. This simulates the traditional spreading through 'word of mouth' of places that they would normally not know of. Users can choose to click on restaurant icons in the application to see recommended restaurants too.

We will use the Google Maps API to uncover popular or underrated restaurants near the user. Should the user visit the recommended restaurant, they will be asked to review their experience in the restaurant. The user feedback collected will be used to enhance our restaurant recommendations for others. When discovering food with friends, users can create party to take note of cost and split cost amongst friends.

Lastly, the feedback feature within Jiak! serves as a channel for users to provide valuable feedback, suggestions, and insights to the development team. By soliciting user input, the app aims to continuously improve and refine its features, user experience, and overall quality.

2.2. Product Functions

Jiak! has the following functions/features:

1. Allow users to register for an account to log in.
2. View user profile to see details such as username and food preferences.
3. Edit user profile to change details such as username and food preferences.
4. View nearby recommended restaurants via notification/restaurant icons in the app.
5. Rate and review restaurants after dining in.
6. Edit/delete reviews for restaurants that they have dined at previously.
7. Generate/Use referral codes for restaurant discounts.
8. Add friends using referral codes
9. Add friends to parties
10. Exit and leave current party that users are in
11. The host can split cost with party.

2.3. User Classes and Characteristics

- **Local Explorers**: Users who use the app to discover new dining experiences within Singapore. They have an open mind to try out new food and value up-to-date information on new and trending local eateries.
- **Food Enthusiasts**: These are users with a keen interest in culinary experiences, seeking not just any food but the best or most unique offerings. They might use the app intensively to research and plan dining experiences, appreciating detailed reviews.
- **Dining Groups:** Group of friends can collaborate and enrich their dining experiences with friends and forge lasting connections within the vibrant culinary landscape of Singapore.

- **Tech-Savvy Users:** Users in this class prioritize the app's integration with other services, such as social media or maps to seek nearby restaurant recommendations. They might do it out of curiosity and see whether the application can provide good recommendations.

2.4. Operating Environment

Product Environment for Jiak!:

The version of the Android Operating System must be Android 5.0 and above.

The version of the IOS Operating System must be IOS 13.4 and above.

Development Environment for Jiak!:

Development Environment	Description
Front-end: React Native with Expo for Mobile Application	React Native is a framework for building mobile applications using JavaScript and React, enabling developers to create cross-platform apps with native-like performance using a single codebase. Expo is a platform and framework that layers on top of React Native, offering a managed environment that simplifies the development process by providing a set of pre-configured elements and APIs. This allows for rapid development and easy deployment of our mobile applications as it has hot-reload and live reload.
Back-end & Database: Firebase	Firebase is considered a backend-as-a-service (BaaS) and supports authentication for usernames and passwords. It also has a real-time database that allows us to store and sync data securely.

2.5. Design and Implementation Constraints

1. Our application requires Google Maps API calls. With limited free credits to use for each API call, we might run out of credits before the entire development process is finished. This can be resolved by upgrading the service when our application goes live.
2. Users with older versions of Android or IOS might face issues when using the application.
3. Firebase supports up to 5GB of cloud storage. With good use of data storage, this constraint should not be major.

2.6. User Documentation

We will provide user manuals that include our screens and functionality in our user manuals that will be provided to stakeholders during operation.

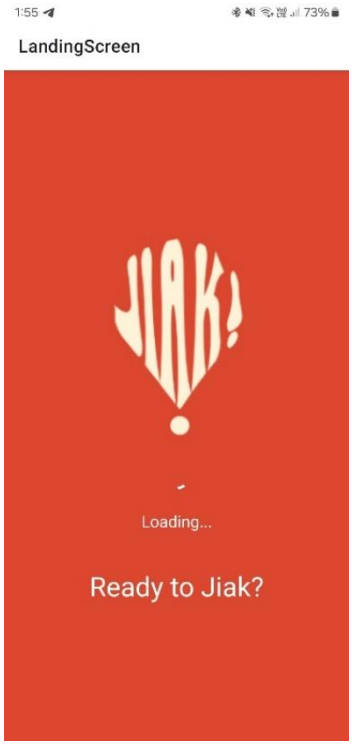
2.7. Assumptions and Dependencies

1. This mobile application assumes that users have their live location turned on and have stable internet connection.
2. The application is dependent on external Google Maps API to obtain restaurant information near users.
3. We assume that the data obtained from Google Maps API is accurate.
4. We assume that users have email when signing up for our application.
5. We assume that users have friends who have the same application when they use our "Add Friend" feature.

3. External Interface Requirements

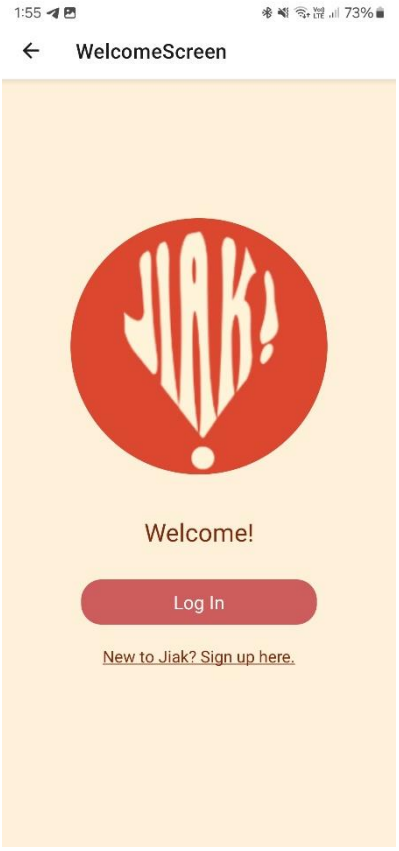
3.1. User Interfaces

3.1.1. Landing Page



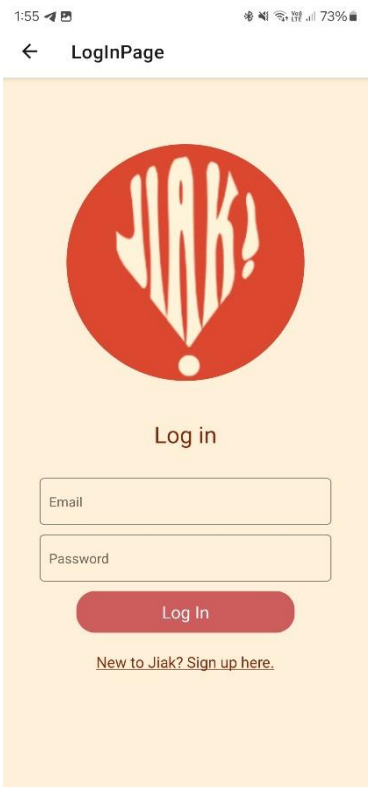
1. This page appears upon the user entering the app. It will then swipe to landing page 2 (3.1.2) after 3 seconds.

3.1.2. Welcome Screen



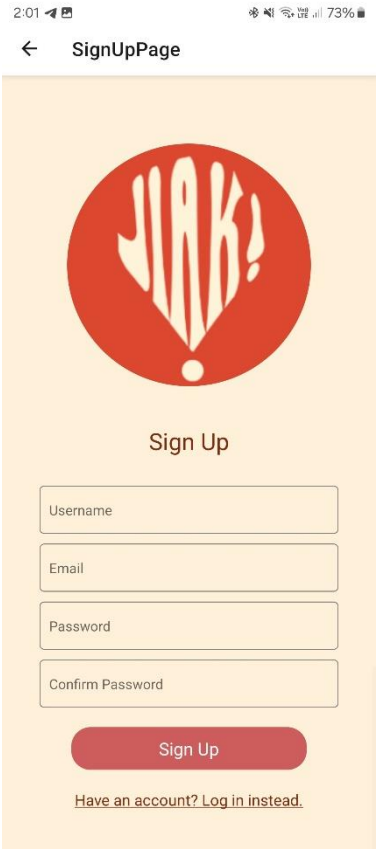
1. Diners can click on the log in button if they already have an account. This will bring them to the log in page (3.1.3).
2. Diners can create an account by clicking the underlined “New to Jiak? Sign up here.”, which brings them to the sign up page (3.1.4).

3.1.3. LoginPage



1. The diner has to input their email and password. Upon logging in, they will be brought to the home page (3.1.7).
2. Diners without an account can also go to the sign-up page from this page by clicking on the underlined “New to Jiak? Sign up here.”

3.1.4. Sign-up Page



1. New diners will have to input their username, email, password, and confirmation of password before clicking Sign-up, which will bring them to the preferences pages (3.1.5).
2. Diners can return to the log in page by click on underlined “Have an account? Log in instead.”

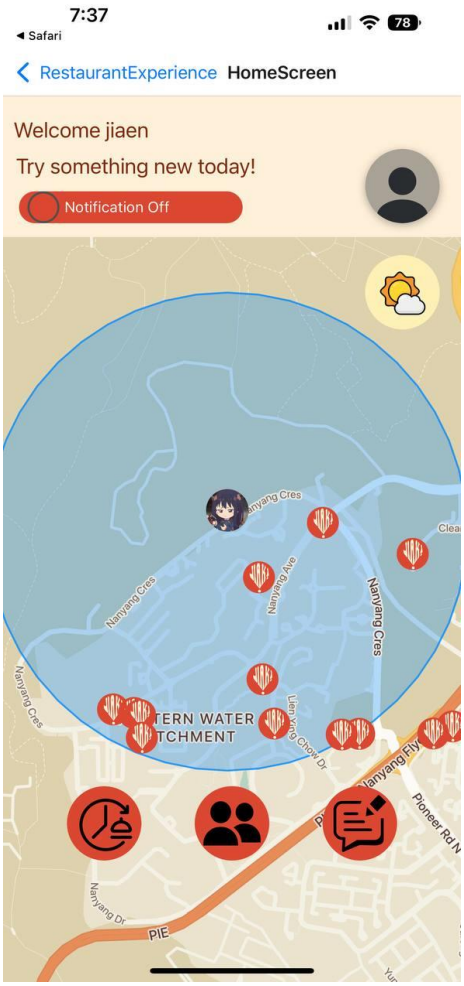
3.1.5. Preferences (Select Cuisine)

1. Diners can click on the icons to select the cuisines they want to receive notifications for.
2. Diners will have to select a minimum of 3 cuisines. Otherwise, the “next” button will not bring them to the next page.
3. The icons will be highlighted upon selection and can be deselected or selected by tapping on the icons.
4. Clicking next will bring the diners to continue selecting their preferences (3.1.6).

3.1.6. Preferences (Restaurant Experience)

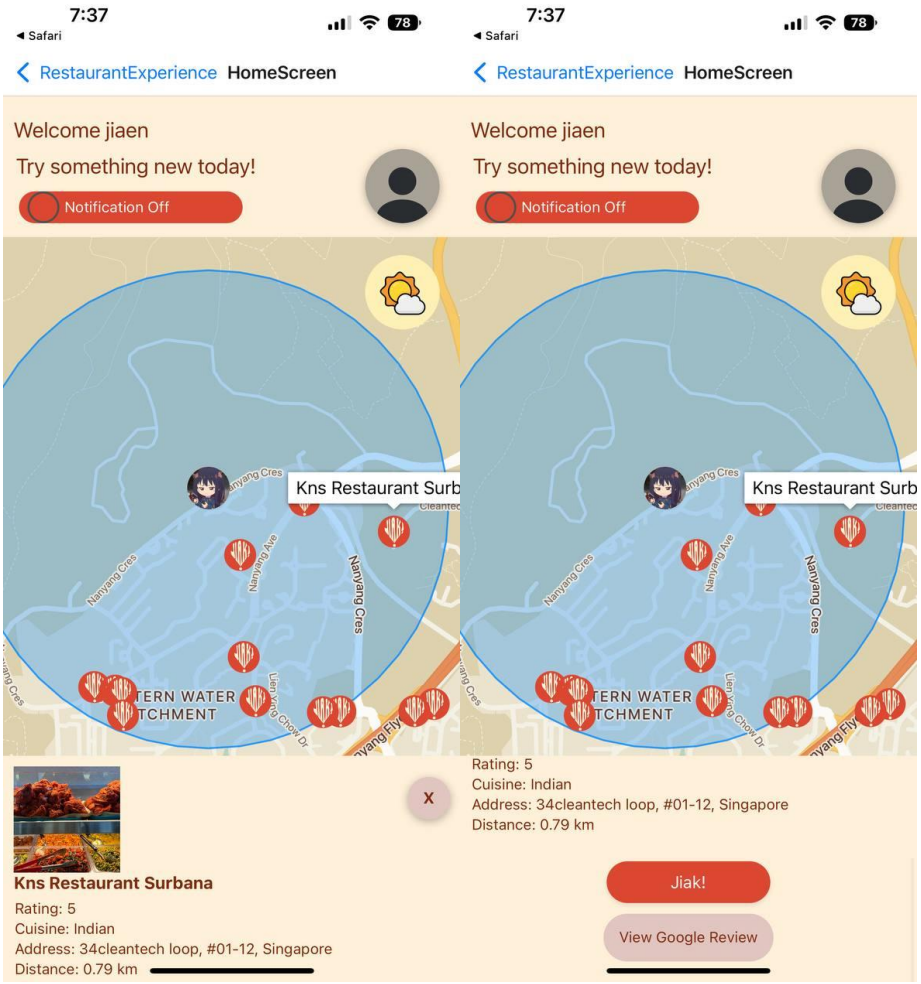
1. The restaurant ratings and proximity bars are sliders, which will allow diners to choose their desired range for both bars.
2. Diners will be allowed to toggle between “open now” or “show all”, which indicates if they want results to only show currently operating businesses or both closed and open businesses.
3. Clicking next will end their sign up and bring them to the home page (3.1.7)
4. Diners can return to the previous preferences page by clicking the back arrow button.

3.1.7. Home Page



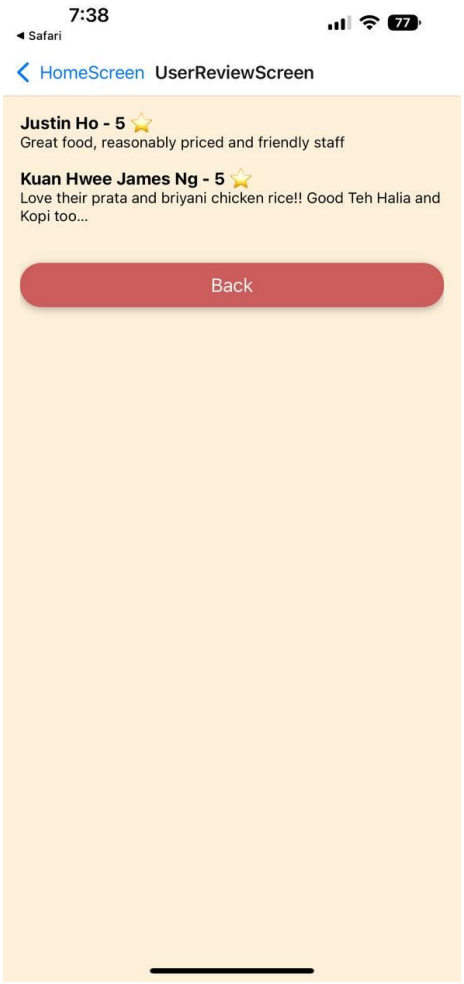
1. Diners can toggle between “notifications on” and “notifications off”, depending on their preference. If notifications are toggled off, they will not receive notifications.
2. Diners can view their profiles by clicking on the profile icon beside the notifications toggle.
3. Diners can view their restaurant history by clicking on the bright red “Jiak!” icon at the bottom right of the page.
4. As diners move, their avatar will move on the map.
5. Diners can click on the “Jiak!” icons on the map to view more information on the restaurant (3.1.8).
6. Diners can view Dining History by clicking on the bottom left most icon on the Home Page.
7. Diners can access Social by clicking on the bottom middle icon of the Home Page.
8. Diners can write app feedback by clicking the bottom righ most icon on the Home Page.

3.1.8. View Restaurant Page



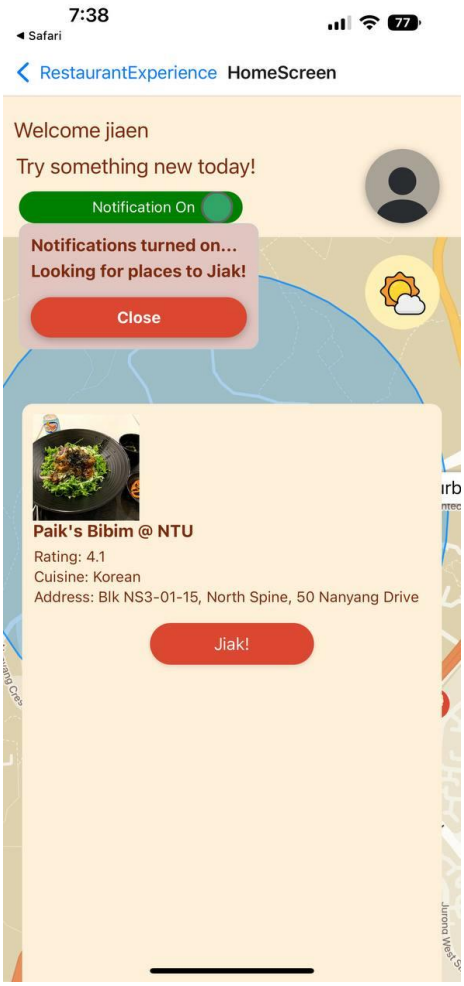
1. Clicking on a “Jiak!” icon on the map will cause this pop up.
2. Diners can click on the “X” at the top right corner of the pop up to return to the home page.
3. Diners can choose to eat at this spot by clicking the “Jiak!” button.
4. Diners can choose to view the restaurant’s google reviews by clicking on “View Google Reviews”.

3.1.9. View Restaurant Review Page



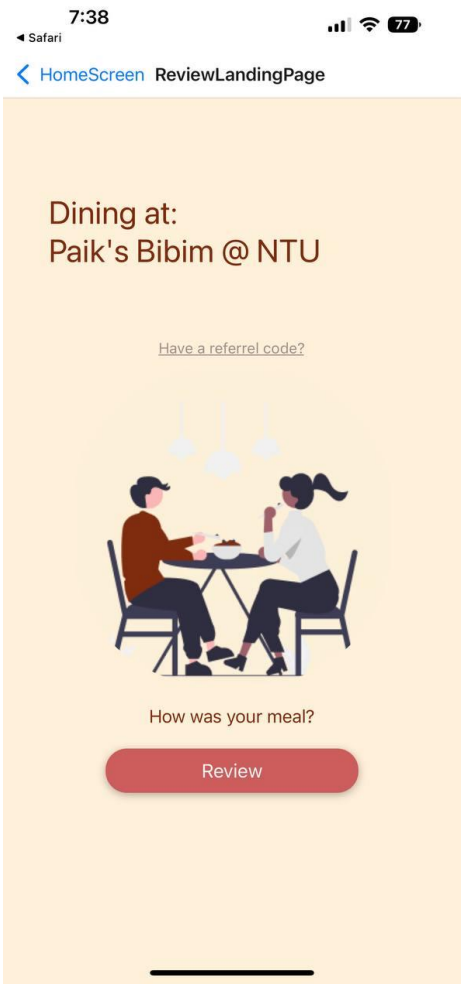
1. Reviews can be scrolled through.
2. Diners can return to the home page by clicking on the “Back” button.

