Software Requirements Specification

for

GoWhere

Version 1.0 approved

Prepared by

Keith Lim En Kai (U2220506C)

Lee Seungju (U2221620J)

Jesica Tjan (U2221011J)

Jerrell Yeoh Shao-En (U2220449D)

Lim Shaojun (U2221218F)

CodeCrafters

12 March 2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This Software Requirement Specification (SRS) document is created for the “GoWhere” web application, version 1.0. The primary goal of this SRS document is to outline the requirement specifications for the HawkerHub web application, streamlining the development and production processes for all stakeholders involved. Every aspect including: system features, limitations, non-functional requirements and interface specifications of the web application is documented within this SRS document.

1.2 Document Conventions

This section describes the conventional standards used throughout this document. All readers must pay attention to the standards listed in this section.

- Font: Arial
- Heading 1: Size 18
- Heading 2: Size 14
- Heading 3: Size 12
- Normal Text: Size 11
- Technical Standards: IEEE 830-1998

Refer Appendix A: Data Dictionary for the definitions of special terms used throughout this documentation.

1.3 Intended Audience and Reading Suggestions

This document is designed for various stakeholders, encompassing GoWhere web application users, the GoWhere development and testing teams, project managers, and the GoWhere marketing team. It commences by articulating the purpose of the GoWhere web application and outlining conventions applied throughout. Following this, it presents a high-level overview of the application's functionalities, along with design constraints and assumptions. Subsequently, the document delineates the interface requirements. Finally, it provides an in-depth exploration of the system features and non-functional requirements.

All stakeholders are advised to initiate their review with Section 1.1 (Purpose), 1.2 (Document Conventions), and Appendix A (Data Dictionary) to familiarize themselves with the web application's purpose, documentation standards, and technical term definitions.

The GoWhere development team is strongly urged to delve into Section 2 (Overall Description) for a comprehensive understanding of application functionalities, design, and constraints. Following this, Section 4 (System Features) provides developers with insights into each system feature to be incorporated. Lastly, developers should consult Section 3 (External Interfaces Requirements) and 5 (Other Nonfunctional Requirements) to grasp the specified requirements for the application's optimal functioning.

Conversely, users of the GoWhere web application, the GoWhere testing team, project managers, and the GoWhere marketing team are encouraged to proceed with a sequential reading of this document.

1.4 Product Scope

1.5 References

Docs:

React.js https://react.dev/learn

Express.js https://expressjs.com/

Firebase. (2024, April 15). Firebase documentation. Firebase Docs. https://firebase.google.com/docs

Google. (n.d.). Google. https://developers.google.com/maps/documentation/places/web-service/overview

Mapbox
https://docs.mapbox.com/mapbox-gl-js/api/

2. Overall Description

2.1 Product Perspective

GoWhere is an innovative, new and self-contained web application that aims to revolutionise the way people discover, access, and interact with { }. It provides users with a new and random location suggestion based on the users preference .

2.2 Product Functions

The system features of the GoWhere web application can be categorized into two primary subgroups: Accounts and Main. This segment offers a broad overview of the system features offered by the web application. Detailed information about each feature, including activity flows, conditions, assumptions, and accomplished functional requirements, is available in Section 4, titled "System Features."

2.2.1 Accounts:

1. Create Account

2. Account Login
3. View Account
4. Change Password
5. Change Profile Picture

2.2.2 Main:

1. The Application allows the User to state their preferences of a location through a 4-question questionnaire.
2. The Application allows the User to Randomly Search for a location based on their preferences.
3. The Application allows the User to view the location of the given location on the map.
4. The Application allows the User to view the location details of the given location.
5. The Application allows the User to save the given location.
6. The Application allows the User to view their saved locations.
7. The Application allows the User to unsave their saved locations.
8. The Application allows the User to view the current weather at the location.
9. The Application prompts the User to bring an Umbrella if it is currently raining at the given location.
10. The Application allows the User to view more reviews about the location.

2.3 User Classes and Characteristics

GoWhere expects its userbase to primarily be the following, ranked in order of their priority.

2.3.1 Locals (Singaporeans/PRs)

Attributes:	Description
Frequency of Use:	High
Functions Used:	1.
Technical Ability:	Medium
Characteristics:	Users of this demographic will use the web application to find a new location to visit based on their indicated preferences through the questionnaire.

2.4 Operating Environment

GoWhere follows a mobile-first approach and is designed to operate on devices equipped with a web browser. It is intended to be compatible with both Android and iOS platforms. For Android,

GoWhere should support Android OS versions from 6.0 (Marshmallow) onwards, while for iOS, compatibility is expected from iOS 10 and later versions.

Internet connectivity is essential for GoWhere to execute all its features seamlessly. The application should function smoothly over both Wi-Fi and mobile data networks (3G, 4G, 5G), ensuring users can access its services without interruption.

To provide accurate recommendations based on the user's current location, GoWhere requires access to the device's location services.

GoWhere adopts a responsive design, allowing it to adapt to various viewports. This ensures a consistent and user-friendly experience across different devices, maintaining functionality irrespective of the device being used.

2.4.1 Production Environment

2.4.2 Development Environment

2.5 Design and Implementation Constraints

2.5.1 Limitations

2.5.2 Design Standards

2.5.3 Regulatory Compliance

2.6 User Documentation

2.7 Assumptions and Dependence

2.7.1 APIs

2.7.1.1 Mapbox API

Dependency: Web application integrates Mapbox for mapping and geological services (eg. finding nearby locations within 2km radius)
Implication: Changes or disruption to Mapbox may affect web application's mapping features and the ability to locate and provide information about nearby locations.

2.7.1.2 Data.gov Weather API

Dependency: Web application relies on Data.gov (Weather API) to obtain information about
Implication: Changes or disruption to Data.gov (Weather API) may affect information regarding

2.7.1.3 Google Places API

2.7.2 Firebase Database

Dependency: Web application relies on Firebase DB to store and retrieve user related data
Implication: Changes or disruption to Firebase DB may affect the web application's data management and retrieval functionalities

2.7.3 Internet Connection

2.7.4 Location Services

3. External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 Create Account

4.1.1 Description and Priority

Users can register for an account using their email and password at the Create Account Page.
Priority: High

4.1.2 Stimulus/Response Sequences

Flow of Events:	1. User clicks on the “Create Account” button to create an account. 2. The user will be redirected to the Account Creation page to register for a new account. 3. They will have to enter a valid email and password, where their email will be parsed and the part before the “@” will act as their username 4. They will also have to retype their password to confirm it in the confirm password input. 5. After the User clicks on “create account”, a verification email will be sent to their email. 6. After being verified, the User will be brought to the Account Details Page.
Alternative Flows:	AF-S1: User accidentally clicks on the “Sign in” button 1. The user will be redirected to the Sign In page to log into their account. (2.2 Account Login) 2. They will have the option to click “Create Account” button to be go back. AF-S3: User has an existing account with the provided email. 1. System prompts User to use a different email.

4.1.3 Functional Requirements

1. The User must be able to create a new account using the sign-up page.
    1.1. During registration, the User must provide an email and a password.
        1.1.1. The System must verify that the email provided is valid.
            1.1.1.1. The Email provided must have a string before and after the “@” Character.
            1.1.1.2. The Email provided must end with either a “.com” or “.sg”
            1.1.1.3. The System must verify if the email provided is already in the database
                1.1.1.3.1. If the email is already in the database, the System will prompt the user to enter another email.
        1.1.2. The System must verify that the password provided is valid.
            1.1.2.1 The Password provided must be an alpha-numeric string of length greater than 5 and less than 30 characters long.
    1.2. To verify the user’s account details, the System must send a verification email to the user’s email
        1.2.1. The Email must contain a link which redirects them back to our website.
2. Upon successful creation, the User must be redirected to the questionnaire page.

4.2 Account Login

4.2.1 Description and Priority

Users can log in to their account using their registered email and password.
Priority: High

4.2.2 Stimulus/Response Sequences

Flow of Events:	1. User launches GoWhere Web Application. 2. User clicks on the “Sign In” button at the top right-hand corner and is redirected to the Sign in Page 3. User enters their email and password and clicks the “Sign In” button to login. 4. System verifies that the User has a valid account. 5. User is brought to the Questionnaire Page of the application.
Alternative Flows:	AF-S2: User clicks on the “Create Account” 1. The user will be redirected to the Account Creation page to register their account. (2.1 Create Account) 2. They will have the option to click “Go back” which returns them back to the homepage. AF-S2: User forgets their account password 3. User clicks on “Forget Password” 4. User will be redirected to the Forgot Password page (2.3 Reset Password) AF-S4: User enters a invalid email or password 5. User will be prompted to enter a valid email or password 6. Once a valid email and password is provided, the user will be brought to the homepage of the application.

4.2.3 Functional Requirements

1. The User must be able to login to the system using their Account.
    1.1. The User must provide their email and password to login
    1.2. The System must verify that the email and password provided are correct
        1.2.1. The Email and Password provided by the User must match the existing user details in the database
        1.2.2. If the email or password entered is invalid, an Error Message must be shown on screen to prompt the user to re-enter their email and password.
2. Upon successful login, the User must be redirected to the questionnaire page.

4.3 Reset Password

4.3.1 Description and Priority

Users would be able to reset their password if they forget their current password.
Priority: Medium

4.3.2 Stimulus/Response Sequences

Flow of Events:	1. User keys in their email. 2. User clicks on “Send Verification Link”, which will be sent to their email. 3. Upon clicking the verification link sent to their email, the User is redirected to the reset password page. 4. After clicking “submit”, new password will be saved to the database. 5. User is redirected to the Account Login page to login.
Alternative Flows:	AF-S2: Email User provided does not exist in the database. 1. A pop up message will appear stating “You do not have an account with us! Please create an account first!”