3.1.10. Notification Page



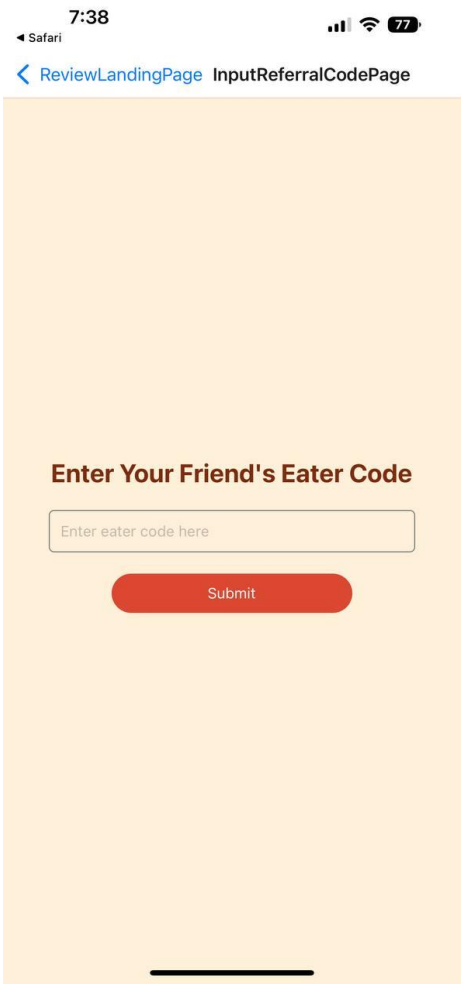
1. This page pops up when notifications are toggled on, and when the restaurant nearby matches the diner's preferences.
2. Diners can choose to dine at the restaurant (accept the suggestion) by clicking on the "Jiak!" button.

3.1.11. Dining Page



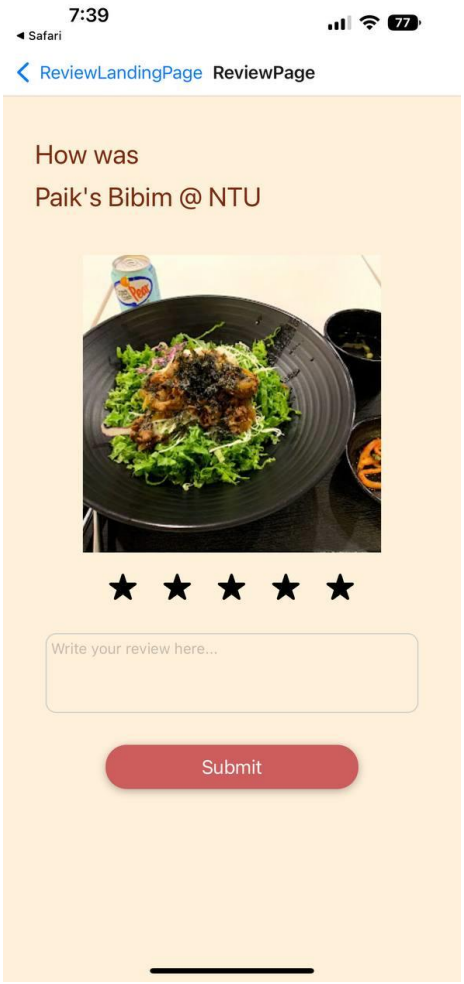
1. Diners will be brought to this page when they choose to dine at a spot.
2. Diners will not be allowed to exit this page.
3. Diners can review the restaurant after dining by clicking the “review” button.
4. Diners can input their referral codes by clicking the underlined “Referral Code” (3.1.12).

3.1.12. Use Referral Code



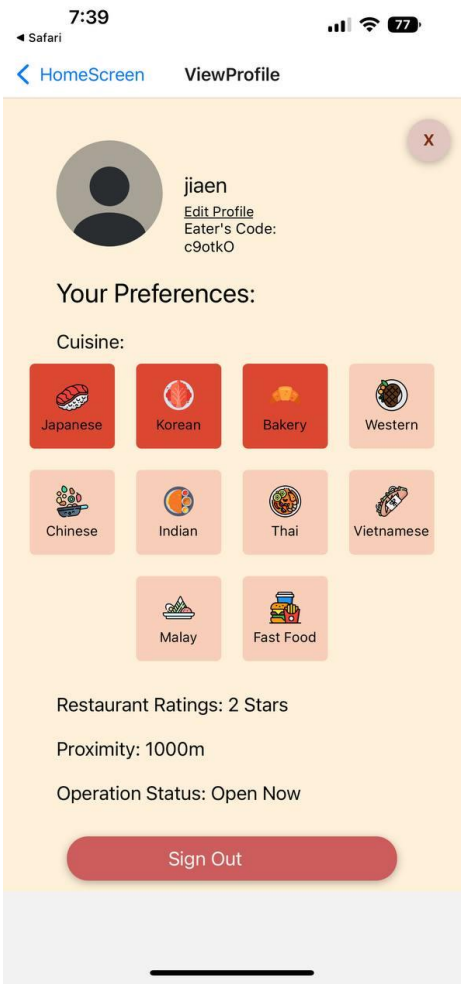
1. Users can enter their referral codes in the box.
2. Users can submit the referral code by pressing the “Submit” button.

3.1.13. Review Page



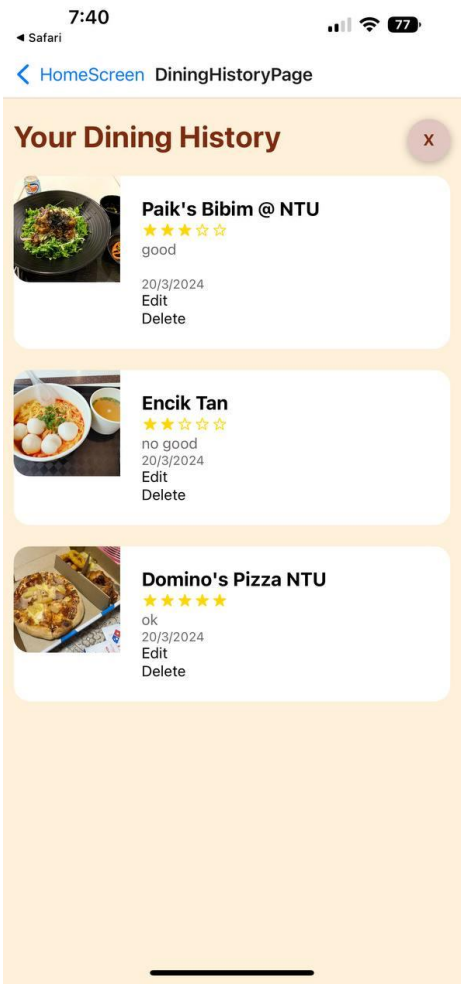
1. Users can rate the restaurant by clicking on the number of stars and by typing in the text box.
2. Clicking on the “Submit” button submits their review.
3. User will return to the home page upon submission.

3.1.14. Profile Page



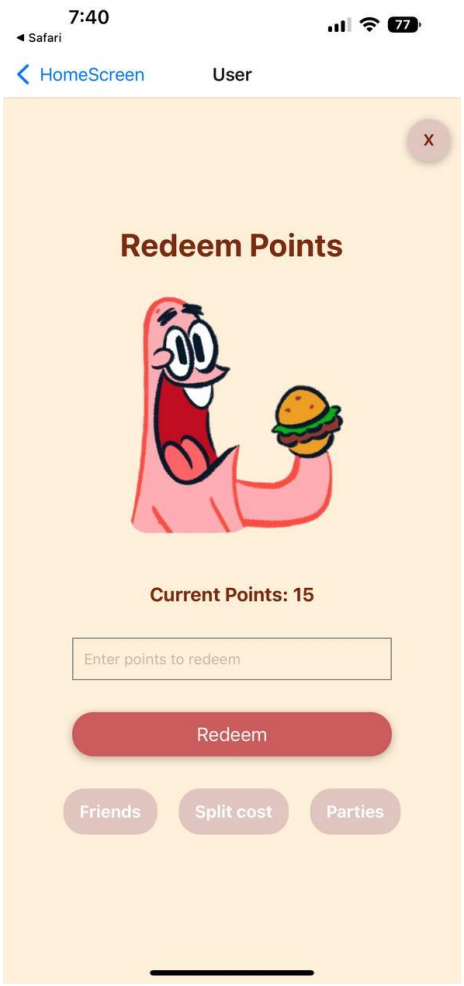
1. Diners can log out by clicking the “Sign Out” button.
2. Diners can return to the home page by clicking on the “X” button.
3. “Edit Profile” brings the diner to the preferences page (3.1.5), where diners can edit their preferences.

3.1.15. History Page



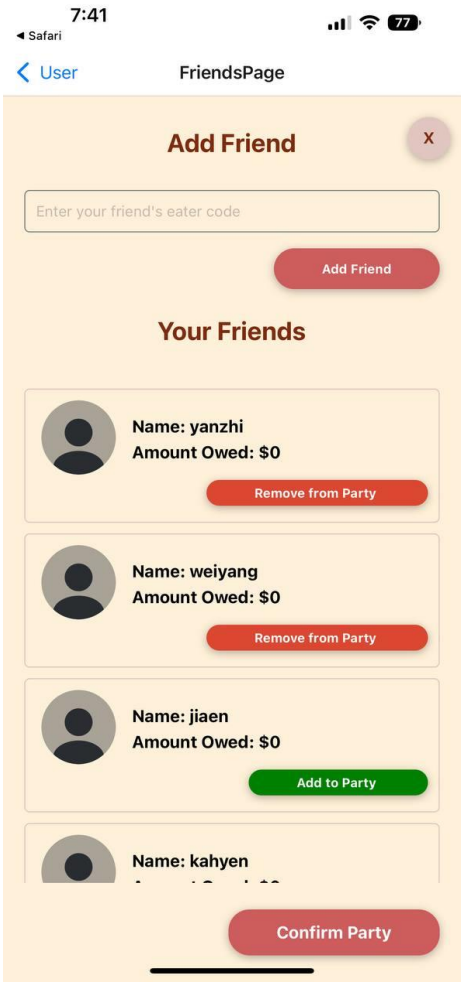
1. Diners can edit their review by clicking on “Edit” .
2. Diners can delete their review by clicking on “Delete”.
3. Diners can return to the home page by clicking the “X” button.

3.1.16. User Page



1. Diner can redeem points by entering the points they wish to redeem in the box and clicking “Redeem”.
2. Diners can go to Friends page by clicking on “Friends” (3.1.17).
3. Diners can split costs by clicking on “Split Cost” (3.1.18).
4. Diner can view Parties by clicking on “Parties” (3.1.19).

3.1.17. Friends Page

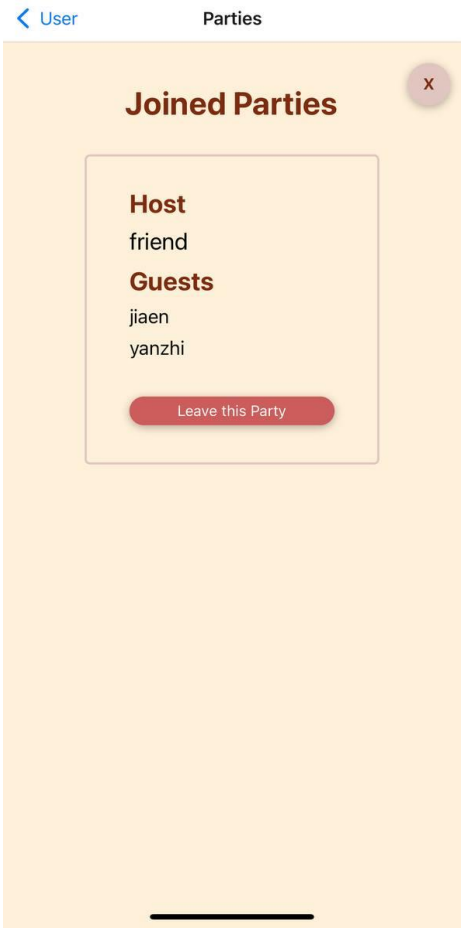


1. Diners can add friends by entering their friend's code and clicking "Add Friend".
2. Diners can add or remove friends from their party by clicking on the "Add to party" or "Remove from Party", then the "Confirm Party button.
3. Diners can return to User Page by clicking on the "X" button.

3.1.18. Split Cost Page

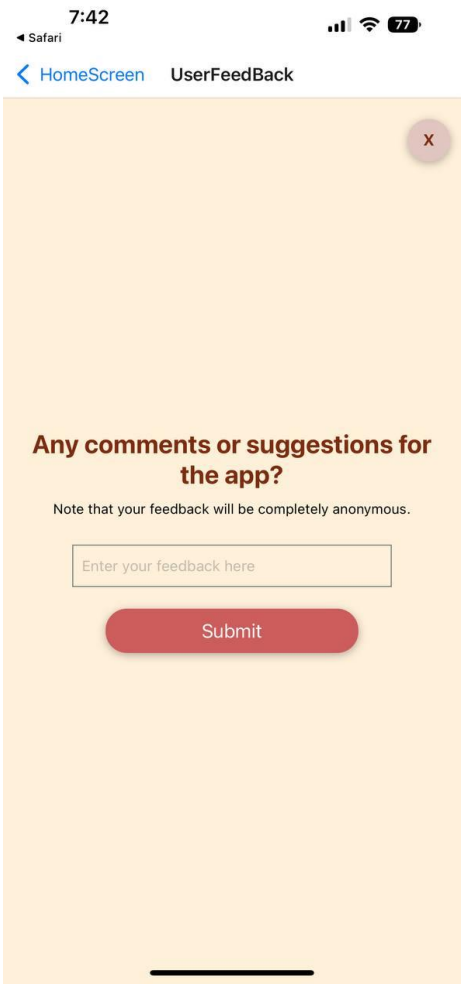
1. Diners can view their current party where they are hosting.
2. Diners can split costs between party members by entering the total cost and clicking “Split Cost”.
3. Diner can return to User Page by clicking the “X” button.

3.1.19. Parties Page



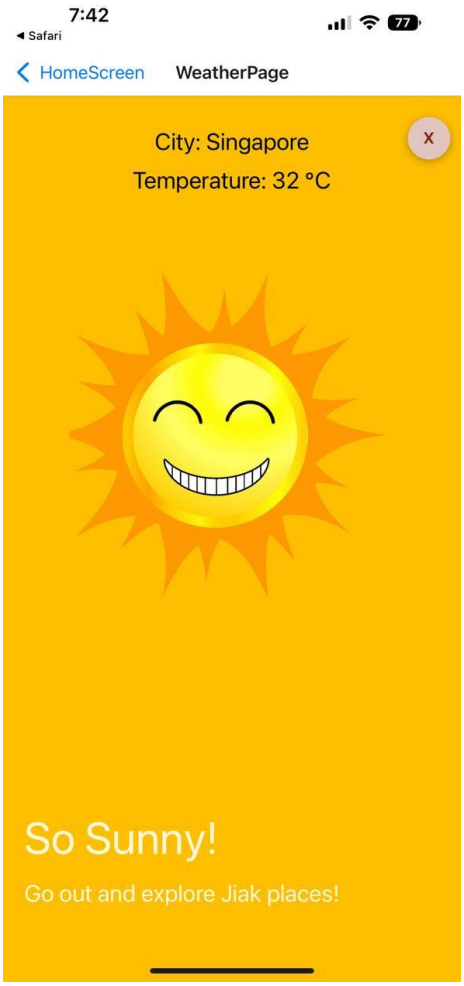
1. Diners can view all their parties in this page.
2. Diners can leave parties by clicking “Leave this Party”.
3. Diners can return to User Page by clicking the “X” button.

3.1.20. User Feedback Page



1. Diners can leave feedback for the app by typing it in the textbox and clicking “Submit”.
2. Diners can return to Home Page by clicking on the “X” button.

3.1.21. Weather Page



1. Diners can view the current weather and temperature.
2. Diners can return to Home Page by clicking the “X” button.

3.2. Hardware Interfaces

1) The mobile application requires a mobile device that supports the Android or IOS platform.
2) The operating system must be up to date and the device must have internet connection capabilities to send and receive data.
3) The device must have GPS function and live location turned on so the application can send recommendations based on user location.

3.3. Software Interfaces

The architecture of this application comprises both front-end and back-end components, leveraging modern frameworks and platforms to deliver a seamless user experience.

- Front-end
    - Built using React Native, a mobile application framework which enables the development of natively rendering mobile apps for iOS and Android. Additionally, we integrate React Native with Expo, a comprehensive framework and platform for universal React applications. Expo streamlines the development process by offering a suite of tools and services that simplify the build, deployment, and iteration of iOS, Android, and web apps from a single JavaScript codebase. With Expo, developers can access a wide range of features and APIs, including push notifications, in-app purchases, and device sensors, enhancing the functionality and user experience of the application.

- Back-end
    - For the back-end infrastructure, we leverage Firebase, a comprehensive platform provided by Google, to power the server-side components of the application. Firebase offers a suite of integrated services that facilitate the development of scalable and secure back-end solutions, including:

        - Realtime Database: Firebase Realtime Database is a cloud-hosted NoSQL database that enables the synchronization of data in real time across clients. It provides a flexible and scalable solution for storing and retrieving structured data, supporting collaborative features such as live chat, multiplayer gaming, and real-time analytics.

        - Authentication: Firebase Authentication offers a secure and easy-to-use authentication system that allows users to sign in to the application using email and password, phone number, or popular identity providers like Google, Facebook, and Apple. It provides robust identity verification and management capabilities, ensuring the security of user accounts and data.

3.4. Communications Interfaces

1) Feedback Submission:
    - The Feedback Submission interface allows users to provide valuable feedback, suggestions, or bug reports directly within the app. Users can fill out a form with

relevant details and submit their feedback, which is then transmitted to the server for processing.

2) API Integration:
    - API Integration enables seamless communication between the mobile app and external services or APIs, such as restaurant review platforms, Google API mapping services and Firestore queries.
3) Authentication:
    - Authentication and Authorization interfaces like Log In function to ensure secure access to the app's features and data by verifying user identities and permissions.

4. System Requirement

Use Case Hierarchy

1.0 AccountRegistration

- 1.1 RegistrationValidation

2.0 Login

2.1 SignInWithEmailandPassword

3.0 ViewProfile

- 3.1 ViewReferalCode
- 3.2 SignOut

4.0 EditProfile

- 4.1 CusinePreferencesSetting
- 4.2 RatingSetting
- 4.3 OperationStatusSetting
- 4.4 ProximitySetting

5.0 RetrieveInformation

- 5.1 NotificationToggle
  - 5.1.1 ViewRestaurants
    - 5.1.1.1 ViewReviews
    - 5.1.1.2 Jiak!AtRestaurant
      - 5.1.1.2.1 UseReferralCode
      - 5.1.1.2.2 ReviewRestaurant

6.0 DiningHistory

- 6.1 EditReview
- 6.2 DeleteReview

7.0 Social

- 7.1 RedeemPoints
- 7.2 SplitCost
- 7.3 Friends
  - 7.3.1 AddFriends
  - 7.3.2 AddtoParty
  - 7.3.3 ViewFriends
  - 7.3.4 ConfirmParty
  - 7.3.5 ViewOwedAmount
- 7.4 Parties
  - 7.4.1 LeaveParty
  - 7.4.2 ViewParties

8.0 UserFeedBack
    8.1 SubmitFeedBack
9.0 WeatherPage

Only main use cases (i.e. x.0) are included in this Software Requirement Specification for Jiak! report. For the use case description of all use cases, please refer to our Use Case for Jiak! Report.

4.1 Account Registration

4.1.1 Description and Priority

The registration feature allows users to register for an account. This feature is of high priority as registering for an account is a prerequisite to logging in and accessing the functions of the application.

4.1.2 Stimulus/Response Sequences

Use Case ID	1.0		
Use Case Name	AccountRegistration		
Created by	Le Yanzhi	Last Updated by	Le Yanzhi
Date Created	4/2/2024	Date Last Updated	27/3/2024

Actor	Diner
Description	First time Diners must register for an Account. Diners must enter their Username, Email, Password and confirm their Password to register for an account. Upon successful creation, Diners would then proceed to Use Case EditProfile (3.1).
Preconditions	1. Diner must have an Email. 2. The Email should not have been registered with another account previously.
Postconditions	1. Diner must have an Account 2. Diner must have entered their food preferences.
Priority	High
Frequency of Use	Once per Diner
Flow of Events	1. Diner clicks on “New to Jiak? Sign up here.” link to register for an account. 2. System prompts user to input Username, Email, Password, Confirm Password. 3. Diner inputs Username, Email, Password, Confirm Password. 4. Diner clicks “Sign Up” button. 5. System validates the information entered using included Use Case RegistrationValidation (1.1). 6. If RegistrationValidation returns True, system creates he Account 7. System updates account information. 8. System displays “Account Created Successfully.” 9. System directs Diner to Use Case EditProfile (3.1). 10. System directs Diner to Home Screen.

Alternative Flows	1.0.AC.1.0: If RegistrationValidation returns UsernameTakenException 1. System displays “Username already exists, please choose a different username or login with an existing account.” 2. System returns to Step 2 1.0.AC.1.1: If RegistrationValidation returns EmailTakenException 1. System displays “Email already exists, please choose a different Email or login with an existing account.” 2. System returns to Step 2 1.0.AC.1.2: If RegistrationValidation returns PasswordInvalidException 1. System displays “Password must be at least 8 characters long and include at least one uppercase character, one lowercase character, one number, and one special character.” 2. System returns to Step 2 1.0.AC.2.0: If Diner clicks on “Have an account? Log in instead.” button 1. System returns to Login Page 1.0.AC.3.0: If Diner did not fill up every information (Username, Email, Password, Confirm Password) when clicking “Sign Up” button. 1. System displays “Please fill in all fields”. 2. System returns to Step 2. 1.0.AC.4.0: If Password and confirm Password does not match 1. System displays “Password does not match” 2. System clears Password and confirm Password text box 3. System returns to Step 2.
Exceptions	NIL
Includes	1.1 RegistrationValidation 4.0 EditProfile
Special Requirements	NIL
Assumptions	NIL
Notes and Issues	NIL

4.1.3 Functional Requirements

1. Each user must have an account.

1.1. Users who download the app for the first time must sign up for an account as a Diner.

1.1.1. Diner must input his unique Username.

1.1.1.1. System must prompt the Diner to change his Username if it has already been taken (not case sensitive).

1.1.2. Diner must input his unique Email.

1.1.2.1. System must prompt the Diner to change his Email if it has already been taken.

1.1.2.2. System must prompt the Diner to enter a valid Email if the Email input

1.1.3. Diner must input his Password.

1.1.3.1. Passwords must be at least 8 characters long, at least 1 special character, 1 uppercase character, 1 lowercase character and 1 number.

1.1.3.2. System must prompt the Diner to change his Password if it does not adhere to the requirements.

1.1.4. Diner must re-enter his Password.

1.1.4.1. System must prompt the Diner to check his Password if the re-entered the Password does not match Password.

4.2 Login

4.2.1 Description and Priority

The login feature enables existing users to access their accounts by entering their email and password and clicking on the “login” button. This feature is crucial as it secures user access and personalizes the user experience by granting access to their profiles and preferences within the application.

4.2.2 Stimulus/Response Sequences

Use Case ID	2.0		
Use Case Name	Login		
Created by	Le Yanzhi	Last Updated by	Le Yanzhi
Date Created	4/2/2024	Date Last Updated	27/3/2024

Actor	Diner
Description	Diner who already has an account will be able to log in to the system and access the functionalities in the Landing Page
Preconditions	1. Diner should have an account created from AccountRegistration Use Case
Postconditions	2. Diner’s login information (Email and Password) has been verified and is successfully logged in to the account
Priority	High
Frequency of Use	Once per diner
Flow of Events	1. Diner press “Log In” Button 2. System prompts Diner to input Email and Password 3. Diner input his Email and Password 4. System checks that the Email and Password matches in Database 5. System displays “Log in Successfully.” 6. System directs the Diner to the Home Screen.
Alternative Flows	2.0.AC.1.0: If Email does not match any entries in Database 1. System displays “Theres no user exists with that email” 2. System returns to Step 2. 2.0.AC.2.0: If Diner did not fill up every information (Email, Password) when clicking “Log In” button. 1. System displays “Please fill in all fields.” 2. System returns to Step 2. 2.0.AC.3.0: If Diner entered an invalid email address format 1. System displays “The email address is invalid.” 2. System returns to Step 2.

	2.0.AC.4.0: If Email and Password does not match any entires in Database. 1. System displays “The password is invalid.” 2. System returns to Step 2. 2.0.AC.5.0: If Diner has tried to log in too many times. 1. System display “Access to this account has been temporarily disabled due to many failed login attempts. Try again later.” 2. System returns to Step 2.
Exceptions	NIL
Includes	NIL
Special Requirements	NIL
Assumptions	NIL
Notes and Issues	NIL

4.2.3 Functional Requirements

1. Diner must be able to log in to his account.

    1.1. Diner must input his Email.

        1.1.1. System must validate the Email with Database.

    1.2. Diner must input his Password.

        1.2.1. System must validate the Password with Database.

    1.3. Diners who have successfully registered for an account for the first time will be automatically logged in to his account.

    1.4. The application must not allow unregistered user to login and will alert the user.

    1.5. The application must not allow users who entered wrong passwords and will alert the user

    1.6. When user is logged in, the user will be redirected to the Home Page

4.3 View Profile

4.3.1 Description and Priority

The user profile management feature allows users to customize their profiles within the app. After accessing their profile settings, users can update personal information such as their name, preferred cuisines, and dietary restrictions. This feature supports the personalization of the app experience and is important for ensuring user data is up-to-date and reflective of individual preferences.

4.3.2 Stimulus/Response Sequences

Use Case ID	3.0		
Use Case Name	ViewProfile		
Created by	Le Yanzhi	Last Updated by	Le Yanzhi
Date Created	5/2/2024	Date Last Updated	27/3/2024

Actor	Diner
Description	Diner is allowed to view his profile after successfully login.
Preconditions	1. Diner must have successful login from Use Case Login (2.0)
Postconditions	2. System displays account information to Diner
Priority	Medium
Frequency of Use	Medium
Flow of Events	1. Diner selects option to view profile in Home Screen 2. System displays Username, Email, Cuisine Preferences, Rating, Operation Status, Proximity.
Alternative Flows	3.0.AC.1.0: If Diner selects the “Edit Profile” button in View Profile Page. 1. System directs Diner to Edit Profile Page using Use Case EditProfile (3.1) 2. System returns to Step 2.
Exceptions	NIL
Includes	3.1 ViewReferralCode
Special Requirements	NIL
Assumptions	NIL
Notes and Issues	NIL

4.3.3 Functional Requirements

1. Diners who have logged in must be able to view his profile.

    1.1. Profile must contain information and preferences about the Diner.

        1.1.1. Profile must include Username of Diner.

        1.1.2. Profile must include Email of Diner.

        1.1.3. Profile must include Food Preference of Diner.

        1.1.4. Profile must include Notification Setting of Diner.

        1.1.5. Profile must include the unique referral code of Diner if it has not been used yet,

4.4 Edit Profile

4.4.1 Description and Priority

The edit profile feature allows users to update their personal information within the app. After navigating to the "Edit Profile" section from the home page, users can modify details such as their username, email address, and password. Additionally, they can edit their preferences, including preferred cuisines and dietary restrictions.

4.4.2 Stimulus/Response Sequences

Use Case ID	4.0		
Use Case Name	EditProfile		
Created by	Le Yanzhi	Last Updated by	Le Yanzhi
Date Created	5/2/2024	Date Last Updated	27/3/2024

Actor	Diner
Description	Diner is allowed to edit his profile after successful login.
Preconditions	1. Diner must have successful login from Login Use Case
Postconditions	2. Diner's Account has been successfully updated in Database
Priority	Medium
Frequency of Use	Medium
Flow of Events	1. Diner selects option to edit profile details in View Profile Page 2. System directs Diner to Preferece Page 1, which includes CuisinePreferencesSetting Use Case (3.1.1) 3. System directs Diner to Preference Page 2, which includes RatingSetting Use Case (3.1.2), OperationStatusSetting Use Case (3.1.3), ProximitySetting Use Case (3.1.4). 4. System update account information. 5. System displays "Preferences Saved". 6. System directs Diner to Home Screen.
Alternative Flows	NIL
Exceptions	NIL
Includes	3.1.1 CuisinePreferencesSetting 3.1.2 RatingSetting 3.1.3 OperationStatusSetting 3.1.4 ProximitySetting
Special Requirements	NIL

Assumptions	NIL
Notes and Issues	NIL

4.4.3 Functional Requirements

1. Diners who have logged in must be able to edit his profile.

    1.1. Diner must be able to select his Food Preference

        1.1.1. Diner must select his food preferences after signing up for an account for the first time.

        1.1.2. Food Preferences must include the Cuisine Preferences.

            1.1.2.1. System must recommend restaurants that match the selected Cuisine Preferences.

        1.1.3. Food preferences must include Rating.

            1.1.3.1. System must recommend restaurants that have a higher Rating than the chosen Rating.

        1.1.4. Food preferences must include Operation Status.

            1.1.4.1. System must recommend restaurants that are currently operating only, or both operating and not operating, based on Diner's preferences.

        1.1.5. Food preferences must include Proximity.

            1.1.5.1. System must recommend restaurants that are within the chosen Proximity.

        1.1.6. Diner must be able to edit his Notification Setting.

            1.1.6.1. Diner must select his Notification Setting after signing up for an account for the first time.

            1.1.6.2. Diner must be able to choose whether he wishes to receive notifications of restaurant recommendations.

4.5 Retrieve Information

4.5.1 Description and Priority

Retrieving information and real-time database from the Google Maps API Database is required for the application to obtain the user's current location and location data of the restaurants. Thus, this use case is to call for API to obtain data of all restaurants in 250m proximity to the diner from the API database. This feature is of high priority.

This the parent use case for other use cases such as ViewRestaurants, SendNotifications and ViewReviews. All these use case requires a call to the Google Maps API to retrieve its real-time database to obtain location data of both the users and the restaurants, and also user reviews of the restaurant submitted into Google Reviews. The function requirements will thus include these use case as considerations.

4.5.2 Stimulus/Response Sequences

Use Case ID	5.0		
Use Case Name	RetrieveInformation		
Created by	Seah Kah Yen	Last Updated by	Le Yanzhi
Date Created	9/2/2024	Date Last Updated	27/3/2024

Actor	Google Maps API
Description	Call for API to obtain data of all restaurants in 1km proximity to the diner, such as locations of user and restaurant, and user reviews from the Google Maps database
Preconditions	1. Diners have successfully logged onto the application.
Postconditions	1. Diners will be able to see the restaurants in 1km proximity to them based from the Google Maps API database. 2. Diners can choose to click and view any restaurants in details.
Priority	High
Frequency of Use	High
Flow of Events	1. System directs Diner to Home Screen with the Google Map Geolocation. 2. System prompts Diner to grant permission to application to access location. 3. Diner grants application to access location. 4. Diner sees all the restaurants in 1km proximity to them on the Google Map interface.
Alternative Flows	5.0.AC.1.0: If Diner clicks on a resturant icon on the map. 1. System directs Diner to Use Case ViewRestaurants (5.1) 2. System returns to Step 4.

	5.0.AC.2.0: If Diner clicks on the dining history button 1. System directs Diner to Use Case DiningHistory (6.0) 2. System returns to Step 4. 5.0.AC.3.0: If Diner click on his own profile. 1. System directs Diner to Use Case ViewProfile (3.0) 2. System returns to Step 4.5 .0.AC.4.0: If Diner click on weather button. System directs Diner to Use Case WeatherPage(9.0) System returns to Step 4. 5.0.AC.5.0: If Diner click on friends button. System directs Diner to Use Case Social (7.0) System returns to Step 4. 5.0.AC.6.0: If Diner click on user feedback button. System directs Diner to Use Case UserFeedBack8.0) System returns to Step 4. 1. 5.0.AC.7.0: If Diner choose to not grant permission to application to access location. 2. System displays “Permission to access location was denied”.
Exceptions	NIL
Includes	NIL
Special Requirements	NIL
Assumptions	NIL
Notes and Issues	NIL

4.5.3 Functional Requirements

1. Diners will be able to see an interface that incorporates Google Maps geolocation features.
    1.1. The user will be able to see themselves in relation to nearby restaurants within and outside of proximity.
2. The user will be able to click and view any restaurant in details that is in 1000 metres proximity.
    2.1. The user can see what type of cuisine that the restaurant provides.
    2.2. The user can see the average ratings of that restaurant based on the Google API database.
    2.3. The user can see the distance to that restaurant based on where they are currently located based on the Google API database
    2.4. The user can choose to “Jiak!” at the specific restaurant they choose to view in details.
3. The user will be able to click and view the recent user reviews of the restaurant the user chose.
    3.1. The user reviews will contain name of the user
    3.2. The user reviews will contain the star ratings out of 5 made by the user.
    3.3. The user reviews will contain the written review typed out by the user.
    3.4. The user reviews will contain the date of the review made by the user.
4. Diners can indicate in the application, by pressing a button labelled “Jiak!”, that they have chosen to eat at a restaurant.
    4.1. The option to indicate dining at a recommended restaurant is alerted by a notification.
5. Upon indicating in the application that the diners have chosen to dine in the restaurant by pressing a button labelled “Jiak!”, diners will be redirected to the Jiak!AtRestaurant page.
    5.1. Diners must press the “Use Referral Code” button to use a referral code.
        5.1.1. Diners will be redirected to the UseReferralCode page.
        5.1.2. Diners must key in a referral code; it will show a confirmation popout.

6. Upon indicating in the application that the diners have finished dining by pressing the "Review Restaurant" button, diners will be redirected to the ReviewRestaurant page.

    6.1. On the ReviewRestaurant page, diners must input their written review and rate the restaurant out of 5 stars.

    6.2. On the ReviewRestaurant page, diners must input their reviews

        6.2.1. Diners must rate the restaurant out of 5 stars.

        6.2.2. Diners must input their own written review.

    6.3. Diners will receive notifications of the recommended restaurants in proximity to the diner, diner can already indicate the distance of proximity to be notified.

        6.3.1. Notifications sent to users will include details of the restaurant.

            6.3.1.1. Notification will include the type of cuisine.

            6.3.1.2. Notification will include the ratings of the restaurant.

            6.3.1.3. Notification will include the distance between you and the restaurants.

4.6 Dining History

4.6.1 Description and Priority

Dining history allows diners to view a list of restaurants that they have dined at before. From this page, shall be able to edit or delete their reviews. This feature is of medium priority.

4.6.2 Stimulus/Response Sequences

Use Case ID	6.0		
Use Case Name	DiningHistory		
Created by	Lim Xin Wei	Last Updated by	Lim Xin Wei
Date Created	8/2/2024	Date Last Updated	8/2/2024

Actor	Diner
Description	Diner is able to view a record of the restaurants that they have dined at in the past.
Preconditions	1. Diner must have successful login from Login Use Case.
Postconditions	1. System displays a chronological list of restaurants that the diner has eaten at in the past.
Priority	Medium
Frequency of Use	Medium
Flow of Events	1. Diner clicks on the dining history button. 2. Diner shall be able to click on their review of a restaurant to see what they have reviewed previously.
Alternative Flows	6.0.AC.1.0 Diner shall able to edit their review of a restaurant they have dined at before. (EditReview 6.1) 6.0.AC.2.0 Diner shall able to delete their review of a restaurant they have dined at before. (DeleteReview 6.2)
Exceptions	NIL
Includes	6.1 EditReview 6.2 DeleteReview
Special Requirements	NIL
Assumptions	NIL
Notes and Issues	EditReview 6.1 and DeleteReview 6.2 can only be done if there already exist at least one review of a restaurant.

4.6.3 Functional Requirements

1. Diners who have logged in must be able to view his dining history.

    1.1. Dining history must contain the reviews that the diner had given to each restaurant.

        1.1.1. Dining history must include name of restaurant.

        1.1.2. Dining history must include star review of the restaurant. OBJ

        1.1.3. Dining history must include written review of the restaurant.

        1.1.4. Dining history must include date of dining at restaurant.

    1.2. Dining history for each review must have an edit button to allow editing of review.

        1.2.1. Edit review must be to the same restaurant.

        1.2.2. Edit review must get the new star rating.

        1.2.3. Edit review must get the new written rating.

        1.2.4. Edit review must update the date of the review.

        1.2.5. Edit review must show a successful message upon successful editing review.

    1.3. Dining history for each review must have a delete button to allow deletion of the review.

        1.3.1. Delete review must ask for a double confirmation of the deletion of the review. OBJ

        1.3.2. Delete review must show a successful message upon successful deletion of review.

4.7 Social

4.7.1 Description and Priority

Social allows diners to add and view friends, host or join parties, split bills between members of the party, as well as redeem points. Diners, as well as the firestore database are required to interact with the system to accomplish this. This feature is of high priority since it is our second main feature.

4.7.2 Stimulus/Response Sequences

Use Case ID	7.0		
Use Case Name	Social		
Created by	Seah Kah Yen	Last Updated by	Seah Kah Yen
Date Created	8/4/2024	Date Last Updated	8/4/2024

Actor	Diner
Description	Diner is able to redeem points, and choose to view and add friends, split bills and host or join parties
Preconditions	1. Diner must have successful login from Login Use Case.
Postconditions	2. System allows diners to redeem points, and choose to view and add friends, split bills and host or join parties
Priority	High
Frequency of Use	Medium
Flow of Events	1. Diner clicks on the Friends button in the homepage UI. 2. Diner should be able redeem point, choose to view and add friends, split bills and host or join parties
Alternative Flows	NIL
Exceptions	NIL
Includes	7.1 RedeemPoints 7.2 SplitCost 7.3 Friends 7.4 Parties
Special Requirements	NIL
Assumptions	NIL

Notes and Issues	NIL

4.7.3 Functional Requirements

1. Diners who have logged in must be able to view the social page.
    1.1. Social page must include the number of points a diner has.
    1.2. Social page must include an input to type in the points .
        1.2.1. Social page must include an option to redeem the points in the input.
        1.2.2. Diners must only be able to redeem integer numbers of points.
        1.2.3. System must alert Diners when unsuccessful redemption occurs.
    1.3. Social page must include a button to view or add friend.
        1.3.1. Clicking the button will redirect them to the Friends Page.
            1.3.1.1. Diners must be able to input Friend code.
                1.3.1.1.1. Diners must be able to add Friend after inputting correct code.
            1.3.1.2. Diners must be able to view all the friends they have added.
            1.3.1.3. Diners must be able to add friends to party.
            1.3.1.4. Diners must be able to remove friends from party.
            1.3.1.5. Diners must have an option to correctly configure their editing of party members before confirming the party settings.
    1.4. Social page must include a button to split cost in the party.
        1.4.1. Clicking the button will redirect them to the Party Page to split cost.
            1.4.1.1. Party Page must display all diners in the party.
                1.4.1.1.1. Party Page must clearly display the host of the party.
                1.4.1.1.2. Party Page must clearly display the guests of the party.
            1.4.1.2. Party Page must allow diners to input the total cost of their bill.
            1.4.1.3. Party Page must allow diners to split the cost of the bill through the application.
                1.4.1.3.1. Splitting the cost must give the correct divided amount for each member of the party.
    1.5. Social page must include a button for diners to view their joined party list as a guest.