4.3.3 Functional Requirements

1. Users must key in their account’s email.

    1.1. System must verify the account corresponding to the email provided exists in the database.

        1.1.1. If email does not exist in the database, System must display a pop up message stating “You do not have an account with us! Please create an account first!”.

2. To verify the user’s account details, the System must send a verification email to the user’s email

    2.1. The Email must contain a link which redirects them back to our website.

3. Upon successful verification, the User must be redirected to the Reset Password page.

4. User must key in their new password.

    4.1. The new password must be keyed in again in the confirm password section and must match the new password.

5. After the “submit” button is clicked, the old password must be replaced by the new password in the database.

4.4 View Account

4.4.1 Description and Priority

User can view the email and password associated with their account, and choose to change their password or view their saved locations.
Priority: Medium

4.4.2 Stimulus/Response Sequences

Flow of Events:	1. User clicks on the “Account” (depicted by their 2. Application loads in the User account page containing their email and password (hidden by ‘*’). 3. If the user clicks on change password, they will be redirected to the change password page. 4. If the user clicks on saved locations, they will be redirected to the saved locations page. 5. If the user clicks on the Upload Image, they will be able to change their profile picture.
Alternative Flows:	-

4.4.3 Functional Requirements

1. Users must be able to view their account.
    1.1. User Accounts must display their account information: email, password.
        1.1.1. Password must be hidden by ‘*’.
        1.1.2. Users must display a link to change their password.
    1.2. User Accounts must display a link to the list of the Users’ Saved Locations
    1.3. User Accounts must allow Users to upload an image for their profile picture.

4.5 Change Password

4.5.1 Description and Priority

Users would be able to edit and change their account's password once logged in.
Priority: Low

4.5.2 Stimulus/Response Sequences

Flow of Events:	1. User keys in their old and new password. 2. User confirms their new password by keying it in one more time and clicks on the "Change Password" button 3. User's new password replaces the old password in the database 4. User is redirected back to the Account Details page.
Alternative Flows:	AF-S2: The old and new passwords are identical 1. The System will prompt the User to key in a password that is different from the previous password.

4.5.3 Functional Requirements

1. Users must key in their old and new passwords.

    1.1. If the old and new passwords keyed in are identical, the System will display a pop-up message stating "Passwords are identical! Change denied"

        1.1.1. The new password must be keyed in again in the confirm password section and must match the new password.

2. After the "submit" button is clicked, the old password must be replaced by the new password in the database.

4.6 Random Search

4.6.1 Description and Priority

Users will be posed a series of questions to help them get a location suggestion based on their preferences.
Priority: High

4.6.2 Stimulus/Response Sequences

Flow of Events:	The questions will be flashed one by one for a total of 4 questions, and questions are switched to the next question upon the completion of the current question. 1. 1st Question: User to pick 1 general category (Food, Shopping, Activities, Entertainment, Others). 2. 2nd Question: User is given a list of categories that have been narrowed down based on the general category they picked. 3. From this list, the User can exclude as many of the categories as they wish by ticking the category they want to exclude. 4. 3rd Question: User to pick their budget from a list of price ranges. 5. 4th Question: User to allow System to use their Current Location and a list of ranges of proximity of locations from their location. 6. Upon submission of questionnaire, the User will be brought to the Map page (2.4 View Map and Location Details) where a random location that fulfills the above criteria will be generated for them along with the location details.
Alternative Flows:	AF-S1: User wants to change their input in a previous question(s). 1. User can go back to the previous questions using the “Go Back” button to cycle through their questions.
Exceptions	EX1: No location which meets all criterias is found. 1. A pop-up “No location can be found! Time to edit your criteria!” will be shown. 2. The user will close the pop-up. 3. The user can then proceed to edit his/her criteria through the drop-downs to edit their criterias.

4.6.3 Functional Requirements

1. The System must provide a total of 5 questions to the User.
    1.1 The User must be logged in to access the questionnaire.
    1.2. The 1st Question must allow the User to pick at least one general category from 5 categories.
        1.2.1. The 5 categories are Food, Shopping, Entertainment, Activities, and Others.

1.3. The 2nd Question must allow the User to pick any specific categories to exclude from the general category.

1.3.1. The User does not have to tick any of the categories if they do not want to exclude any categories.

1.4. The 3rd Question must allow the User to pick their budget from a list of differing price ranges.

1.5. The 4th Question must allow the User to pick from a list the range of distances they are willing to travel.

1.5.1. Before the user is able to submit this question, the System must display a pop-up message asking the user for permission to use their current location.

2. The user must click on the “Next” button to move on to the next question.

2.1. If the “Go Back” button is clicked instead, the page shall change to the previous question.

3. After filling in the questionnaire and clicking the “Submit” button, the System must change its page to the interactive map page. (next section 2.4)

3.1. If at least one location meets the criterias derived from the questionnaire, the location must be pinned and shown on the map page.

3.2. If no location which meets all criteria is found, the System shall display a pop-up message stating “No location can be found! Time to edit your criteria!”.

3.2.1. The User can then edit the location criteria through the drop-down on the map page.

4.7 View Map and Location Details

4.7.1 Description and Priority

Users will see details of the randomized place they have been given, such as its name, ratings and reviews, address, and location on the map.
Priority: High

4.7.2 Stimulus/Response Sequences

Flow of Events:	1. Based on the questionnaire, Users are allocated a valid randomized location based on their preferences stated. 2. This location is then displayed on a Map page along with the location details in a sidebar. 3. Users can click on the Google Maps link to find directions to the location 4. Users can click on the bookmark icon to save this location in their account.
Alternative Flows:	AF-S1.1: User feels that the suggested location is not suitable and wants another location. 1. Users can click on the “I’m feeling lucky” button to be provided 2. Users will be provided with a new randomized location AF-S1.2: User changes their mind on the type of location wanted. 1. Users can change the criteria of location in the dropdown menu. 2. Users must click on the “I’m feeling lucky” button to be given a new location with modified criteria.

4.7.3 Functional Requirements

1. The System must display an interactive map that shows the allocated random location.
    1.1. The Map must be able to grow and shrink around the User’s location.
    1.2. The User must have done the questionnaire before being re-directed to the Map.
2. The System must display the details of the allocated random location.
    2.1. These details include: Reviews, Ratings, Descriptions, Estimated Budget, and Address.
        2.1.1. System must pull these details through the Google Places API using the name of the location.
3. The System must display a dropdown allowing users to change the criteria for their preferred location.
    3.1. The Dropdown should contain the same criteria as the questionnaire.

4.8 View More Reviews

4.8.1 Description and Priority

Users can view more reviews about a location if they want to know more reviews about the location.
Priority: Low

4.8.2 Stimulus/Response Sequences

Flow of Events:	1. At the Map Page, User clicks on the “Read More Reviews” button. 2. System will call the Google Places API and pull the reviews for the specified location. 3. System will then display the additional information pulled from the Google Places API in another page.
Alternative Flows:	-

4.8.3 Functional Requirements:

1. Users must be able to view more reviews of a specific location.
    1.1. These details include: Reviews, Ratings and the Reviewer’s Name.
        1.1.1. System must pull the details using the Google Places API using the name of the location.

4.9 View Current Weather

4.9.1 Description and Priority

User can view the current weather at the given location. Users would be prompted to bring an umbrella if the weather forecast predicts rainy weather.
Priority: Low

4.9.2 Stimulus/Response Sequences

Flow of Events:	1. Using the location or neighborhood of the given location, System retrieves the weather using the Gov 2-Hour Weather API. 2. The System displays the current weather conditions of the location. 3. If the system receives information that there is going to be or is raining, the system will remind the User to bring an Umbrella.
Alternative Flows:	-

4.9.3 Functional Requirements

1. Whenever the user gets allocated a random location, the System must check the weather for the next 2 hours using the Weather Forecast API.

    1.1. If the weather forecast predicts that it will rain within the next 2 hours, the System will display a message stating “It might rain, please bring an umbrella!” under the location details.

    1.2. If the weather forecast does not predict any rain within the next 2 hours, the System will not show the warning message.

4.10 Save Locations

4.10.1 Description and Priority

Users would be able to save they want to revisit in the future and remove saved locations.
Priority: Medium

4.10.2 Stimulus/Response Sequences

Flow of Events:	1. In the Location Details on the Map Page, User clicks on the “Save Location” button (depicted by a white bookmark icon). 2. System adds the saved location to the User’s list of saved locations. 3. “Save Location” button changes to “Unsave Location” (Depicted by a black bookmark icon).
Alternative Flows:	AF-S2: The User has already saved the location to their saved list. 1. The “Save Location” button has been changed to “Unsave Location” (Black Bookmark Icon) 2. The User clicks on the “Unsave Location” button. 3. The System removes the location from the User’s saved locations list. 4. “Save Location” button changes back to the “Unsave Location”.

4.10.3 Functional Requirements:

1. Users must be able to save a location to their list of saved locations.
    1.1. The User must be logged into the Application before they can save a location.
2. If the location has already been saved before, Users must be able to unsave the location.
    2.1. The User must be logged into the Application before they can unsave a location.
    2.2. The System must change the “Save” option to the “Unsave” option.

4.11 View Saved Locations

4.11.1 Description and Priority

Users would be able to view previously saved locations.
Priority: Low

4.11.2 Stimulus/Response Sequences

Flow of Events:	1. Under the Account Details Page, User clicks on “View saved locations” 2. Users will be redirected to a page where they can view the list of their saved locations. Each location will have an “Unsave Location” button beside it. 3. If a user clicks on the saved location, it will open up the details of that location in a larger information card (Name, Address, Reviews, Ratings).
Alternative Flows:	AF-S3: The User wants to Unsave a location. 1. The User clicks on the “Unsave Location” button and the saved location is removed from the list.

4.11.3 Functional Requirements:

1. Users must be able to view their list of saved locations.
    1.1. Users must be logged in and re-directed from the View Profile page to view this page.
    1.2. Users must be able to click on a saved location to view more details about it.
        1.2.1. System must expand the information card to show more details.
2. Users must be able to remove saved locations from their list of saved locations.

5. Other Non-functional Requirements

Non-Functional Requirements describe the properties the system must have, that is not directly related to the functional behavior of the system.

5.1 Performance Requirements

1. The landing page must provide an 8-second or less response time in a desktop browser.
2. Each search result of a user's input must be displayed within 5 seconds.
3. The application must return the correct data to the user based on the selected value of inputs.3.2 Portability Requirements
1. The web application must be functional on the latest versions of any browser type and in the operating system MacOS, Microsoft Windows, Android and iOS.

5.2 Security Requirements

1. Upon account registration, the system must send a verification letter to the email account selected by the user.
2. The system must accept only a password with a minimum one for each special character, upper case and lower case, number.
3. The system must ensure a user enters a valid email address and password to log into the account.
4. Users must be logged in using their registered email before using the search functions of the application.

5.3 Availability

1. The web application must maintain at least a 98% daily uptime for users located in Singapore, under normal network conditions.

5.4 Localization

1. The date in the system should follow Singapore Standard Time (UTC+8:00).
2. The default language of the web application should be English.

5.5 Usability

1. Users must retrieve an output from the system within 1 minute after logging into the web application, assuming no error has occurred.
2. The system must return relevant results with an accuracy of at least 95% when using a search feature. Relevance is determined based on the distance, rating, and price chosen by the user.
3. Users must receive information regarding maintenance schedules through a dedicated screen that appears upon accessing the web application.

5.6 Reliability

1. The system must not exceed an error rate of 5% for the completion of common tasks (eg. selecting customized inputs used to randomly search restaurants).

5.7 Maintainability

1. The system must maintain a mean time between failures (MTBF) of at least 30 days.
2. The system must have a maximum mean time to repair (MTTR) of 3 hours for critical failures.

5.8 Testability

1. The system must be capable of executing a comprehensive set of automated tests covering at least 90% of the codebase within an hour following any modifications made to the codebase.

5.9 Reusability

1. At least 60% percent of the entire codebase shall be reusable in the future projects requiring the same API.

7. Use Case Model

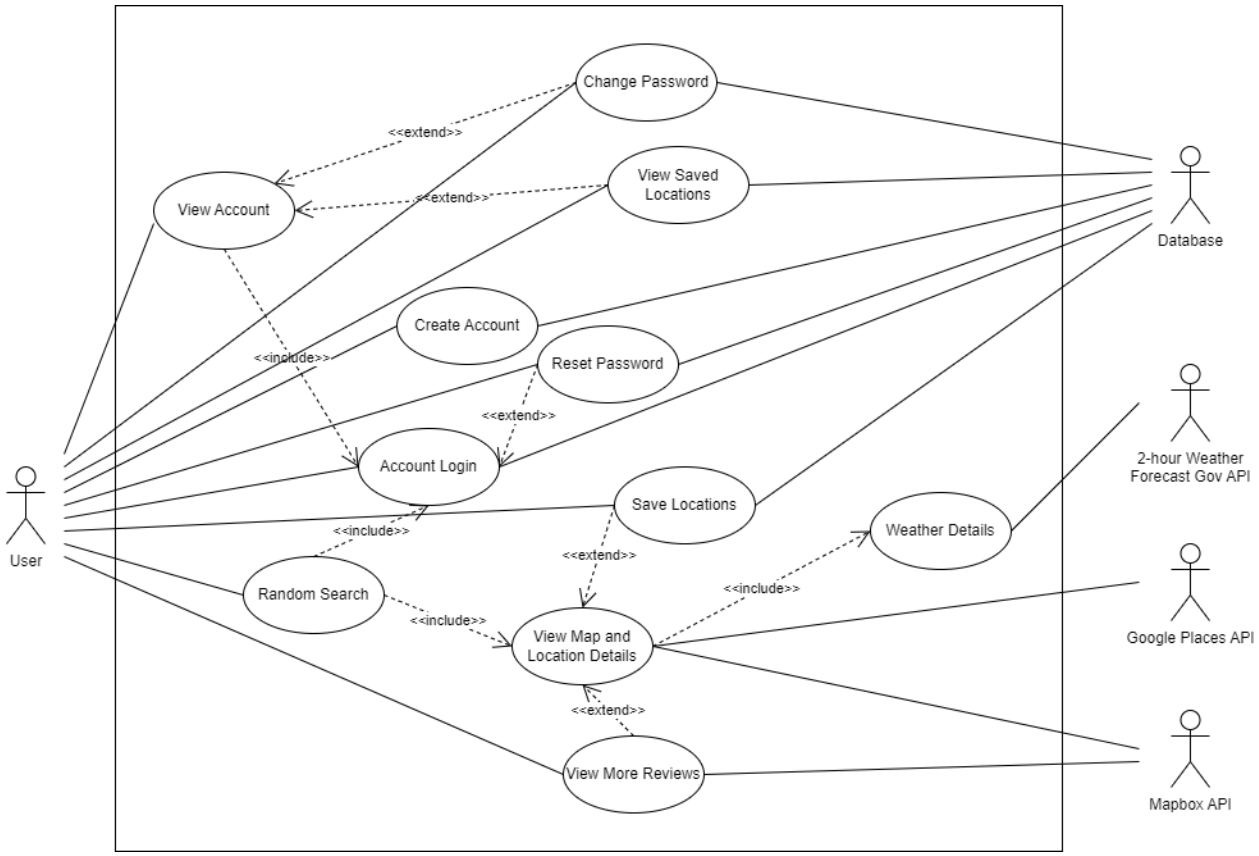


Figure 1: Use-Case Diagram

Use Case ID:	1		
Use Case Name:	Create Account		
Created By:	Shaojun	Last Updated By:	Keith
Date Created:	28/1/2024	Date Last Updated:	17/2/2024

Actor:	● User ● Database
Description:	Users can register for an account using their email and password.
Preconditions:	● User must provide a valid email and password to create an account. ● User must not have a GoWhere account with the email already. ● The password provided must be an alpha-numeric string of length greater than 5 and less than 30 characters.
Postconditions:	● User account created and stored in database ● Automatically logged in and re-directed to Account Details page
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User clicks on the “Create Account” button to create an account. 2. The user will be redirected to the Account Creation page to register for a new account. 3. They will have to enter their email and password, where their email will be parsed and the part before the “@” will act as their username 4. They will also have to retype their password to confirm it in the confirm password input. 5. After they click “Create Account”, a verification email will be sent to their email. 6. After being verified, the User will be brought to the Account Details Page.
Alternative Flows:	AF-S1: User clicks on the “Sign in” button 1. The user will be redirected to the Sign In page to log into their account. (2.2 Account Login) 2. They will have the option to click “Go back” which returns them to the homepage. AF-S3: The User has an existing account with the email

	they provided 1. System prompts User to use a different email.
Exceptions:	NIL
Includes:	NIL
Extends:	NIL
Special Requirements:	NIL
Assumptions:	NIL
Notes and Issues:	NIL

Use Case ID:	2		
Use Case Name:	Account Login		
Created By:	Shaojun	Last Updated By:	Keith
Date Created:	28/1/2024	Date Last Updated:	17/2/2024

Actor:	● User ● Database
Description:	Users can login to their account using their registered email and password.
Preconditions:	● User has a GoWhere account
Postconditions:	● User is directed to the questionnaire page
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User launches GoWhere Web Application. 2. User clicks on the “Sign In” button at the top right-hand corner and is redirected to the Sign in Page 3. User enters their email and password and clicks the “Sign In” button to login. 4. System verifies that the User has a valid account. 5. User is brought to the Questionnaire Page of the application.
Alternative Flows:	AF-S2: User clicks on the “Create Account” 1. The user will be redirected to the Account Creation page to register their account. (2.1 Create Account) 2. They will have the option to click “Go back” which returns them back to the homepage. AF-S2: User forgets their account password 1. User clicks on “Forget Password” 2. User will be redirected to the Forgot Password page. AF-S4: User enters a invalid email or password 1. User will be prompted to enter a valid email or password 2. Once a valid email and password is provided, the user will be brought to the homepage of the application.
Exceptions:	NIL
Includes:	NIL
Extends:	Reset Password
Special Requirements:	Email and Password are compulsory
Assumptions:	NIL
Notes and Issues:	NIL

Use Case ID:	3		
Use Case Name:	Reset Password		
Created By:	Shaojun	Last Updated By:	Keith
Date Created:	17/2/2024	Date Last Updated:	17/2/2024

Actor:	● User ● Database
Description:	Users would be able to reset their password if they forget their current password.
Preconditions:	● User has a GoWhere account ● User has forgotten their password
Postconditions:	● User has reset their password ● User is redirected to the Login Page
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. User keys in their email. 2. User clicks on “Send Verification Link”, which will be sent to their email. 3. Upon clicking the verification link sent to their email, the User is redirected to the reset password page. 4. After clicking “submit”, new password will be saved to the database. 5. User is redirected to the Account Login page to login.
Alternative Flows:	AF-S2: Email User provided does not exist in the database. 1. A pop up message will appear stating “You do not have an account with us! Please create an account first!”
Exceptions:	NIL
Includes:	NIL
Special Requirements:	Email and Password are compulsory fields
Assumptions:	NIL
Notes and Issues:	NIL

Use Case ID:	4		
Use Case Name:	View Account		
Created By:	Shaojun	Last Updated By:	Keith
Date Created:	3/2/2024	Date Last Updated:	17/2/2024