1.5.1. Clicking the button will redirect them to the Parties Page.

1.5.1.1. Diners must be able to see the parties they have joined as a guest.

1.5.1.1.1. Diners who are host of a party should not be able to view any joined parties.

1.5.1.2. Diners, who are guests of a party, must be able to leave the party.

1.5.1.2.1. Diners who have left a party should be removed from the party.

1.5.1.2.2. Diners should be able to join back the same party, if invited by the host.

4.8 UserFeedBack

4.8.1 Description and Priority

User Feedback. This feature allows any users to input their feedback and their opinion on anything, such as any improvements to be done on our application. This feature is of low priority.

4.8.2 Stimulus/Response Sequences

Use Case ID	8.0		
Use Case Name	UserFeedBack		
Created by	Seah Kah Yen	Last Updated by	Seah Kah Yen
Date Created	8/4/2024	Date Last Updated	8/4/2024

Actor	Diner
Description	Diner is able to input any opinion or feedback they have about our application
Preconditions	1. Diner must have successful login from Login Use Case.
Postconditions	1. Diner has successfully submitted their user feedback and it will be stored in the firebase such that we, the operators of the application can view.
Priority	Low
Frequency of Use	Low
Flow of Events	1. Diner clicks on the UserFeedBack button in the homepage UI. 2. Diner would be able to type in their feedback in the input, and submit it to be stored in the firebase.
Alternative Flows	8.1.AC.1.0: If user leaves input field blank 1. System displays: “Error. Please enter your feedback before submitting.” 2. Goes back to step 2
Exceptions	NIL
Includes	NIL
Special Requirements	NIL
Assumptions	NIL

Notes and Issues	NIL

4.8.3 Functional Requirements

1. Diners who have logged in must be able to view the user feedback page.

    1.1. Diner should be able to type and input in their feedback or opinions.

        1.1.1. Diner should be able to press submit to store their feedback in the firebase.

4.9 Weather Page

4.9.1 Description and Priority

Weather Page allows diners to view the current weather. This feature is of low priority.

4.9.2 Stimulus/Response Sequences

Use Case ID	9.0		
Use Case Name	WeatherPage		
Created by	Seah Kah Yen	Last Updated by	Seah Kah Yen
Date Created	8/4/2024	Date Last Updated	8/4/2024

Actor	Diner
Description	Diner is able to view the weather page
Preconditions	1. Diner must have successful login from Login Use Case.
Postconditions	2. Diner would know the current weather in their area.
Priority	Low
Frequency of Use	Low
Flow of Events	1. Diner clicks on the weather button in the homepage UI. 2. Diner would know the current weather
Alternative Flows	NIL
Exceptions	NIL
Includes	NIL
Special Requirements	NIL
Assumptions	NIL
Notes and Issues	NIL

4.9.3 Functional Requirements

1. Diners who have logged in must be able to view the weather page upon clicking the weather icon.

    1.1. Weather page should correctly display the temperature of current weather.

    1.2. Weather page should correctly display the current state of weather.

5. Other Nonfunctional Requirements

5.1. Performance Requirements

5.1.1. Jiak! application should load within 3 seconds after launching.
5.1.2. After a successful launch, a welcome page for users to log in or create account should be loaded within 3 seconds.
  5.1.2.1. Users can choose to create a new account to log in.
  5.1.2.2. Users can choose to log in as Diner with their created account.
5.1.3. After a successful account creation from users, the Cuisine Preference Page should be loaded within 3 seconds.
  5.1.3.1. User should be able to successfully submit and filter their food preference.
5.1.4. After successful cuisine selection, the Dining Experience Page should be loaded within 3 seconds.
  5.1.4.1. User should be able to successfully submit and their dining experiences.
5.1.5. The interface incorporating Google Maps geolocation features should be loaded within 10 seconds, displaying the nearby restaurants and the user's location.
5.1.6. Notifications of in-proximity recommended restaurants should be sent to users within 10 seconds.
5.1.7. Users should be able to successfully submit all their reviews, of any types, within 10 seconds.

5.2. Safety Requirements

5.2.1. The Jiak! application must not encourage or promote reckless behavior, such as distracted dining or unsafe navigation while using the app.
5.2.2. All features and functionalities of the Jiak! application must comply with relevant safety regulations and standards to ensure user safety.
5.2.3. The application should include safety reminders or notifications to discourage unsafe actions, such as dining in hazardous locations or using the app while driving.

5.3. Security Requirements

5.3.1. If the system detects an incorrect input of the user’s password 3 times, the account will be locked and disabled till further verification. (yet to be implemented).
  5.3.1.1. When an account is locked, the system will send a verification email to the account’s registered email. (yet to be implemented).
5.3.2. User information such as personal email and location data should not be leaked.
  5.3.2.1. User information must not be accessible or used by unauthorised parties.
  5.3.2.2. User information shall be permanently deleted upon account deletion.
5.3.3. The application should aim to obtain relevant security certifications, build trust with users and potential partners by ensuring that the application meets rigorous security requirements.
  5.3.3.1. Relevant security certifications include ISO 27001
  5.3.3.2. Relevant security certifications include SCO 2

5.4. Software Quality Attributes

5.4.1. Jiak! will provide the accurate distance with margin of error up to 25m on the proximity of the recommended restaurant.
5.4.2. Jiak! will provide the specific and fully accurate ratings on the ratings of the recommended restaurant.

5.4.3. Jiak! will provide the specific and fully accurate operation status of the recommended restaurant.
5.4.4. Jiak! will provide the specific and fully accurate cuisines of the recommended restaurant.
5.4.5. Notifications will be sent reliably to users who have enabled the feature, with a notification success rate of at least 95%.
5.4.6. Reviews from user would be 100% from the user, with no external edits.

5.5. Software Maintainability

5.5.1. Jiak! application should have a customer support function where users can chat with customer support for any issues they encounter.
5.5.2. Jiak! should have a small pop-up feedback screen for users every 10 times they log into application,
5.5.3. If the real-time API of Google Maps geolocation is unresponsive or unavailable within 10 seconds, the application must not crash while loading.
5.5.4. If the application cannot connect to the APIs or get location data within 10 seconds, it will display a dialog window to inform the user and prompt for a reconnection attempt.
5.5.5. If the user is unable to submit a review, the application should provide an error message to ask the user to resubmit again within 10 seconds.

5.6. Software Scalability

5.6.1. The backend database should be designed to handle large increase in reviews from users.
5.6.2. The system will be designed to integrate new features such as a community page.

5.7. Software Usability

5.7.1. At least 80% of users, including first time users, should be able to discover a highly rated dining experience within 2 minutes of entering the Jiak! Application.
5.7.2. The restaurants appearing near proximity of the user on the screen should be easy to see and locate.
5.7.3. Notifications must be formatted in such a way it presents clear and concise information of the recommended restaurant.
5.7.4. The review interface shall allow diners to easily rate and provide optional written reviews to encourage user engagement and feedback.

5.8. Business Rules

5.8.1. All features of the Jiak! application should be loaded and function within 10 seconds. This includes the welcome page, account creation, food preference interface, Google Maps geolocation features and review submission.
5.8.2. Accuracy and Reliability:
    5.8.2.1. Jiak! should provide accurate and reliable recommendations, including distance, ratings, operation status and cuisines of recommended restaurants.
5.8.3. Notifications should be sent to users who have enabled the feature with a notification success rate of at least 95%.
5.8.4. Reviews provided by users should be 100% from the user with no external edits.
5.8.5. At least 80% of users, including first time users, should be able to discover a highly rated dining experience within 2 minutes of entering the Jiak! Application.

5.8.6. The restaurants appearing near proximity of the user on the screen should be easy to see and locate.

6. Other Requirements

1. Database Requirements:
    1.1. Jiak! database should be designed to handle queries related to user authentication, restaurant recommendations, and review management.

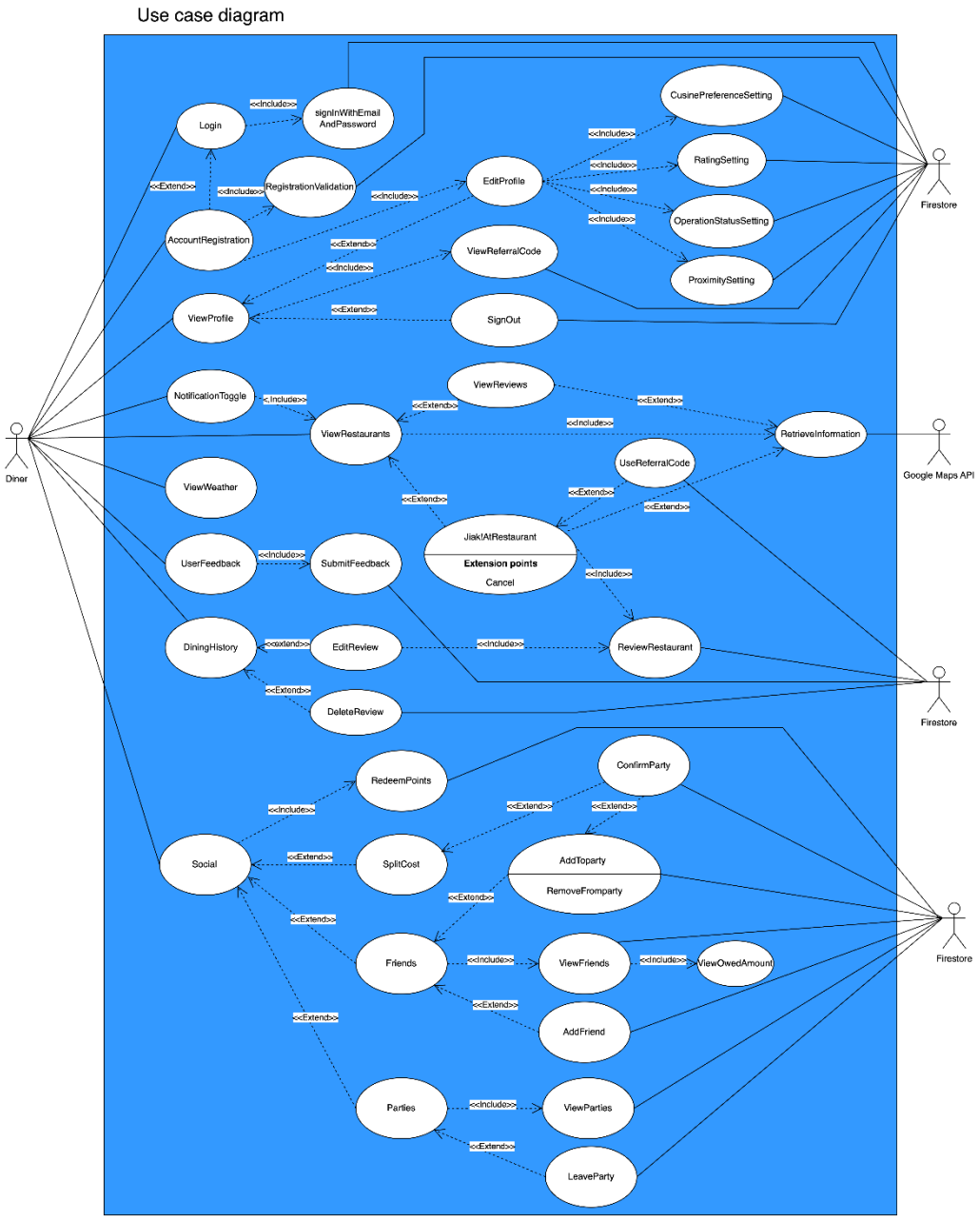
Appendix A: Data Dictionary

Term	Definition
Account	The profile of the Diner, which contains the Diner's login details, such as their Username, Email, and Password, as well as any other information associated with the Diner, such as Cuisine Preferences, Rating given, and Restaurant History.
Username	A (non-unique) display name chosen by the Diner.
Email	A unique email address associated with the Diner, to be used for log in.
Password	A string of characters that contains letters, numbers and symbols that allow Diners to login and access their account.
Diner	A person who owns the account and uses the function of the application.
Cuisine Preferences	The list of cuisines a Diner prefers, to be used as a guideline for restaurant recommendations of notification by the system. Includes different types of cuisine such as Japanese, Korean, Chinese.
Restaurant History	The list of restaurants a Diner has patronised.
Google Maps API	Application Programming Interface from Google to fetch and display data, specifically live location, and restaurants information.
Rating	Rating given by a Diner who has visited the restaurant. Rating ranges from 0 to 5 and takes on integer values. Rating is stored in Database.
Operation Status	Shows whether a restaurant is open or closed.
Proximity	The distance where restaurant will be recommended to the Diner by Notification.
Notification	In-App notification pop-up sent to the Diner, to recommend nearby restaurant that aligns with his Cuisine Preferences, Rating, Operation Status, and Proximity.
Review	Review includes Rating and a short description of user experience.
Database	The system's database (firebase) is used to store Account and any related data.
Eater Code	A unique code given to Diners upon registration. This can be used as a referral code (single use) by other diners to enjoy discount, or to add friends by entering the Eater Code of Diners that one wish to add friends.

Friend	The friends of a Diner. A Diner can add friends by entering their Eater Code. Diners can add their friends to his party. Diners can also be added to the party of his friends.
Party	A group of friends who dine together. After dining, the party may Split Cost among its members.
Split Cost	A feature where after a Party dines, the host of the Party can know how much each person will have to pay.