Actor:	● User ● Database
Description:	User can view the email and password associated with their account, and choose to change their password or view their saved locations or update their profile picture.
Preconditions:	● User has a GoWhere account ● User is signed into their account
Postconditions:	● User can change their password ● User can view their saved locations ● User can change their profile picture
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. User clicks on the “Account” (depicted by their 2. Application loads in the User account page containing their email and password (hidden by ‘*’). 3. If the user clicks on change password, they will be redirected to the change password page. 4. If the user clicks on saved locations, they will be redirected to the saved locations page. 5. If the user clicks on the Upload Image, they will be able to change their profile picture by uploading an image.
Alternative Flows:	NIL
Exceptions:	NIL
Includes:	Account Login
Extends:	View Saved Locations, Change Password
Special Requirements:	NIL
Assumptions:	NIL
Notes and Issues:	NIL

Use Case ID:	5		
Use Case Name:	Change Password		
Created By:	Shaojun	Last Updated By:	Shaojun
Date Created:	3/2/2024	Date Last Updated:	17/2/2024

Actor:	● User ● Database
Description:	Users would be able to edit and change their account's password.
Preconditions:	● User must be logged in ● User must click on the change password option on the account details page.
Postconditions:	● User's password will be changed in the database ● User will be redirected to the account details page
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1. User keys in their old and new password. 2. User confirms their new password by keying it in one more time and clicks on the "Change Password" button 3. User's new password replaces the old password in the database 4. User is redirected back to the Account Details page.
Alternative Flows:	NIL
Exceptions:	NIL
Includes:	NIL
Extends:	NIL
Special Requirements:	NIL
Assumptions:	NIL
Notes and Issues:	NIL

Use Case ID:	6		
Use Case Name:	Random Search		
Created By:	Shaojun	Last Updated By:	Keith
Date Created:	28/1/2024	Date Last Updated:	17/2/2024

Actor:	● User
Description:	Users will be posed a series of questions to help them get a location suggestion based on the preferences they provide.
Preconditions:	● User must be logged in
Postconditions:	● User will be directed to a map with details of a randomized location.
Priority:	High
Frequency of Use:	High
Flow of Events:	The questions will be flashed one by one for a total of 4 questions, and questions are switched to the next question upon the completion of the current question. 1. 1st Question: User to pick 1 general category (Food, Shopping, Activities, Entertainment, Others). 2. 2nd Question: User is given a list of categories that have been narrowed down based on the general category they picked. 3. From this list, the User can exclude as many of the categories as they wish by ticking the category they want to exclude. 4. 3rd Question: User to pick their budget from a list of price ranges. 5. 4th Question: User to allow System to use their Current Location and a list of ranges of proximity of locations from their location. 6. Upon submission of questionnaire, the User will be brought to the Map page (2.6 View Map and Location Details) where a random location that fulfills the above criteria will be generated for them along with the location details.
Alternative Flows:	AF-S1: User wants to change their input in a previous question(s). 1. User can go back to the previous questions using the “Go Back” button to cycle through their questions.
Exceptions:	EX1: No location which meets all criterias is found. 1. A pop-up “No location can be found! Time to edit your criteria!” will be shown. 2. The user will close the pop-up.

	3. The user can then proceed to edit his/her criteria through the drop-down menus to edit their criteria.
Includes:	Account Login, View Map and Location Details
Special Requirements:	NIL
Assumptions:	NIL
Notes and Issues:	NIL

Use Case ID:	7		
Use Case Name:	View Map and Location Details		
Created By:	Shaojun	Last Updated By:	Keith
Date Created:	3/2/2024	Date Last Updated:	17/2/2024

Actor:	● User ● Google Places API ● Mapbox API
Description:	User will see details of the randomised place they have been given, such as its name, ratings and reviews, address, and location on the map.
Preconditions:	● The Location given must be within Singapore ● A valid location must be generated by the questionnaire in order to search it on the Mapbox Map.
Postconditions:	● User can choose to be directed to Google Maps for directions. ● User can choose to save this location. ● User is shown the location on the Map Page ● User is shown the location details of the given location in a side bar
Priority:	High
Frequency of Use:	High
Flow of Events:	1. Based on the questionnaire, Users are allocated a valid randomized location based on their preferences stated. 2. This location is then displayed on a Map page along with the location details in a sidebar. 3. Users can click on the Google Maps link to find directions to the location 4. Users can click on the bookmark icon to save this location in their account.
Alternative Flows:	AF-S1.1: User feels that the suggested location is not suitable and wants another location. 1. Users can click on the “I’m feeling lucky” button to be provided 2. Users will be provided with a new randomized location AF-S1.2: User changes their mind on the type of location wanted.

	1. Users can change the criteria of location in the dropdown menu. 2. Users must click on the “I’m feeling lucky” button to be given a new location with modified criteria.
Exceptions:	NIL
Includes:	NIL
Extends:	Save Locations, Weather Details, View More Reviews
Special Requirements:	NIL
Assumptions:	NIL
Notes and Issues:	NIL

Use Case ID:	8		
Use Case Name:	View More Reviews		
Created By:	Shaojun	Last Updated By:	Shaojun
Date Created:	17/2/2024	Date Last Updated:	17/2/2024

Actor:	● User ● Google Places API
Description:	Users can view more reviews about a location if they want to know more reviews about the location.
Preconditions:	● User must have been given a randomised location. ● The location must have more than 1 review
Postconditions:	● User will be redirected to a page solely for reviews.
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1. At the Map Page, User clicks on the “Read More Reviews” button. 2. System will call the Google Places API and pull the reviews for the specified location. 3. System will then display the additional information pulled from the Google Places API in another page.
Alternative Flows:	NIL
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	NIL
Notes and Issues:	NIL

Use Case ID:	9		
Use Case Name:	View Current Weather		
Created By:	Shaojun	Last Updated By:	Shaojun
Date Created:	3/2/2024	Date Last Updated:	17/2/2024

Actor:	● User ● 2-hour Weather Forecast Gov API
Description:	Users would be prompted to bring an umbrella if the weather forecast predicts rainy weather at the time the User is given their randomized location.
Preconditions:	● User has been given a randomised location on the map
Postconditions:	● User is shown the current weather at the given location ● User is prompted to bring an umbrella if it is raining at the given location
Priority:	Low
Frequency of Use:	High
Flow of Events:	1. Using the location or neighborhood of the given location, System retrieves the weather using the Gov 2-Hour Weather API. 2. The System displays the current weather conditions of the location. 3. If the system receives information that there is going to be or is raining, the system will remind the User to bring an Umbrella.
Alternative Flows:	NIL
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	NIL
Notes and Issues:	NIL

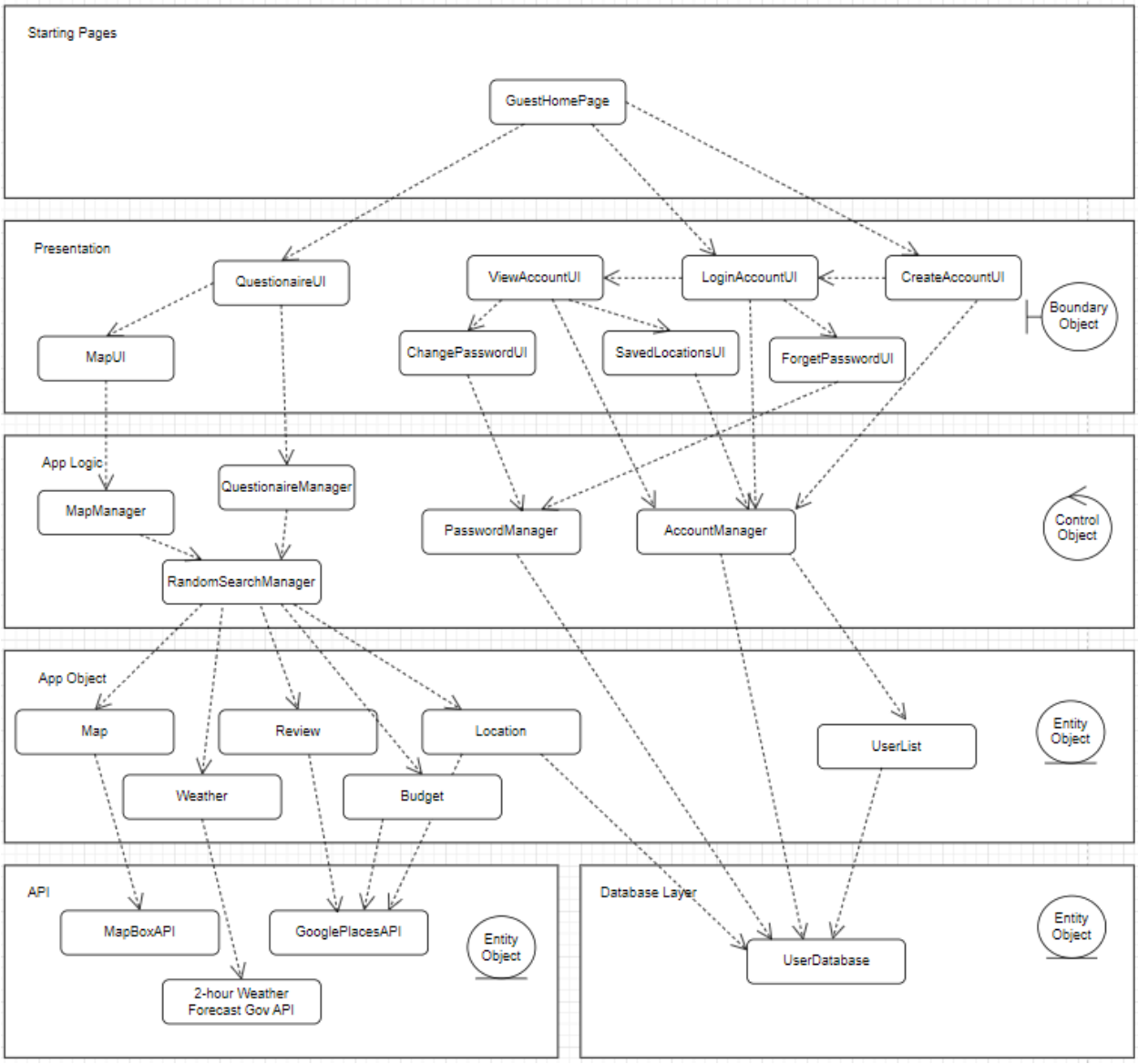
Use Case ID:	10		
Use Case Name:	Save Locations		
Created By:	Shaojun	Last Updated By:	Keith
Date Created:	3/2/2024	Date Last Updated:	17/2/2024