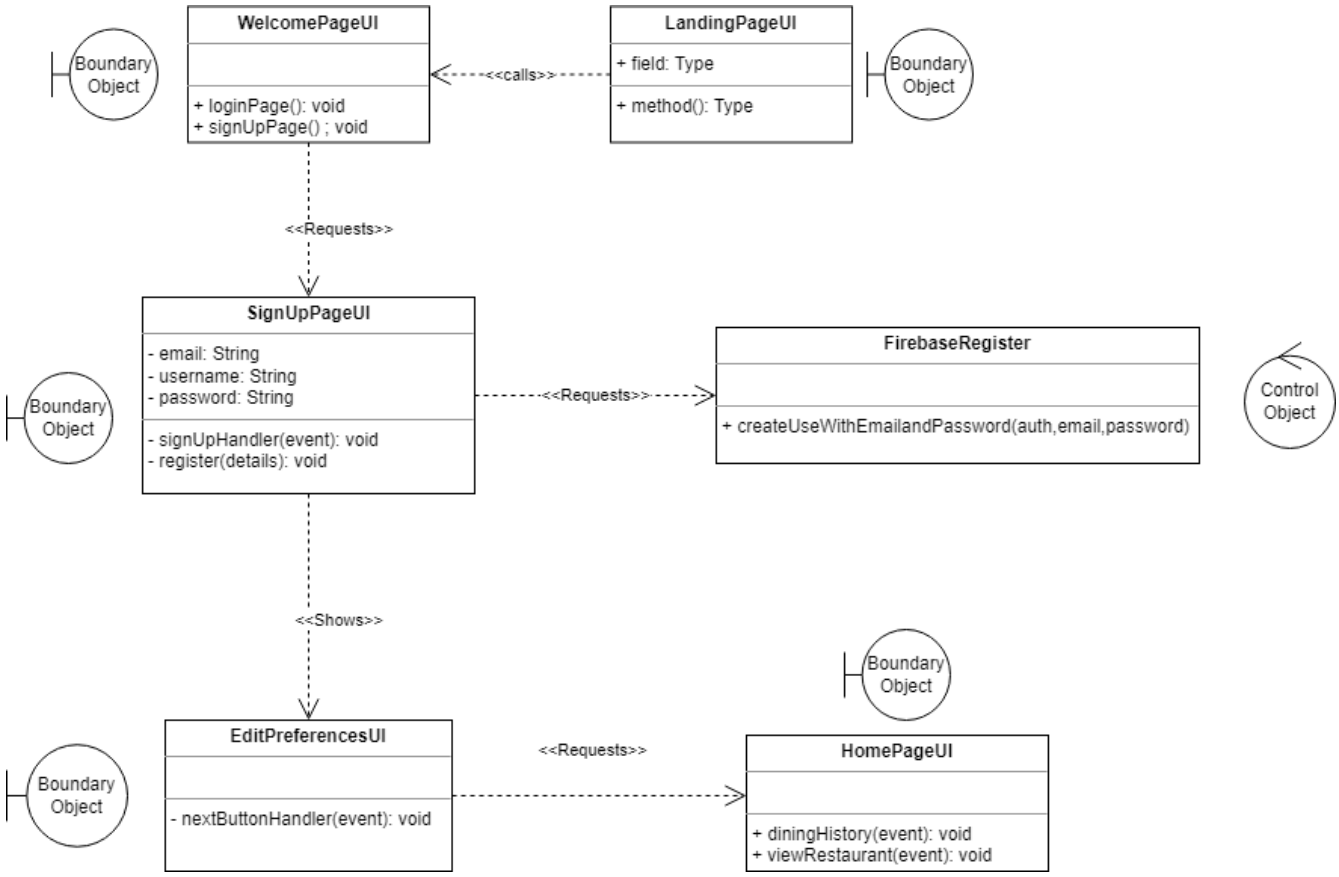
Appendix B: Analysis Models

1. Use Case Diagram

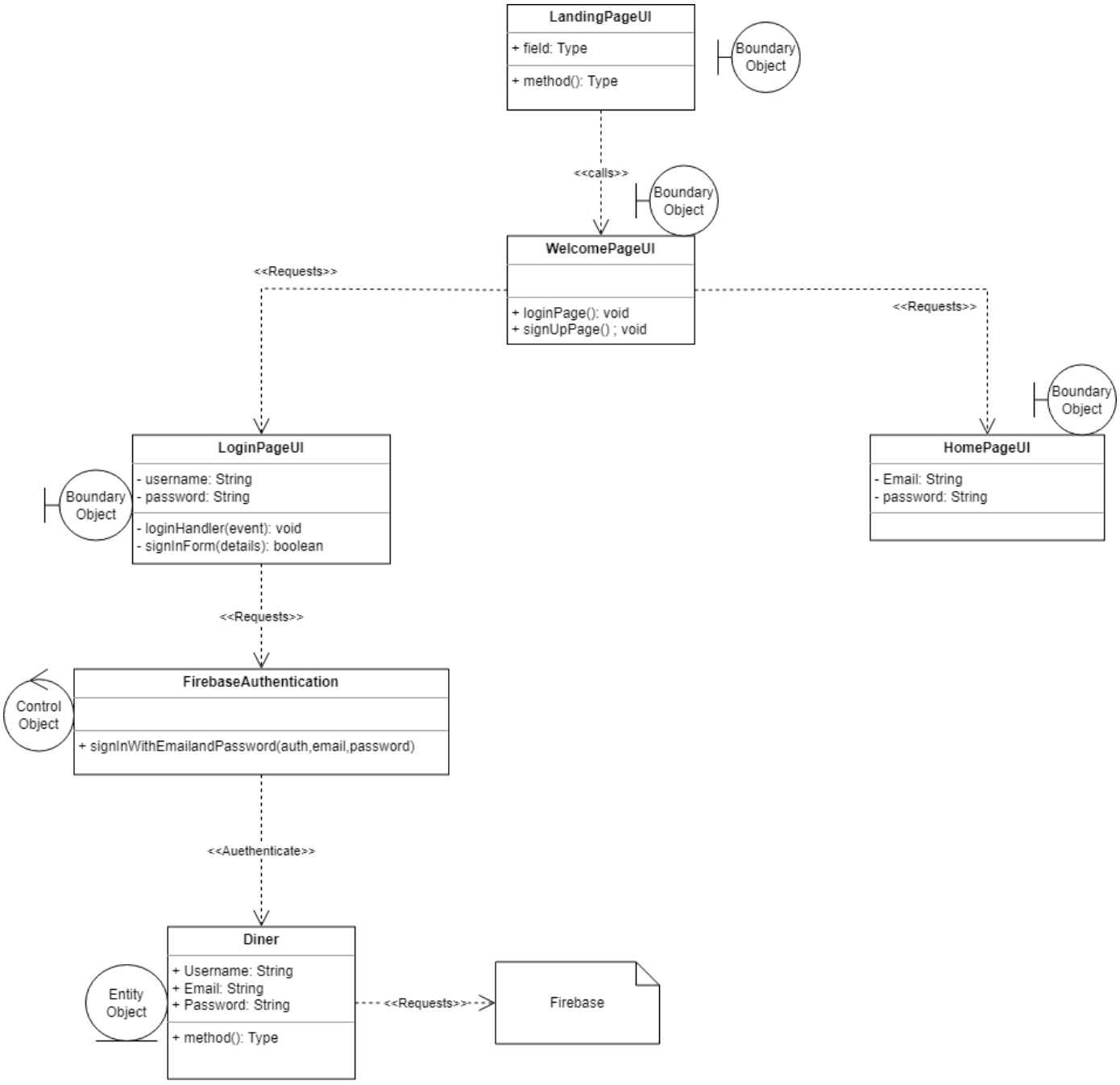


2. Class Diagram

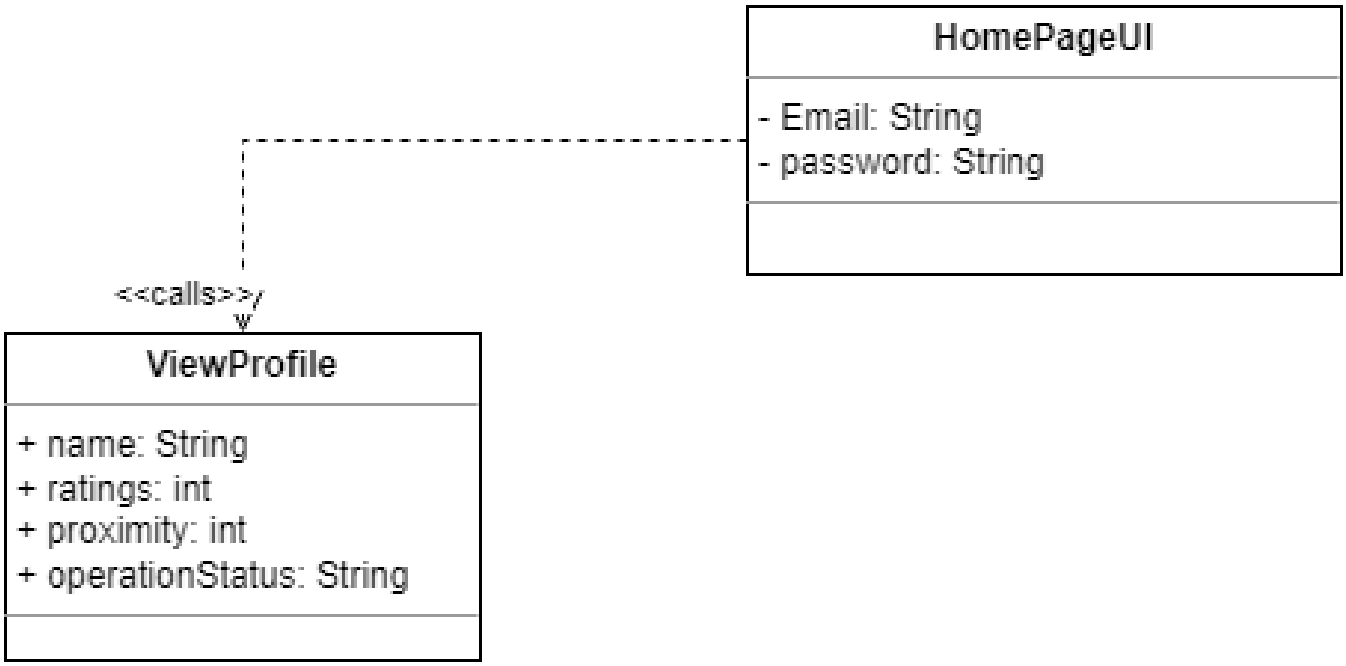
2.1 AccountRegistration



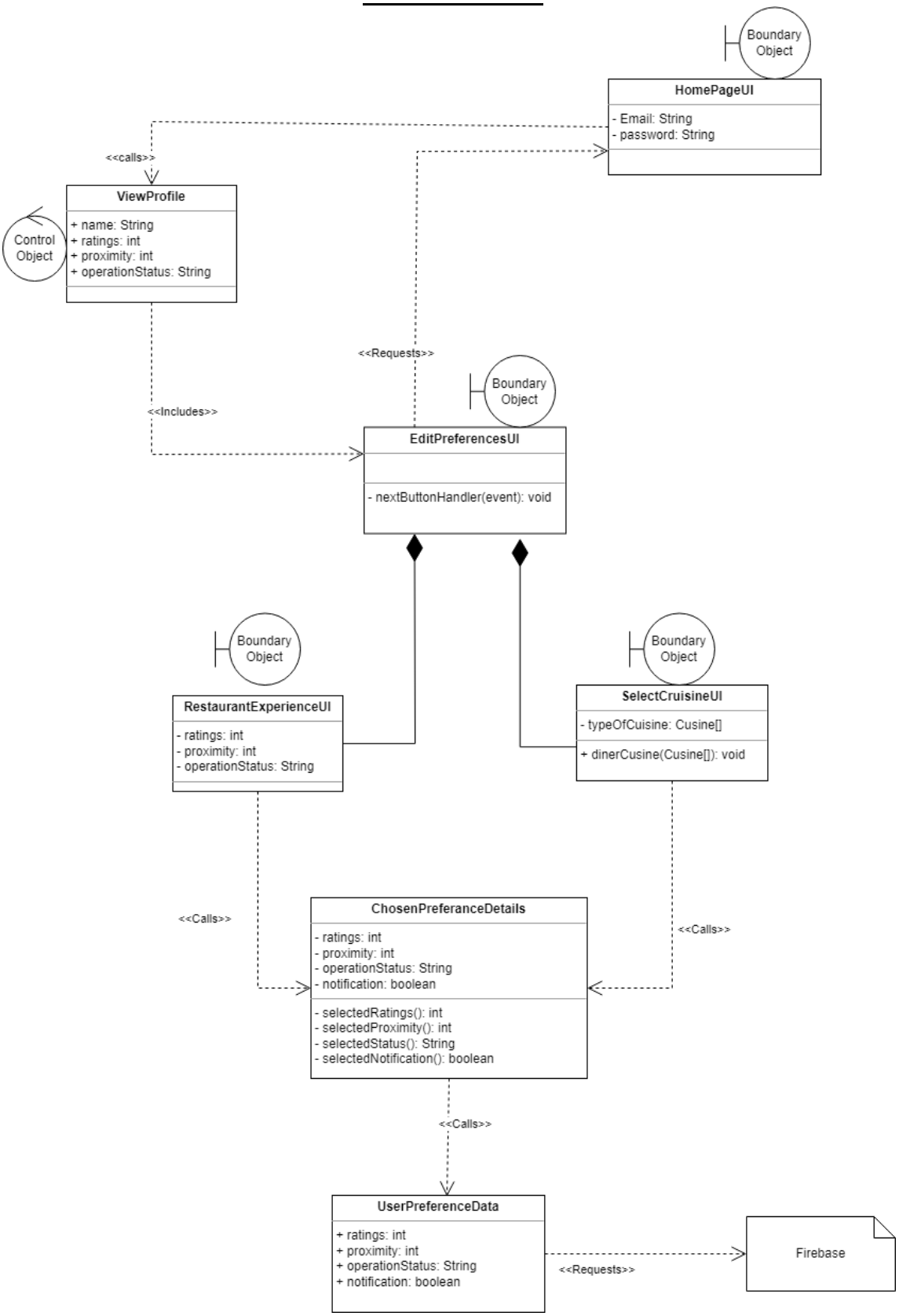
2.2 Login


LandingPageUI
+ field: Type
+ method(): Type
Boundary Object
<<calls>>
Boundary Object
WelcomePageUI
+ loginPage(): void
+ signUpPage() ; void
<<Requests>>
<<Requests>>
Boundary Object
HomePageUI
- Email: String
- password: String
LoginPageUI
- username: String
- password: String
- loginHandler(event): void
- signInForm(details): boolean
Boundary Object
<<Requests>>
FirebaseAuthentication
+ signInWithEmailandPassword(auth,email,password)
Control Object
<<Auethenticate>>
Diner
+ Username: String
+ Email: String
+ Password: String
+ method(): Type
Entity Object
<<Requests>>
Firebase