Actor:	● User ● Database
Description:	Users would be able to save they want to revisit in the future and remove saved locations.
Preconditions:	● User must be signed into their account
Postconditions:	● Location details will be added to Saved Locations page
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. In the Location Details on the Map Page, User clicks on the “Save Location” button (depicted by a white bookmark icon). 2. System adds the saved location to the User’s list of saved locations. 3. “Save Location” button changes to “Unsave Location” (Depicted by a black bookmark icon).
Alternative Flows:	AF-S2: The User has already saved the location to their saved list. 1. The “Save Location” button has been changed to “Unsave Location” (Black Bookmark Icon) 2. The User clicks on the “Unsave Location” button. 3. The System removes the location from the User’s saved locations list. 4. “Save Location” button changes back to the “Unsave Location”.
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	NIL
Notes and Issues:	NIL

Use Case ID:	11		
Use Case Name:	View Saved Locations		
Created By:	Shaojun	Last Updated By:	Shaojun
Date Created:	3/2/2024	Date Last Updated:	17/2/2024

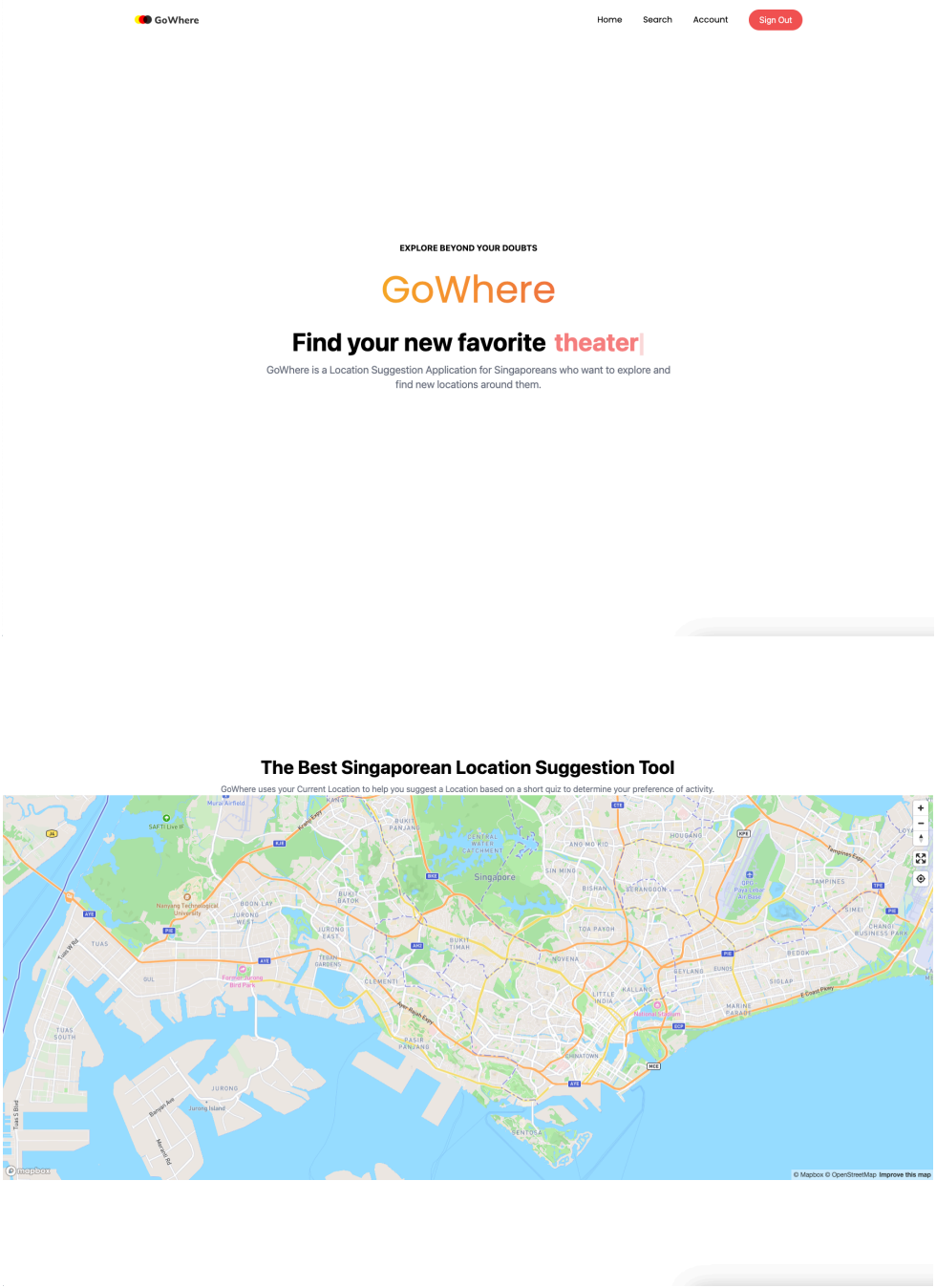
Actor:	● User ● Database
Description:	Users would be able to view previously saved locations.
Preconditions:	● User must be signed into their account
Postconditions:	NIL
Priority:	Low
Frequency of Use:	Medium
Flow of Events:	1. Under the Account Details Page, User clicks on “View saved locations” 2. Users will be redirected to a page where they can view the list of their saved locations. Each location will have an “Unsave Location” button beside it. 3. If a user clicks on the saved location, it will open up the details of that location in a larger information card (Name, Address, Reviews, Ratings).
Alternative Flows:	AF-S3: The User wants to Unsave a location. 1. The User clicks on the “Unsave Location” button and the saved location is removed from the list.
Exceptions:	NIL
Includes:	NIL
Extends:	NIL
Special Requirements:	NIL
Assumptions:	NIL
Notes and Issues:	NIL

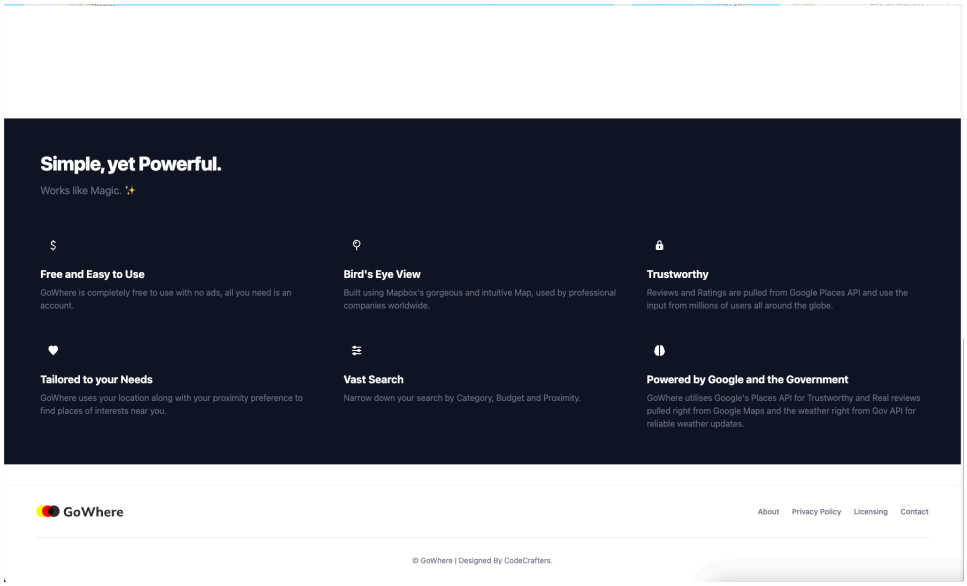
8. System Architecture - Lab 3 Deliverables



9. Application Skeleton - Lab 3 Deliverables

9.1 Home Page

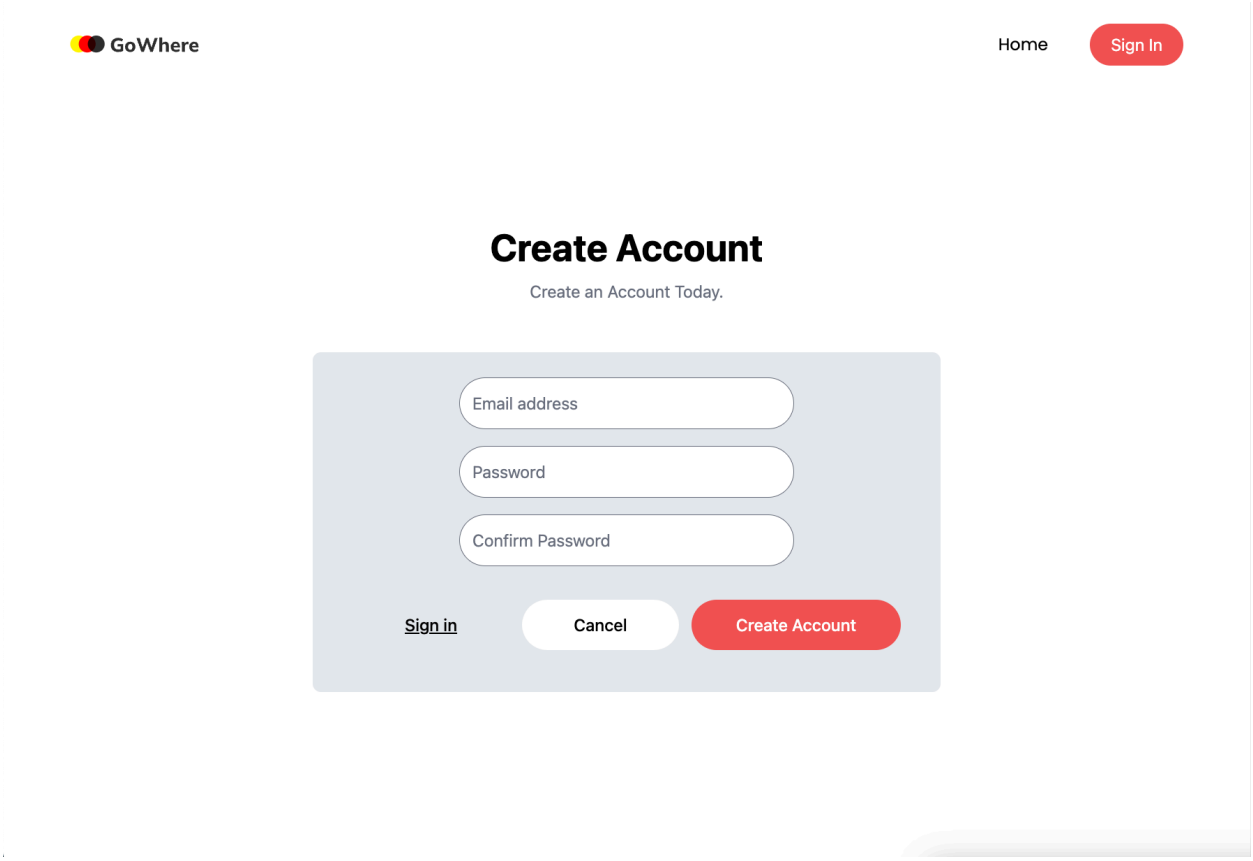



Simple, yet Powerful.
Works like Magic. ✨
Free and Easy to Use
GoWhere is completely free to use with no ads, all you need is an account.
Bird's Eye View
Built using Mapbox's gorgeous and intuitive Map, used by professional companies worldwide.
Trustworthy
Reviews and Ratings are pulled from Google Places API and use the input from millions of users all around the globe.
Tailored to your Needs
GoWhere uses your location along with your proximity preference to find places of interests near you.
Vast Search
Narrow down your search by Category, Budget and Proximity.
Powered by Google and the Government
GoWhere utilises Google's Places API for Trustworthy and Real reviews pulled right from Google Maps and the weather right from Gov API for reliable weather updates.
GoWhere
About
Privacy Policy
Licensing
Contact
© GoWhere | Designed By CodeCrafters.

```
import styles from './style';
import {Hero} from './components';
import Features from './components/Features';
import { Link } from 'react-router-dom';
import Map from './mapbox/Map';
import React, { useState, useEffect } from 'react';
import { onAuthStateChanged } from "firebase/auth";
import { auth } from './firebase/firebase';

export const Home = () => {

  const [loggedin, setLoggedin] = useState(false);

  useEffect(()=>{
    onAuthStateChanged(auth, (user) => {
        if (user) {
          const uid = user.uid;
          console.log(uid)
          setLoggedin(true);
          // ...
        } else {
          setLoggedin(false);
        }
      });

  }, [])
```

```
return (
  <div className="bg-white w-full overflow-hidden">
    <div className={`bg-white ${styles.flexStart}`}>
      <div className={`${styles.boxWidth}`}>
        <Hero/>

      </div>
    </div>

    <div className={`bg-white ${styles.paddingX} ${styles.flexStart}`}>
      <div className={`${styles.boxWidth}`}>

      </div>
    </div>

    <hr/>

    <div className='w-full h-screen mx-auto text-center flex flex-col justify-center'>
      <div className='grid grid-cols-1 grid-row'>
          <div className=''>
              <h1 className='md:text-4xl text-l font-bold text-black-500'>The Best Singaporean Location Suggestion Tool</h1>
              <p className='md:text-m text-l font-sm text-gray-500 mt-[10px]'>
              GoWhere uses your Current Location to help you suggest a Location based on a short quiz to determine your preference of activity.
              </p>
              <Map/>

          </div>
        </div>
    </div>

    <Features/>

    {loggedin ? null
    : <div className='w-full my-20  mx-auto text-center flex flex-col justify-center'>
    <div className='grid grid-cols-1 grid-row'><div className=''>
    <h1 className='md:text-4xl text-l font-bold text-black-500'>Create An Account Today!</h1>
    <Link to='/createaccount' className='text-red-500'>
    <button className='bg-red-500 w-[200px] rounded-full font-medium my-6 mx-auto py-3 text-white'>Create Account</button>
    </Link>
    </div></div></div>
    }

  </div>
)
```

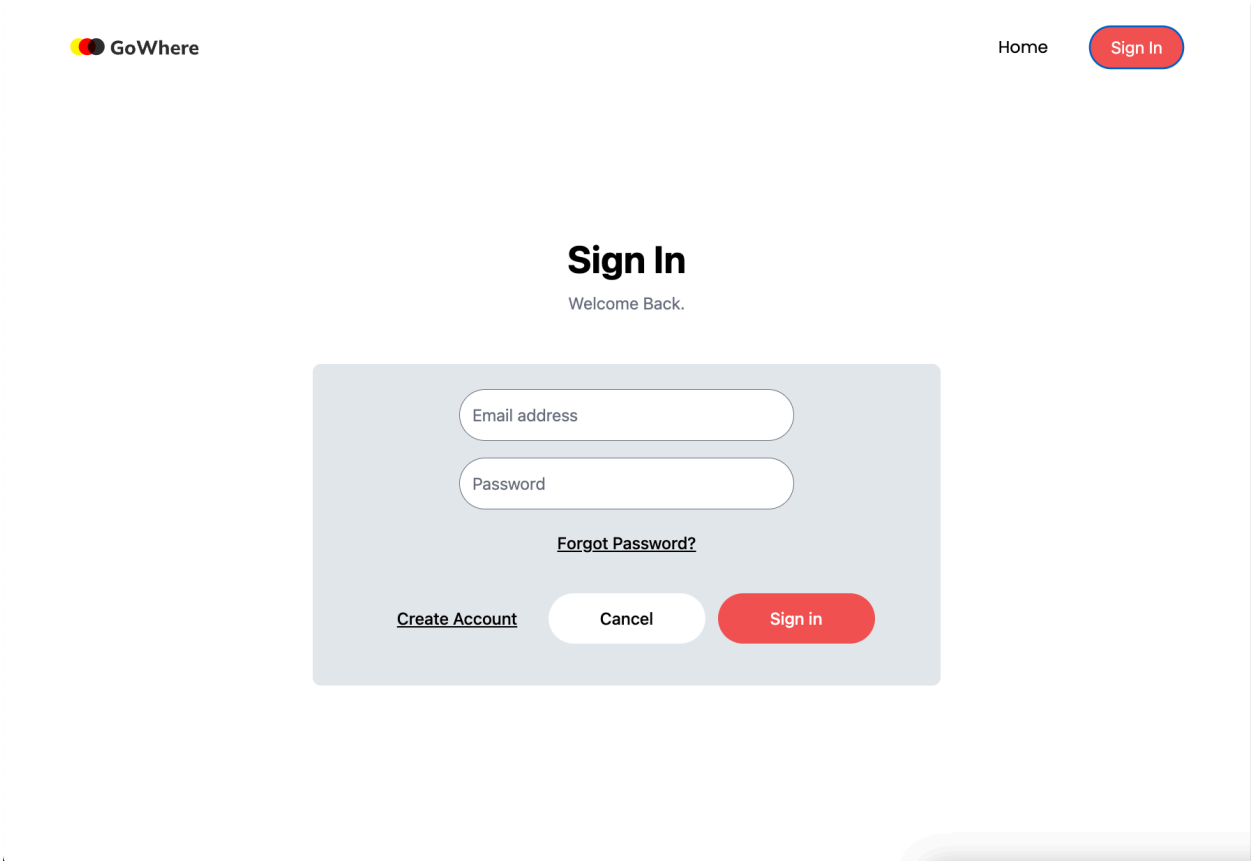
9.2 Create Account Pag



```
import { createUserWithEmailAndPassword } from 'firebase/auth';
import React, { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { auth } from '../firebase/firebase';

export const CreateAccount = () => {
    const navigate = useNavigate();

    const [email, setEmail] = useState('')
    const [password, setPassword] = useState('');
    const [confirmPassword, setConfirmPassword] = useState('');

    const onSubmit = async (e) => {
        e.preventDefault()

        if (password !== confirmPassword) {
            alert('Passwords do not match. Please try again.');
            return;  // Exit the function if passwords don't match
        }

        await createUserWithEmailAndPassword(auth, email, password)
          .then((userCredential) => {
              // Signed in
              const user = userCredential.user;
              console.log(user);
              navigate("/login")
              // ...
          })
          .catch((error) => {
              const errorCode = error.code;
              const errorMessage = error.message;
              if(errorCode === 'auth/weak-password'){
                alert('The password is too weak, please key in at least 6 characters.');
              }
              else if(errorCode === 'auth/email-already-in-use'){
                alert('The email is already in use, please key in a different email.');
              }
              else if(errorCode === 'auth/invalid-email'){
                alert('The email is invalid, please key in a valid email.');
              }
              else if(errorCode === 'auth/missing-password'){
                alert('Please key in a password.');
              }
              else {
                alert(errorCode);
              }

              // ..
          });
    }

    const handleKeyDown = (event) => {  // Function to handle the Enter key
        if (event.key === 'Enter') {
            onSubmit(event);
        }
    }
```