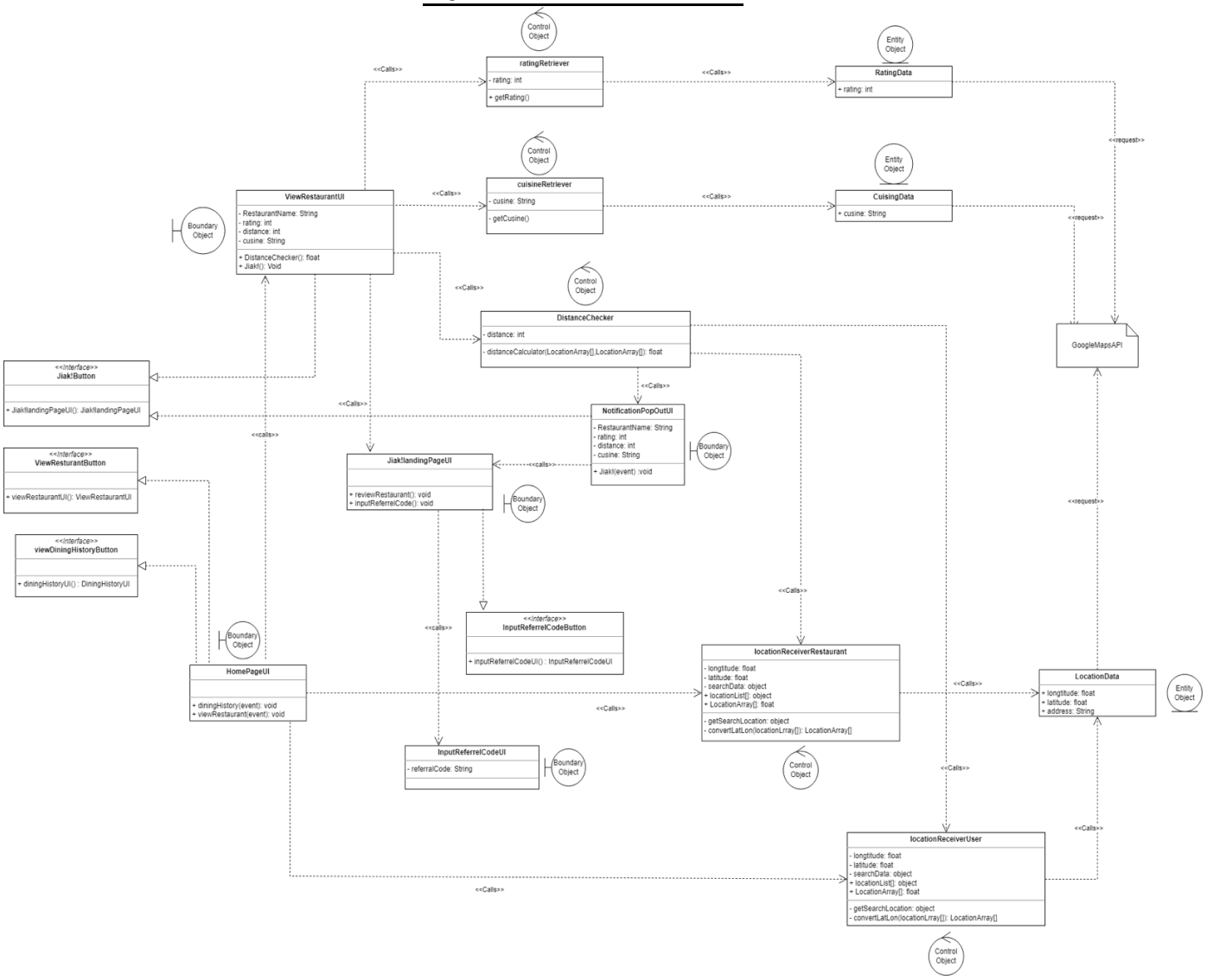
2.3 View Profile



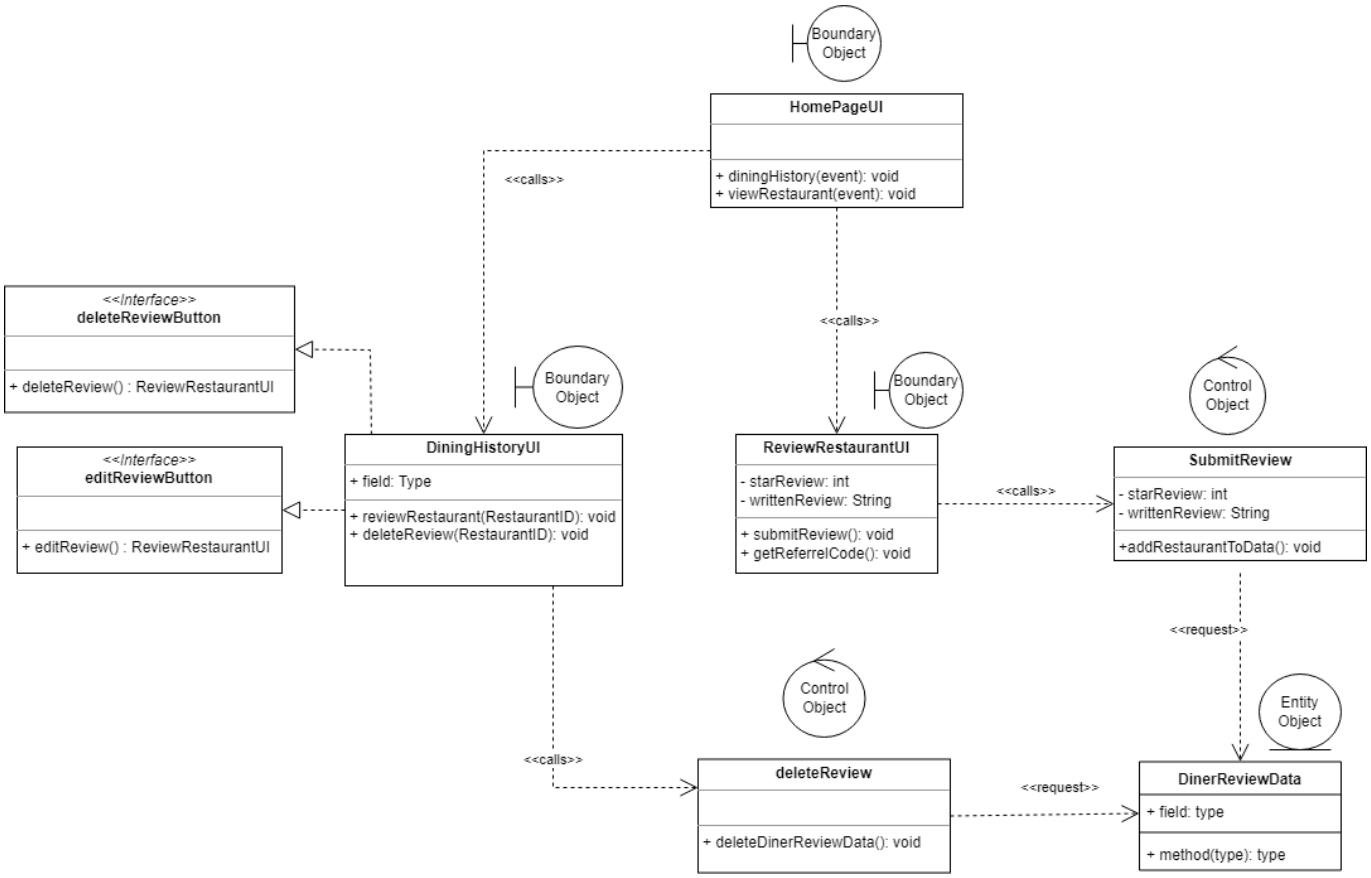
2.4 Edit Profile



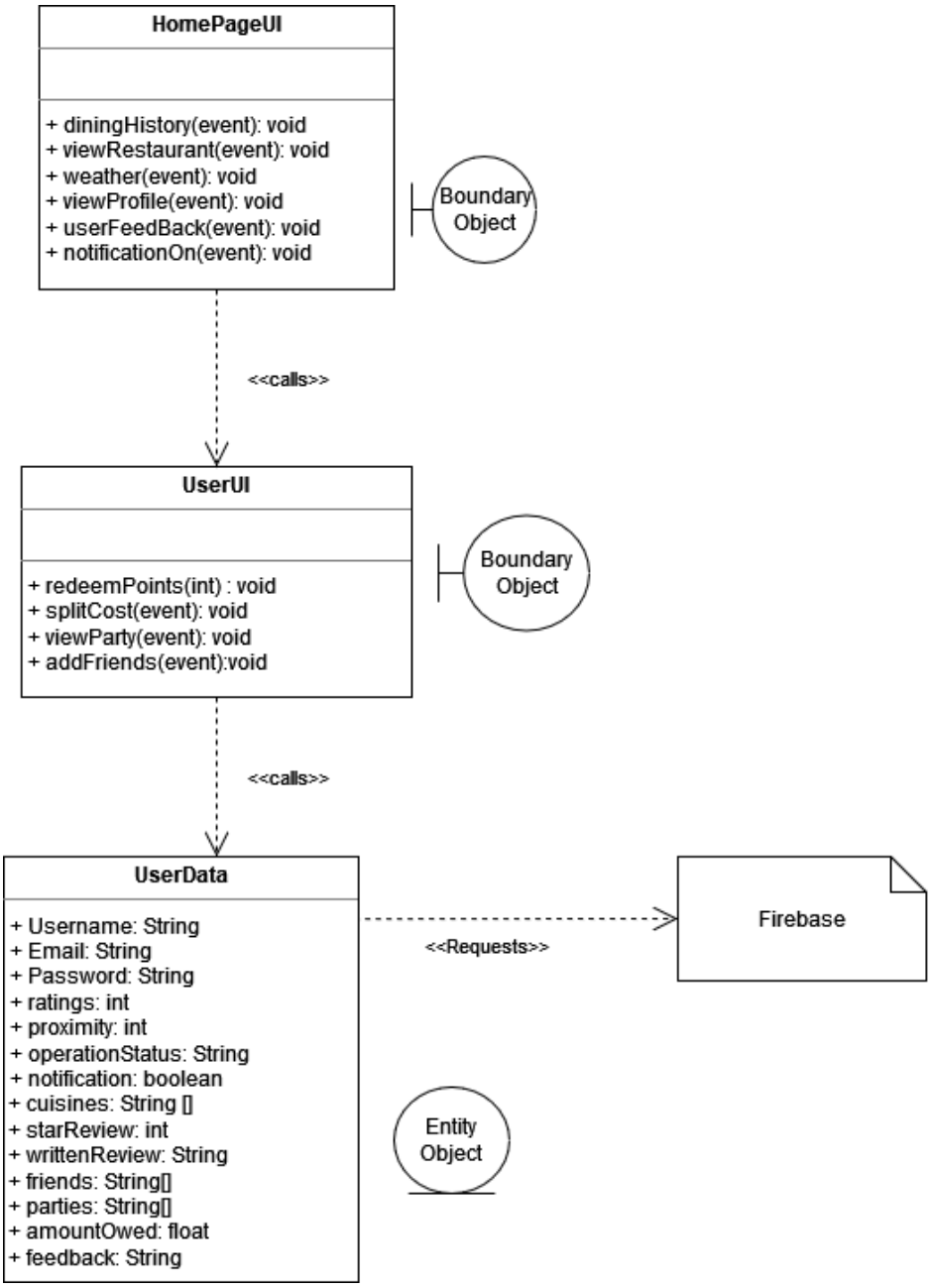
2.5 Retrieve Information



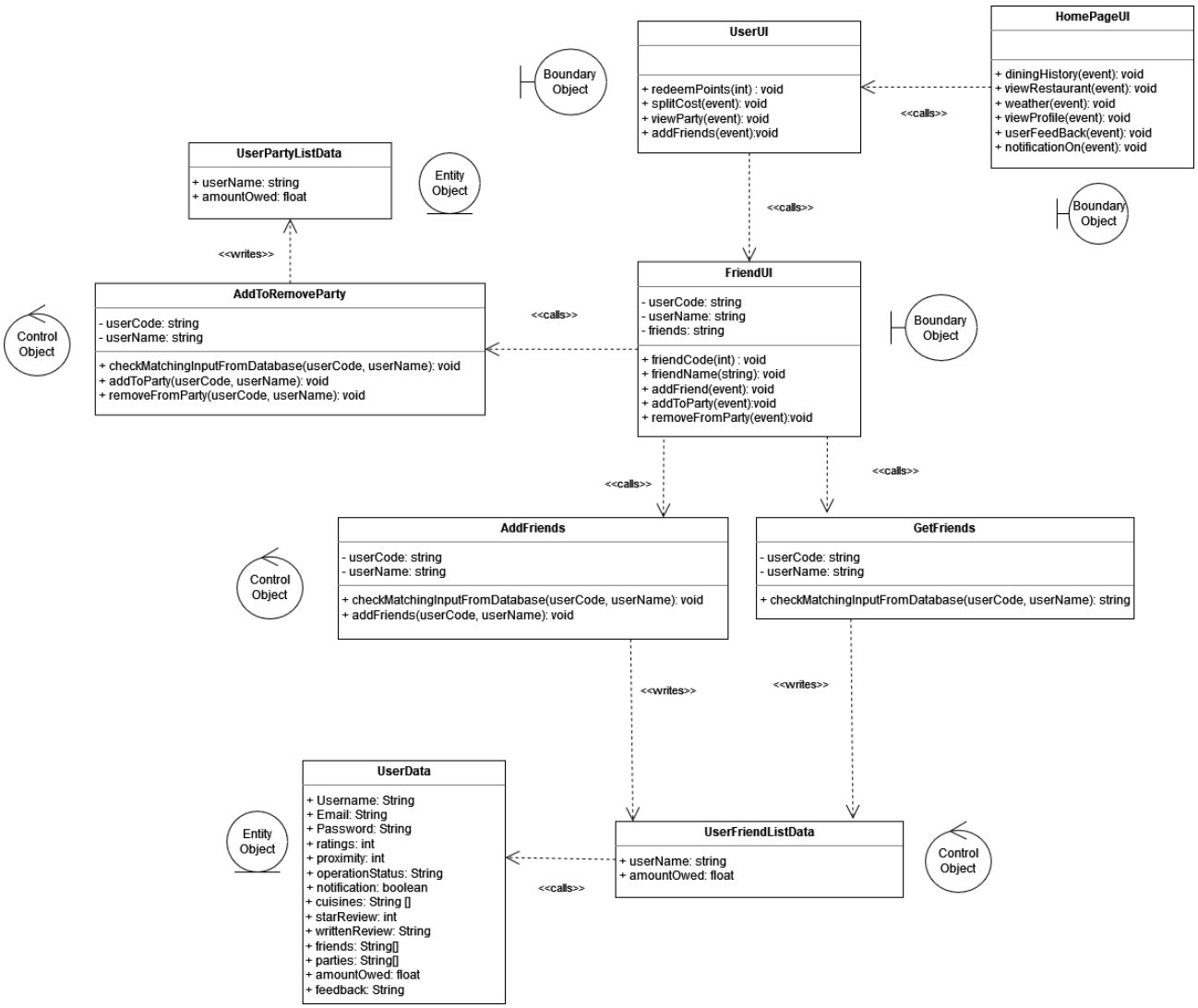
2.6 Dining History



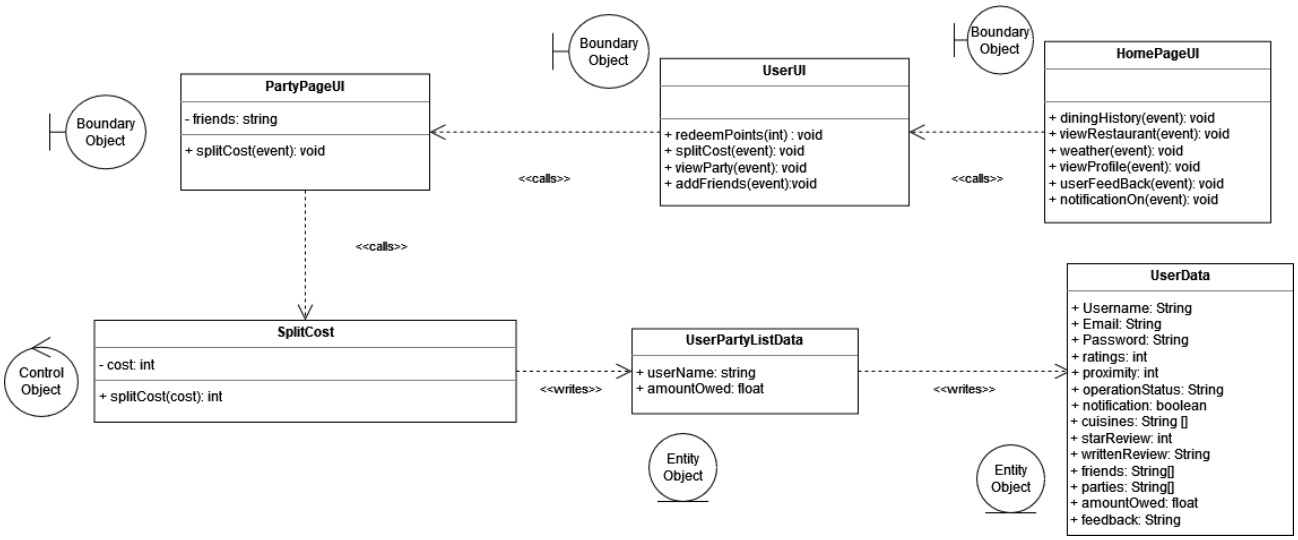
2.7 Social (Users)



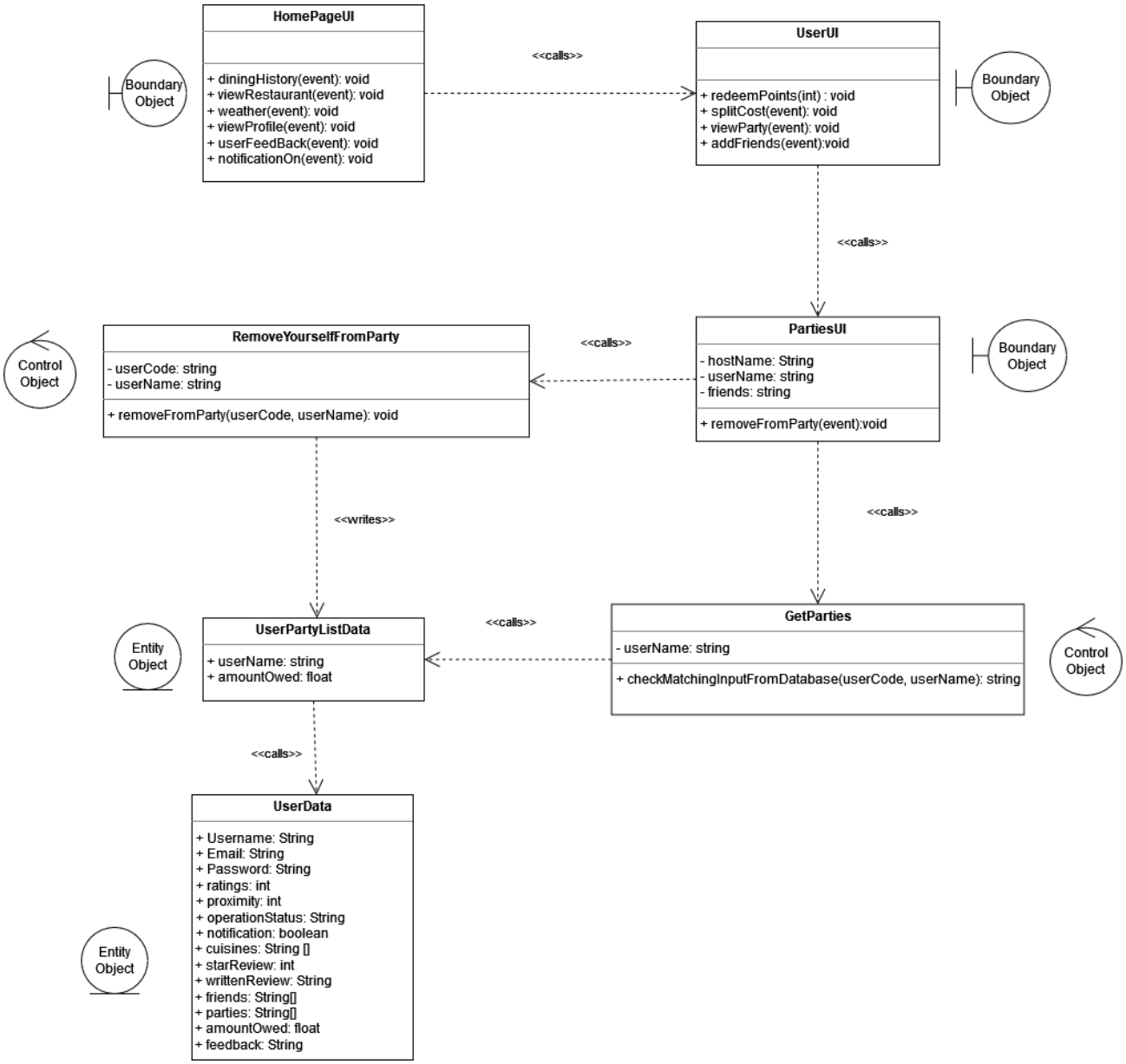
2.7 Social (Friends)



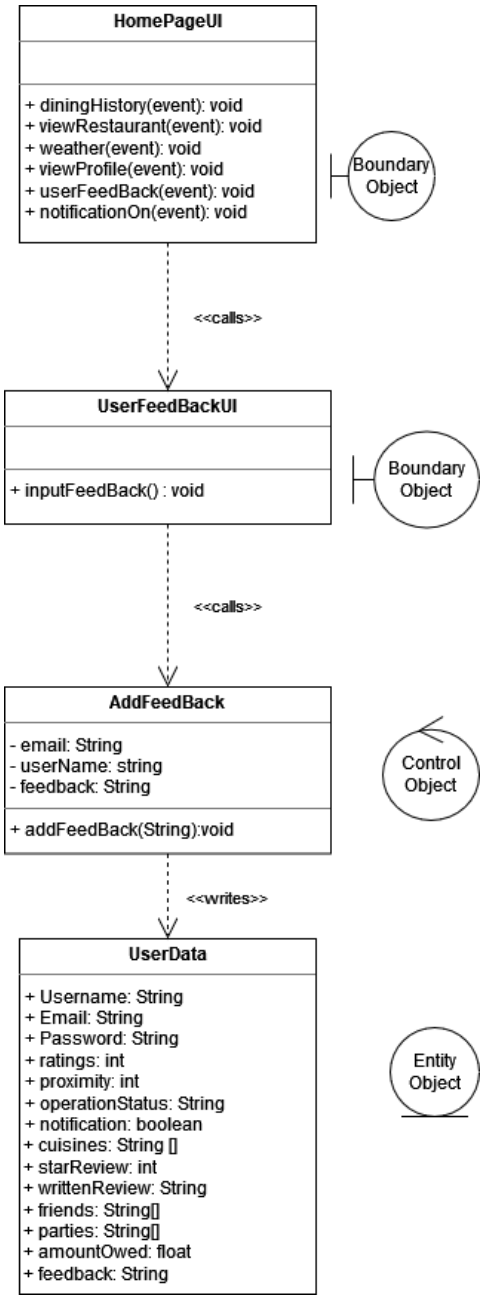
2.7 Social (SplitCost)



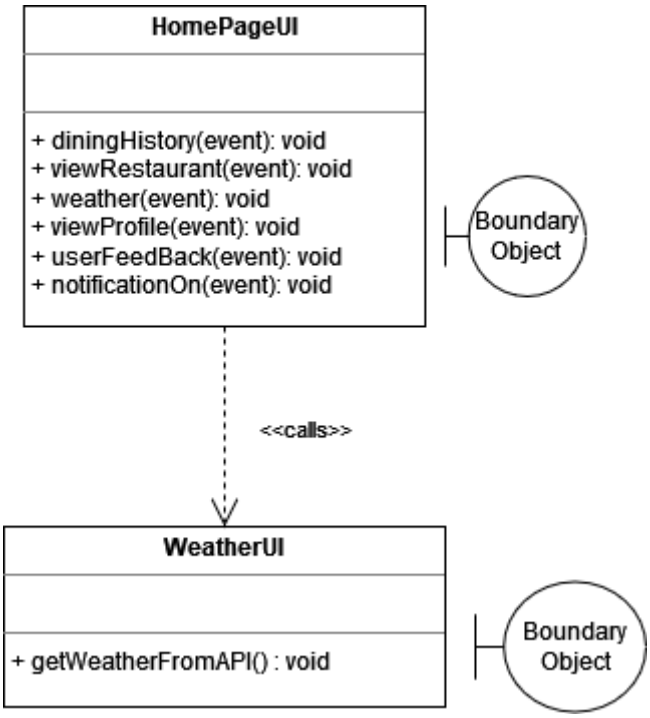
2.7 Social (Parties)