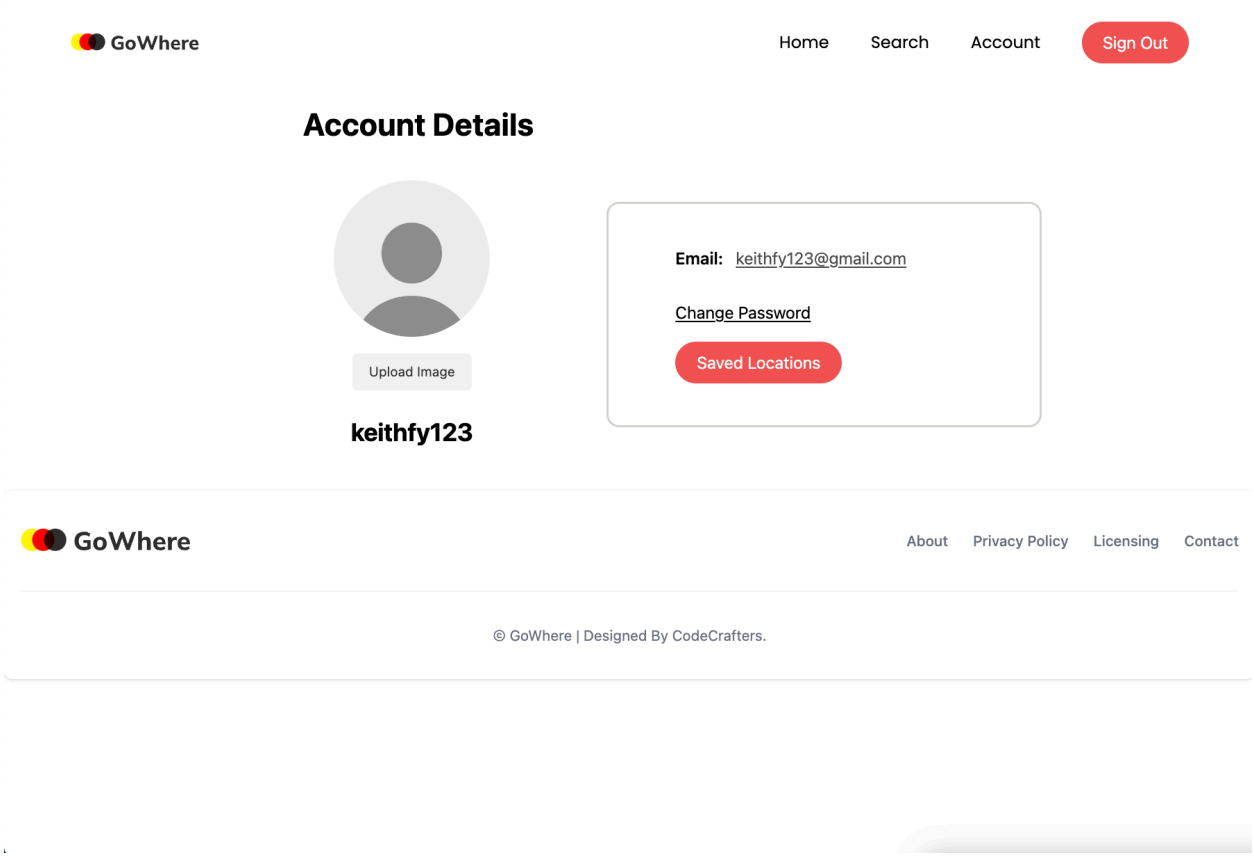
```
export const CreateAccount = () => {

    return (
    <div className='max-w-[600px] w-full h-screen mx-auto text-center flex flex-col justify-center'>
        <div className='grid grid-cols-1 grid-row'>
            <div className='mt-[-120px]'>
                <h1 className='md:text-4xl text-l font-bold text-black-500'>
                    Create Account
                </h1>
                <p className='md:text-m text-l font-sm text-gray-500 mt-[10px]'>
                    Create an Account Today.
                </p>
            </div>

            <div className='bg-gray-200 rounded-lg py-4'>

                <form className='flex flex-col items-center'>
                    <input
                        type="email"
                                label="Email address"
                                value={email}
                                onChange={(e) => setEmail(e.target.value)}
                                required
                                placeholder="Email address"
                        className='p-3 w-80 my-2 rounded-full'/>

                    <input
                        type="password"
                                label="Create password"
                                value={password}
                                onChange={(e) => setPassword(e.target.value)}
                                required
                                placeholder="Password"
                        className='p-3 w-80 my-2 rounded-full'/>

                    <input
                        type='password'
                        placeholder='Confirm Password'
                        value={confirmPassword}
                        onChange={(e) => setConfirmPassword(e.target.value)}
                        required
                        onKeyDown={handleKeyDown}
                        className='p-3 w-80 my-2 rounded-full'/>

                    <div className='flex'>
                        <Link to='/login'><button className='bg-white-200 w-[150px] rounded-full font-medium my-6 mx-auto py-3 text-black'><u>Sign in</u></button></Link>
                        <Link to='/'><button className='bg-white w-[150px] mx-3 rounded-full font-medium my-6 py-3 text-black'>Cancel</button></Link>
                        <button
                            type="submit"
                            onClick={onSubmit}
                            className='bg-red-500 w-[200px] rounded-full font-medium my-6 mx-auto py-3 text-white'>
                                Create Account
                        </button>
                    </div>
                </form>

            </div>



        </div>
    </div>
  )
}
```

9.3 Login Page



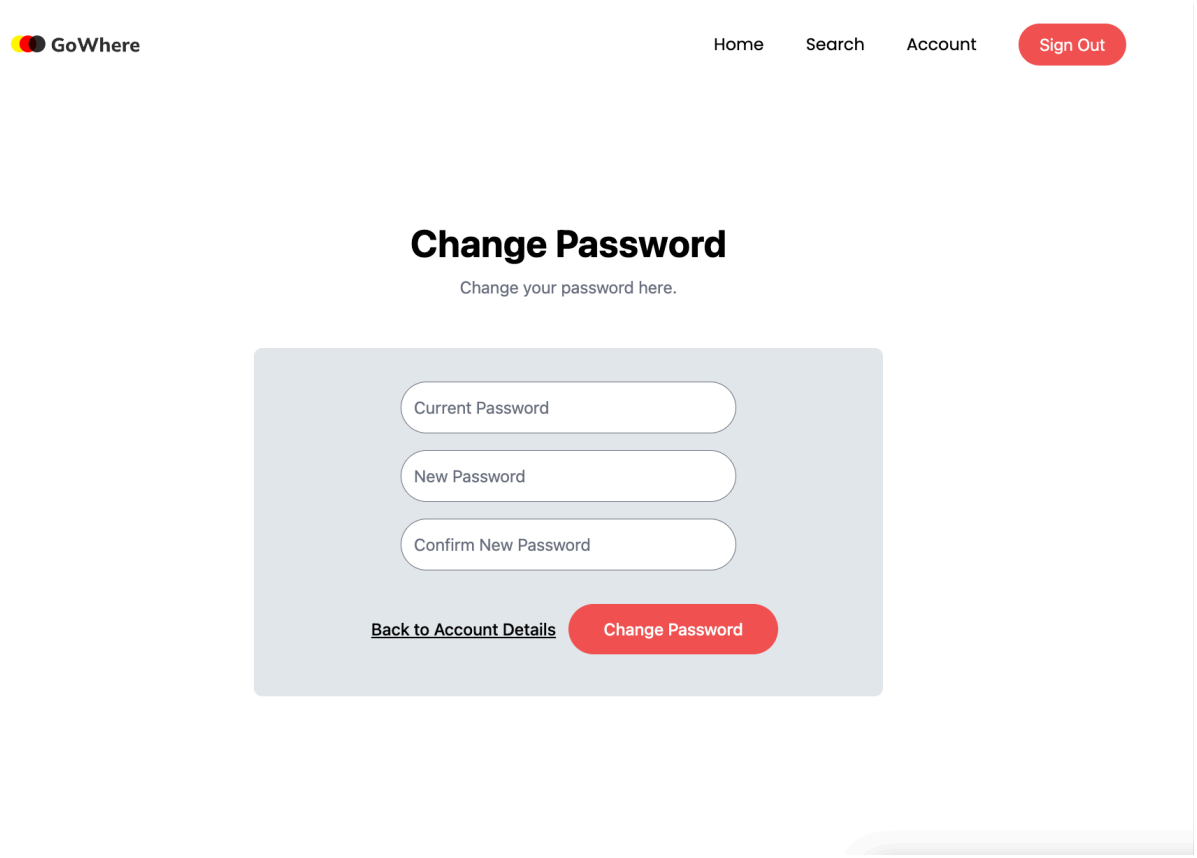
```jsx
export const Login = () => {
  return (
    <div className='max-w-[600px] w-full h-screen mx-auto text-center flex flex-col justify-center'>
        <div className='grid grid-cols-1 grid-row'>
            <div className='mt-[-120px]'>
                <h1 className='md:text-4xl text-l font-bold text-black-500'>
                    Sign In
                </h1>
                <p className='md:text-m text-l font-sm text-gray-500 mt-[10px]'>
                    Welcome Back.
                </p>
            </div>

            <div className='bg-gray-200 rounded-lg py-4'>
                <form className='flex flex-col items-center'>
                    <input
                        id="email-address"
                        name="email"
                        type="email"
                        required
                        placeholder="Email address"
                        onChange={(e)=>setEmail(e.target.value)}
                        onKeyDown={handleKeyDown}
                    className='p-3 w-80 my-2 rounded-full'/>


                    <input
                        id="password"
                        name="password"
                        type="password"
                        required
                        placeholder="Password"
                        onChange={(e)=>setPassword(e.target.value)}
                        onKeyDown={handleKeyDown}
                    className='p-3 w-80 my-2 rounded-full'/>



                    <div className='flex'>
                        <Link to='/forgotpassword'><button className='bg-white-200 w-[150px] rounded-full font-medium mx-auto py-3 text-black'><u>Forgot Password?</u></button></Li

                    </div>

                    <div className='flex'>
                        <Link to='/createaccount'><button className='bg-white-200 w-[150px] rounded-full font-medium my-6 mx-auto py-3 text-black'><u>Create Account</u></button></
                        <Link to='/'><button className='bg-white mx-3 w-[150px] rounded-full font-medium my-6 py-3 text-black'>Cancel</button></Link>
                        <button onClick={onLogin} className='bg-red-500 w-[150px] rounded-full font-medium my-6 mx-auto py-3 text-white'>
                            Sign in
                        </button>
                    </div>
                </form>
            </div>

        </div>
    </div>
  )
}
```