2.8 UserFeedback

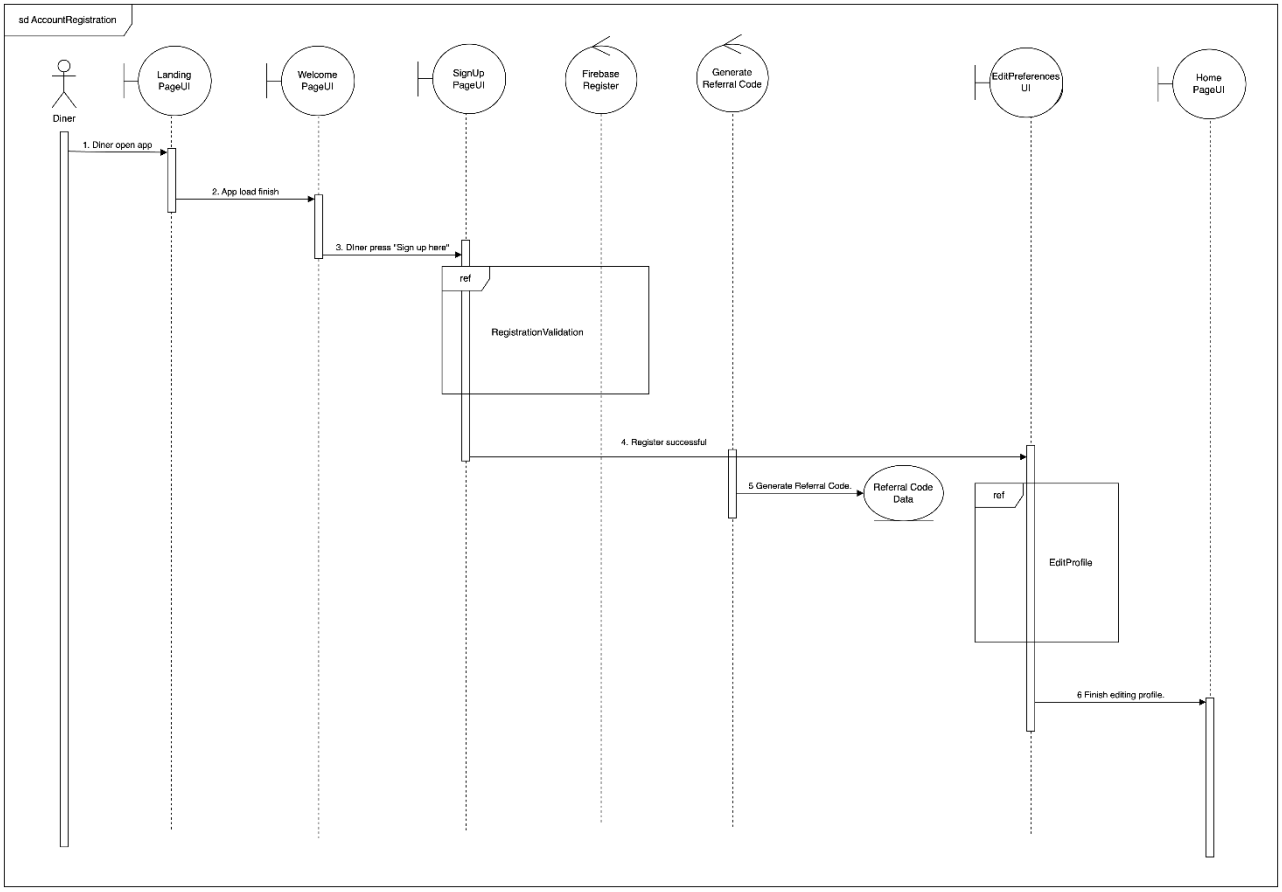


2.9 WeatherPage

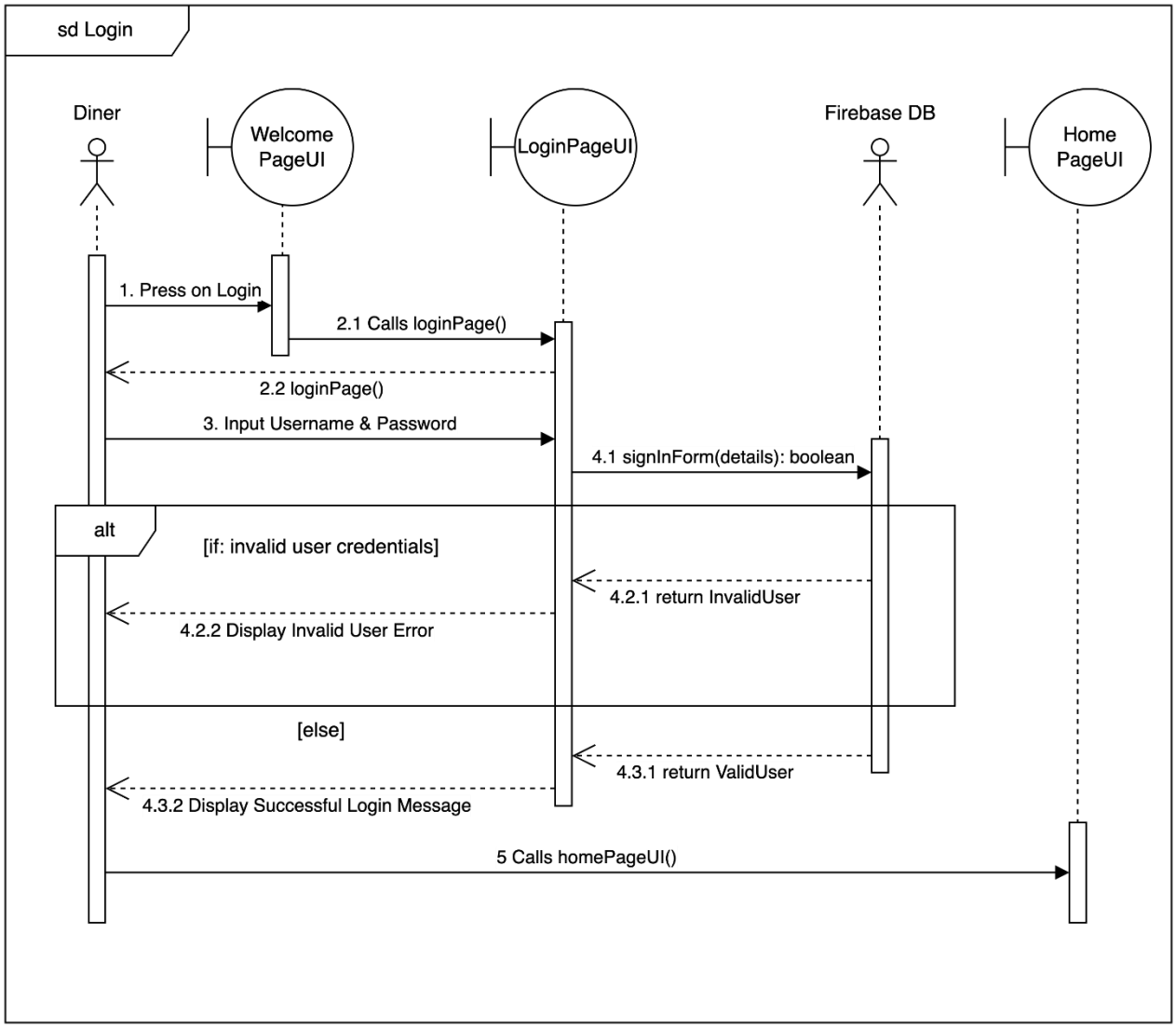


3. Sequence Diagram

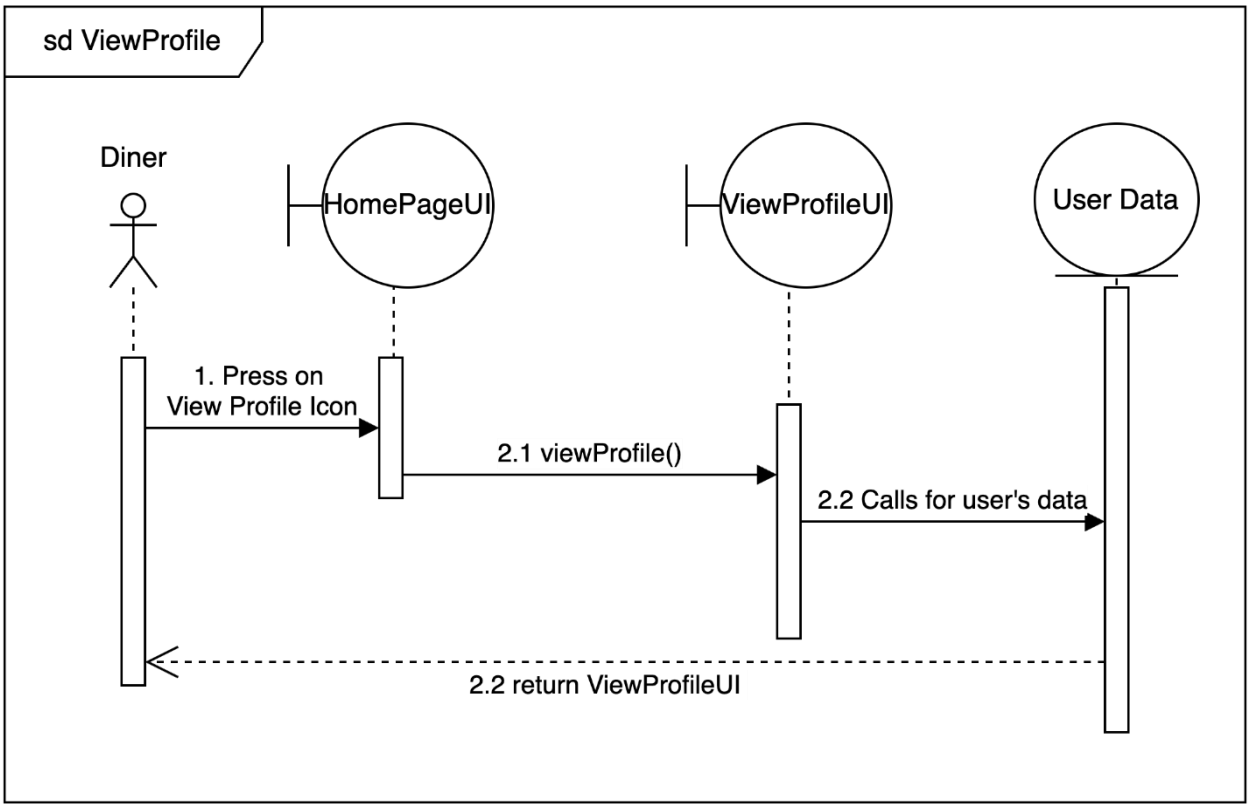
3.1 Account Registration



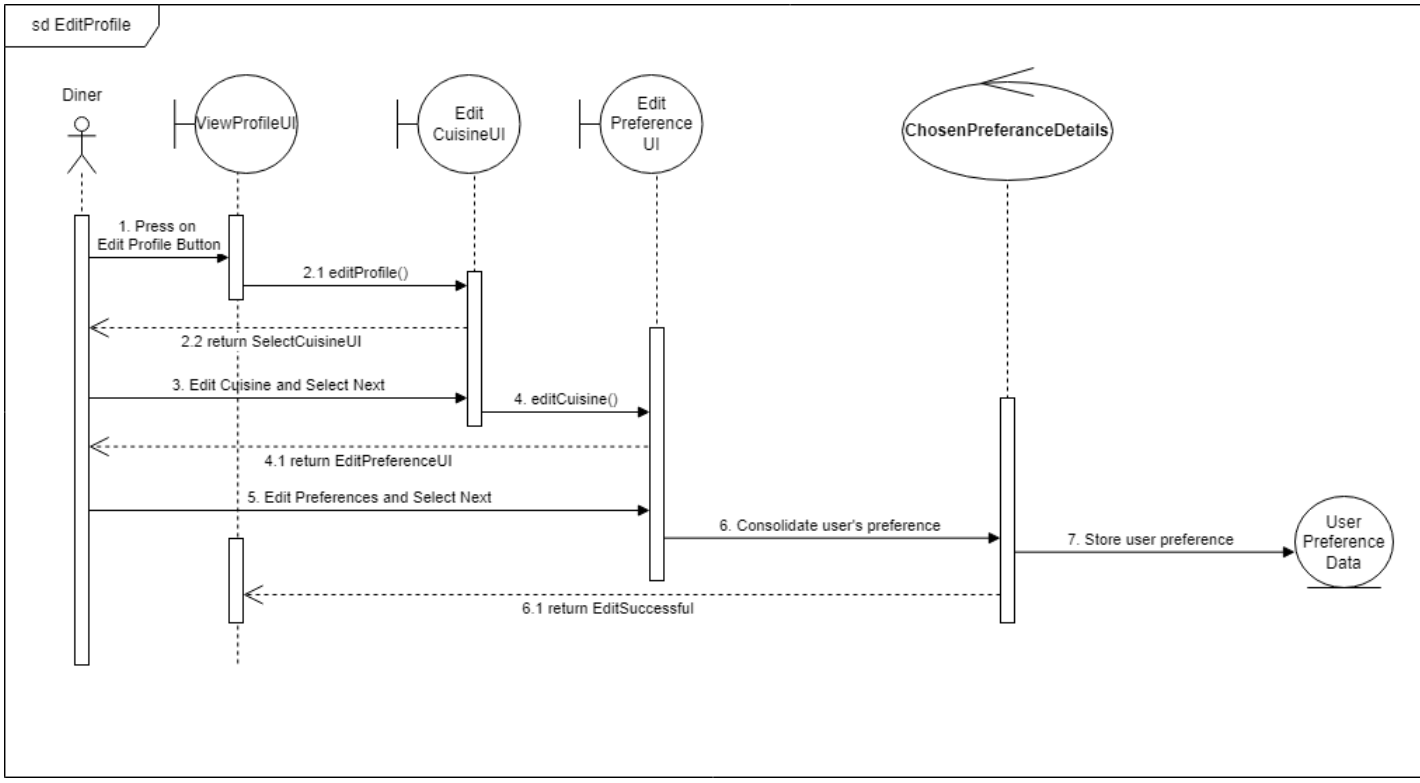
3.2 Login


sd Login
Diner
Welcome PageUI
LoginPageUI
Firebase DB
Home PageUI
1. Press on Login
2.1 Calls loginPage()
2.2 loginPage()
3. Input Username & Password
4.1 signInForm(details): boolean
alt
[if: invalid user credentials]
4.2.1 return InvalidUser
4.2.2 Display Invalid User Error
[else]
4.3.1 return ValidUser
4.3.2 Display Successful Login Message
5 Calls homePageUI()

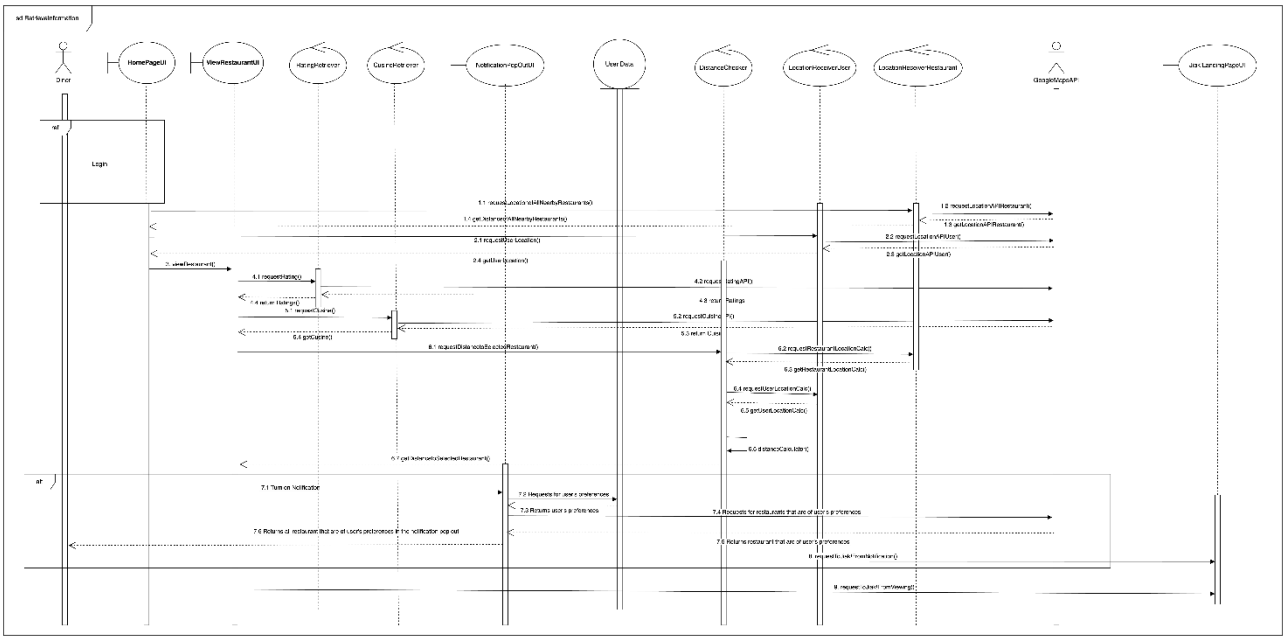
3.3 ViewProfile



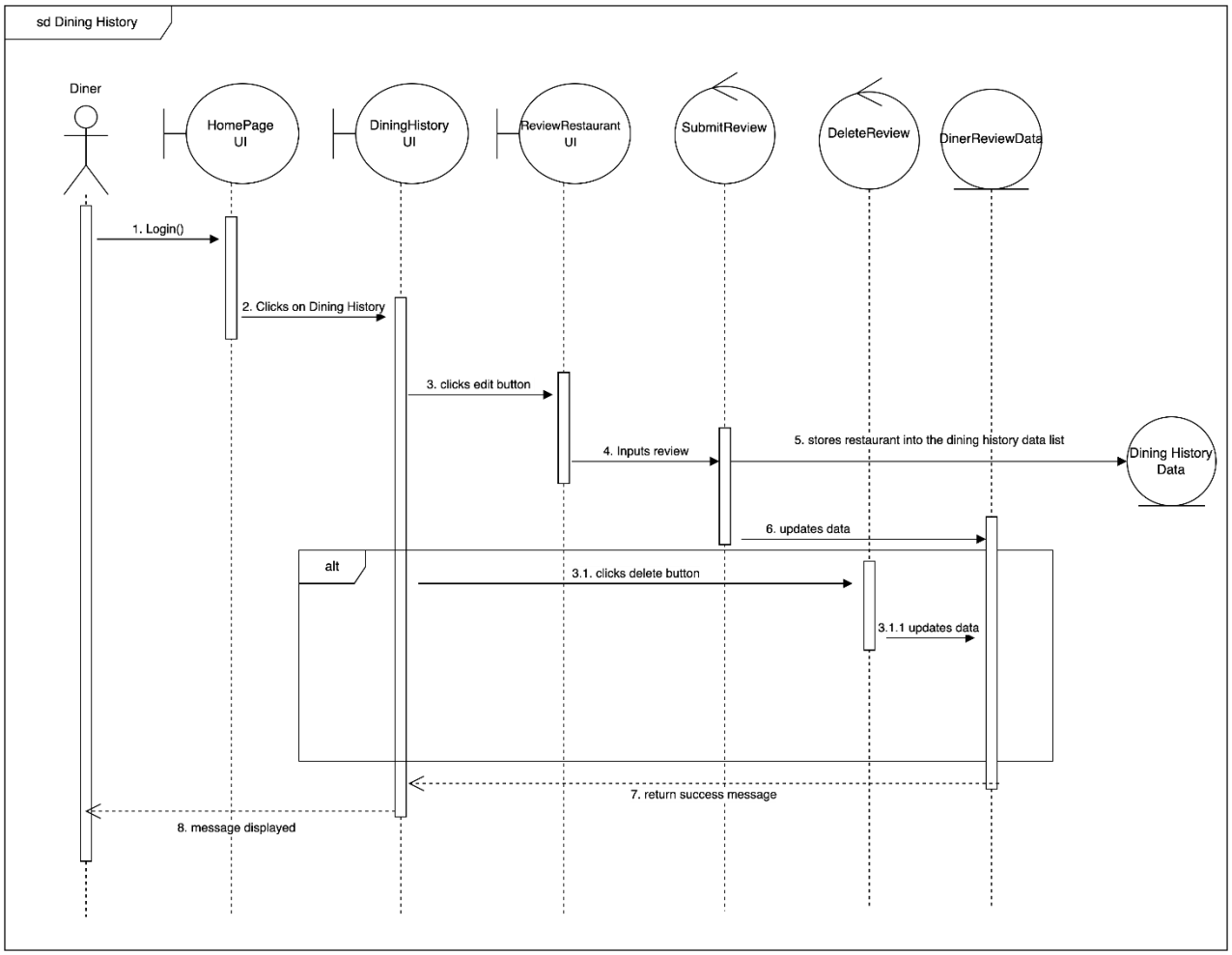
3.4 EditProfile



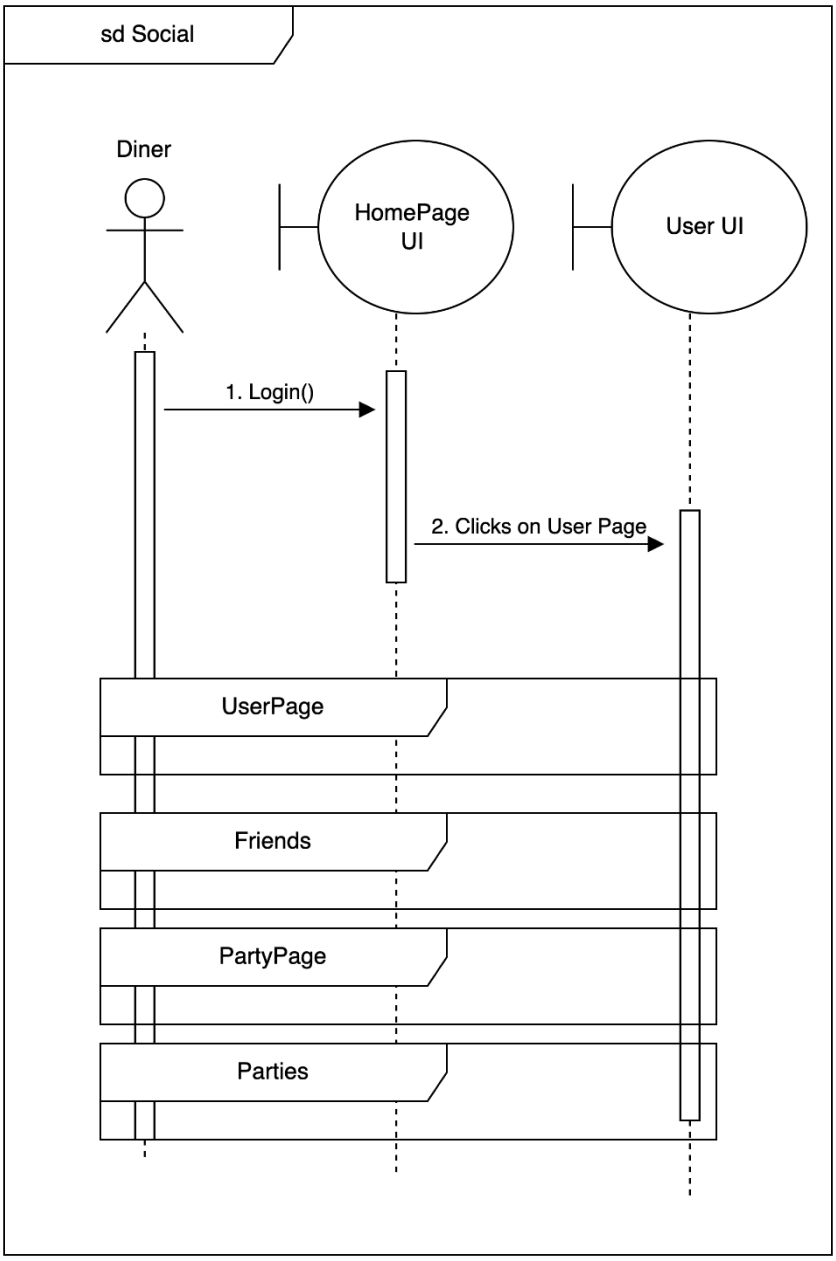
3.5 RetrieveInformation Diagram



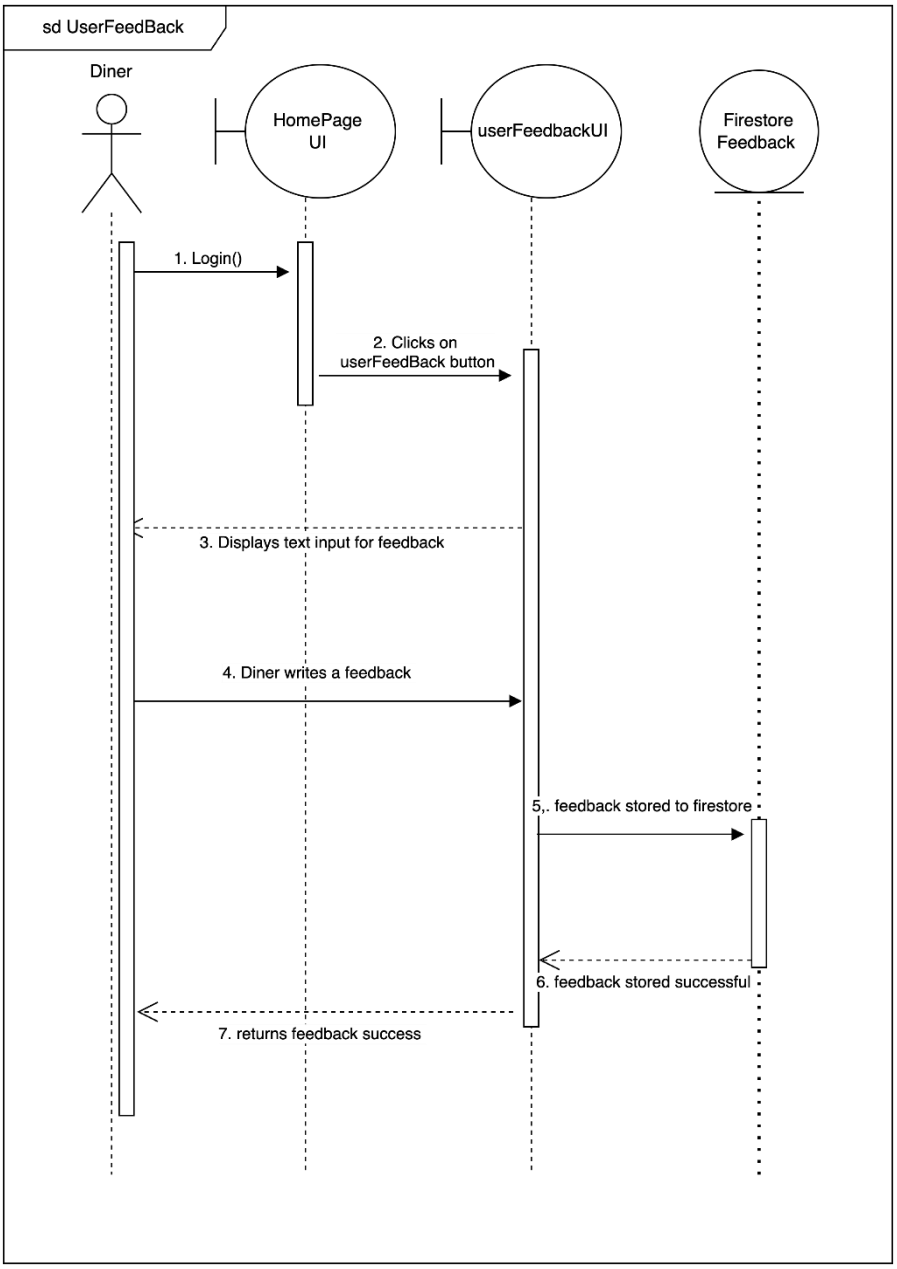
3.6 DiningHistory



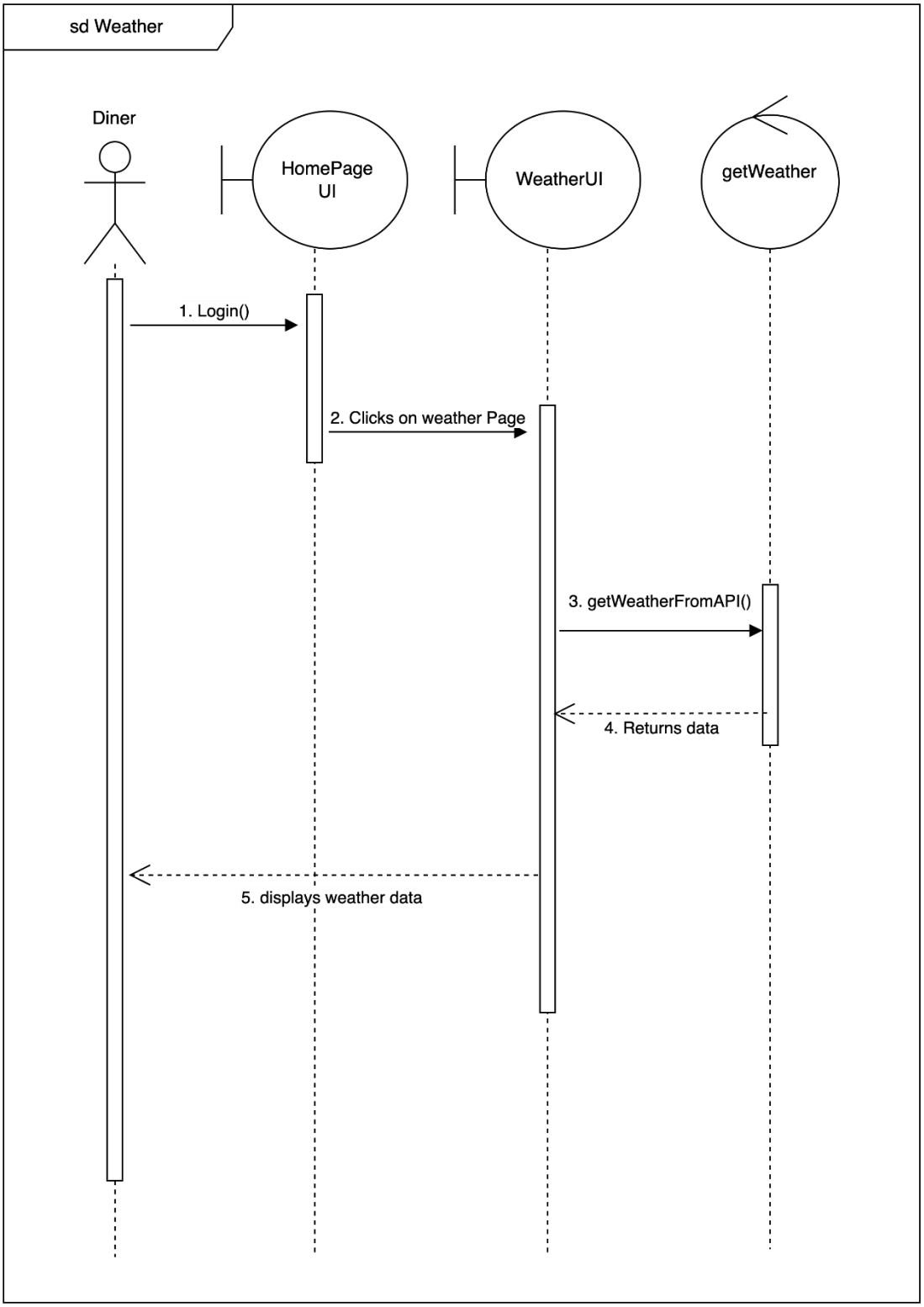
3.7 Social


sd Social
Diner
HomePage UI
User UI
1. Login()
2. Clicks on User Page
UserPage
Friends
PartyPage
Parties

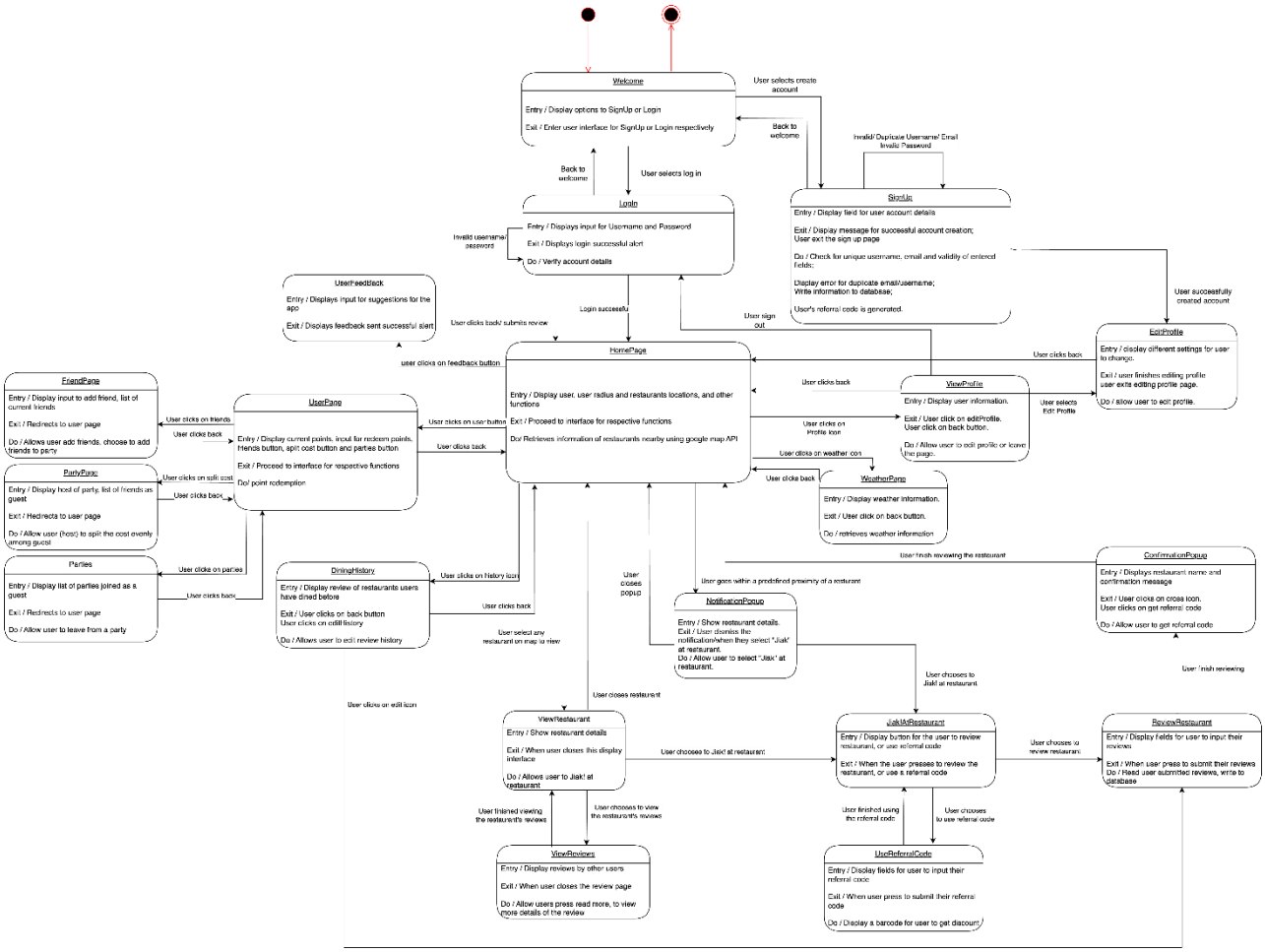
3.8 UserFeedBack



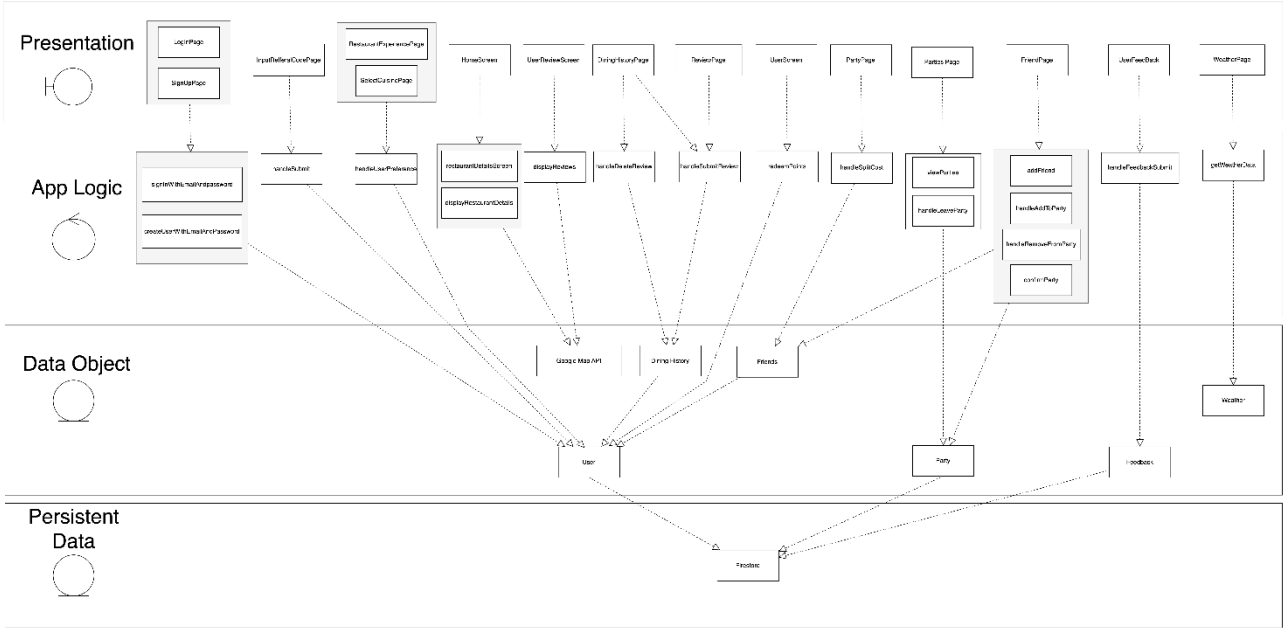
3.9 WeatherPage



4. State Machine Diagram


Welcome
Entry / Display options to SignUp or Login
Exit / Enter user interface for SignUp or Login respectively
User selects create account
Back to welcome
Back to welcome
User selects log in
Invalid/ Duplicate Username/ Email
Invalid Password
Login
Entry / Displays input for Username and Password
Exit / Displays login successful alert
Do / Verify account details
Invalid username/ password
SignUp
Entry / Display field for user account details
Exit / Display message for successful account creation;
User exit the sign up page
Do / Check for unique username, email and validity of entered fields;
Display error for duplicate email/username;
Write information to database;
User's referral code is generated.
User successfully created account
UserFeedBack
Entry / Displays input for suggestions for the app
Exit / Displays feedback sent successful alert
User clicks back/ submits review
Login succeeful
User sign out
EditProfile
Entry / display different settings for user to change;
Exit / user finishes editing profile user exits editing profile page.
Do / allow user to edit profile.
User clicks back
HomePage
Entry / Display user, user radius and restaurants locations, and other functions
Exit / Proceed to interface for respective functions
Do/ Retrieves information of restaurants nearby using google map API
user clicks on feedback button
User clicks back
ViewProfile
Entry / Display user information.
Exit / User click on editProfile.
User click on back button.
Do / Allow user to edit profile or leave the page.
User selects Edit Profile
User clicks on Profile icon
User clicks on weather con
User clicke back
WeatherPage
Entry / Display weather information.
Exit / User click on back button.
Do / retrieves weather information
FriendPage
Entry / Display input to add friend, list of current friends
Exit / Redirects to user page
Do / Allows user add friends, choose to add friends to party
User clicks on friends
User clicks back
UserPage
Entry / Display current points, input for redeem points, friends button, split cost button and parties button
Exit / Proceed to interface for respective functions
Do/ point redemption
User clicks on user button
User clicke back
PartyPage
Entry / Display host of party, list of friends as guest
Exit / Redirects to user page
Do / Allow user (host) to split the cost evenly among guest
User clicks on split cost
User clicke back
Parties
Entry / Display list of parties joined as a guest
Exit / Redirects to user page
Do / Allow user to leave from a party
User clicke on parties
User clicks back
DiningHistory
Entry / Display review of restaurants users have dined before
Exit / User clicks on back button
User clicks on edit history
Do / Allows user to edit review history
User clicks on history icon
User clicks back
User select any restaurant on map to view
User finish reviewing the restaurant
ConfirmationPopup
Entry / Displays restaurant name and confirmation message
Exit / User clicks on cross icon.
User clicks on get referral code
Do / Allow user to get referral code
User goes within a predefined proximity of a restaurant
User closes popup
NotificationPopup
Entry / Show restaurant details.
Exit / User dismiss the notification/when they select "Jiak" at restaurant.
Do / Allow user to select "Jiak" at restaurant.
User chooses to Jiak! at restaurant
User finish reviewing
User clicks on edit icon
User closes restaurant
ViewRestaurant
Entry / Show restaurant details
Exit / When user closes this display interface
Do / Allows user to Jiak! at restaurant
User chooses to Jiak! at restaurant
JiakIARestaurant
Entry / Display button for the user to review restaurant, or use referral code
Exit / When the user presses to review the restaurant, or use a referral code
User chooses to review restaurant
ReviewRestaurant
Entry / Display fields for user to input their reviews
Exit / When user press to submit their reviews
Do / Read user submitted reviews, write to database
User finished viewing the restaurant's reviews
User chooses to view the restaurant's reviews
ViewReviews
Entry / Display reviews by other users
Exit / When user closes the review page
Do / Allow users press read more, to view more details of the review
User finished using the referral code
User chooses to use referral code
UseReferralCode
Entry / Display fields for user to input their referral code
Exit / When user press to submit their referral code
Do / Display a barcode for user to get discount

5. System Architecture



Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

Source:
http://www.frontiernet.net/~kwiegers/process_assets/srs_template.doc