9.4 Account Details Page



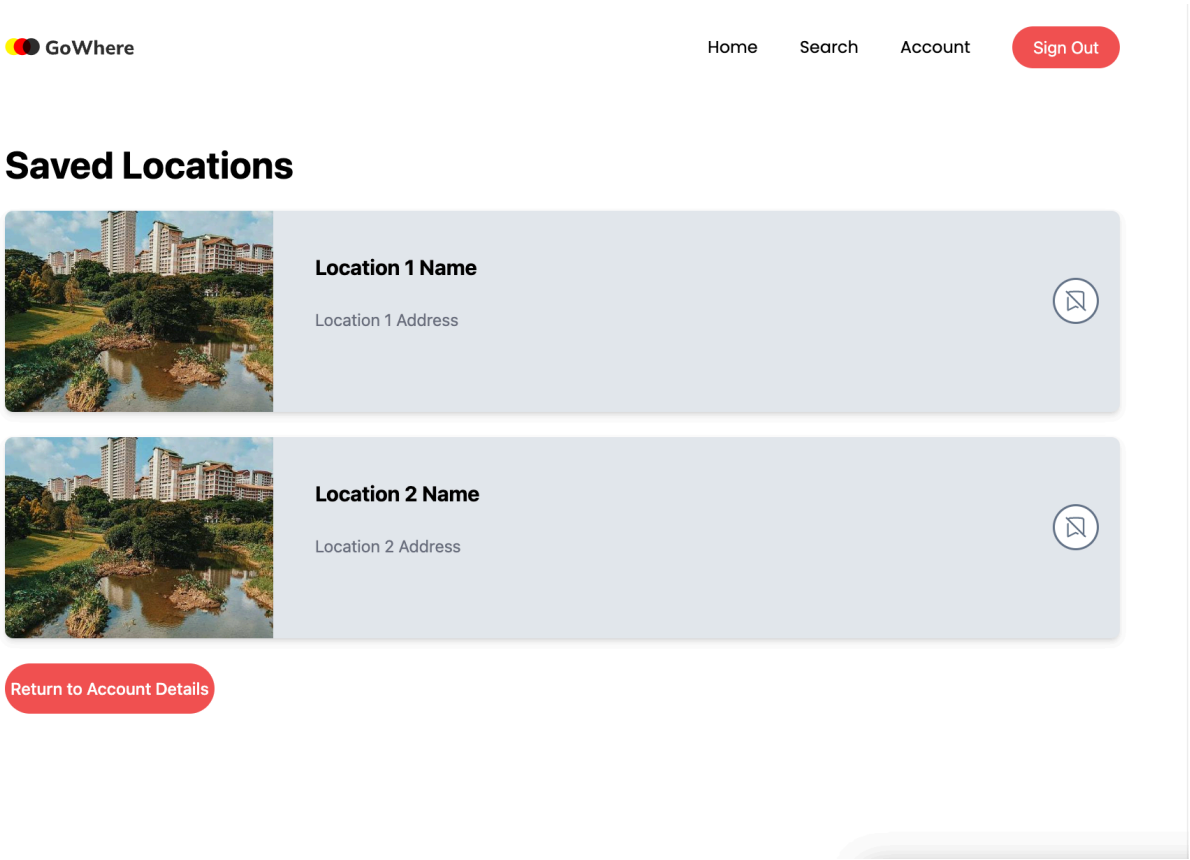
```jsx
export const AccountDetails = () => {

  return (
    <div className='flex-col my-4'>
      <div className="flex flex-col w-2/5 items-end ml-7">
        <h1 className='md:text-3xl text-l font-bold text-black-500'>Account Details</h1>
      </div>
      <div className="pt-5 flex sm:flex-row wrap my-4">
        <div className="w-2/5 flex flex-row justify-end">
          <div className="flex flex-col justify-end items-center">
            {imageUrl ? <img src={imageUrl} alt="Uploaded Image" /> : <img src={ProfileAvatar} alt="Profile Picture" className='w-[150px] h-[1
            <label id='uploadimage' for="imageInput" >Upload Image</label>
            <input
              type="file"
              id="imageInput"
              onChange={handleImageChange}
            />
            <h2 className='md:text-2xl text-l font-bold text-black-500 pt-2'>{username}</h2>
          </div>
        </div>
        <div className="w-3/5 flex flex-col justify-center items-start pl-24 mx-4">
          <div className="flex flex-col border-2 border-stone-300 rounded-xl pl-16 pr-32 py-10">
            <div className="grid gap-x-3 gap-y-3" style={{ gridTemplateColumns: 'repeat(2, minmax(0, max-content))' }}>
              <div className='font-semibold'>Email:</div>
              <div className='underline underline-offset-2 opacity-70'>{email}</div>
              <div className='col-span-2 mt-4'>
                <Link to='/changepassword'><button className='underline underline-offset-2'>Change Password</button></Link>
              </div>
            </div>
            <div className='mt-4'>
              <Link to='/savedlocations'><button className='bg-red-500 w-[160px] py-2 rounded-full text-white'>Saved Locations</button></Link>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};
```

9.5 Change Password Page



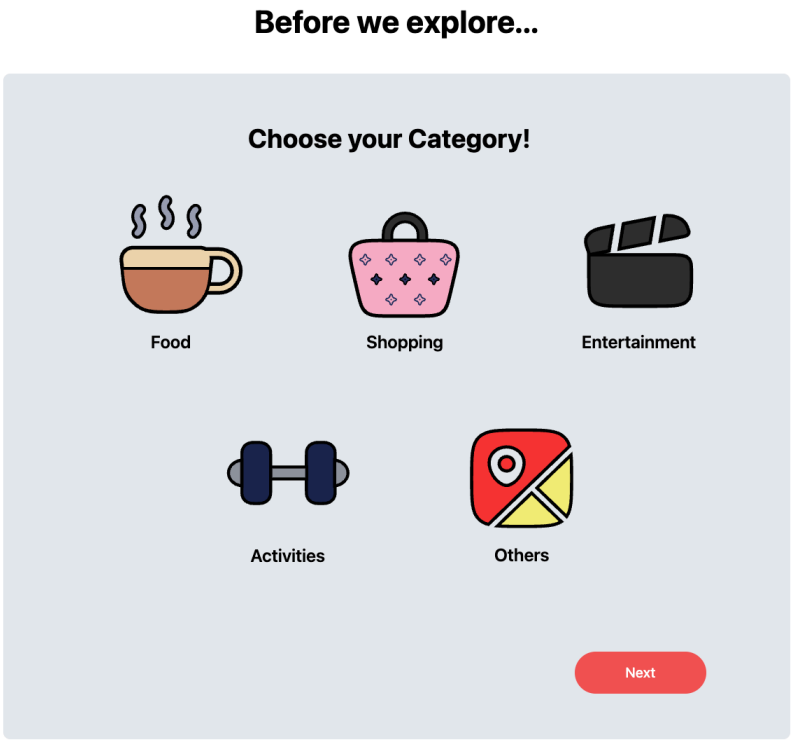
```
export const ChangePassword = () => {

  return (
    <div className='max-w-[600px] w-full h-screen mx-auto text-center flex flex-col justify-center'>
        <div className='grid grid-cols-1 grid-row'>
            <div className='mt-[-120px]'>
                <h1 className='md:text-4xl text-l font-bold text-black-500'>
                    Change Password
                </h1>
                <p className='md:text-m text-l font-sm text-gray-500 mt-[10px]'>
                    Change your password here.
                </p>
            </div>
            <div className='bg-gray-200 rounded-lg py-4'>
                    <form className='flex flex-col items-center' onSubmit={handleSubmit}>
                    {error && <p className='text-red-500 font-bold'>{error}</p>}
                    {success && <p className='text-green-500 font-bold'>{success}</p>}
                    <input
                        type='password'
                        name='currentPassword'
                        placeholder='Current Password'
                        value={currentPassword}
                        onChange={handleInputChange}
                        className='p-3 w-80 my-2 mt-4 rounded-full'
                        required/>
                    <input
                        type='password'
                        name='newPassword'
                        placeholder='New Password'
                        value={newPassword}
                        onChange={handleInputChange}
                        className='p-3 w-80 my-2 rounded-full'
                        required/>
                    <input
                        type='password'
                        name='confirmPassword'
                        placeholder='Confirm New Password'
                        value={confirmPassword}
                        onChange={handleInputChange}
                        className='p-3 w-80 my-2 rounded-full'
                        onKeyDown={(e) => e.key === 'Enter' && handleSubmit(e)}
                        required/>
                    <div className='flex'>
                        <Link to='/accountdetails'><button className='bg-white-200 mx-3 rounded-full font-medium my-6 mx-3 py-3 text-black'><u
                        <button type='submit' className='bg-red-500 w-[200px] rounded-full font-medium my-6 mx-auto py-3 text-white'>Change Pa
                    </div>
                </form>
            </div>
        </div>
    </div>
  )
}
```

9.6 Saved Locations Page



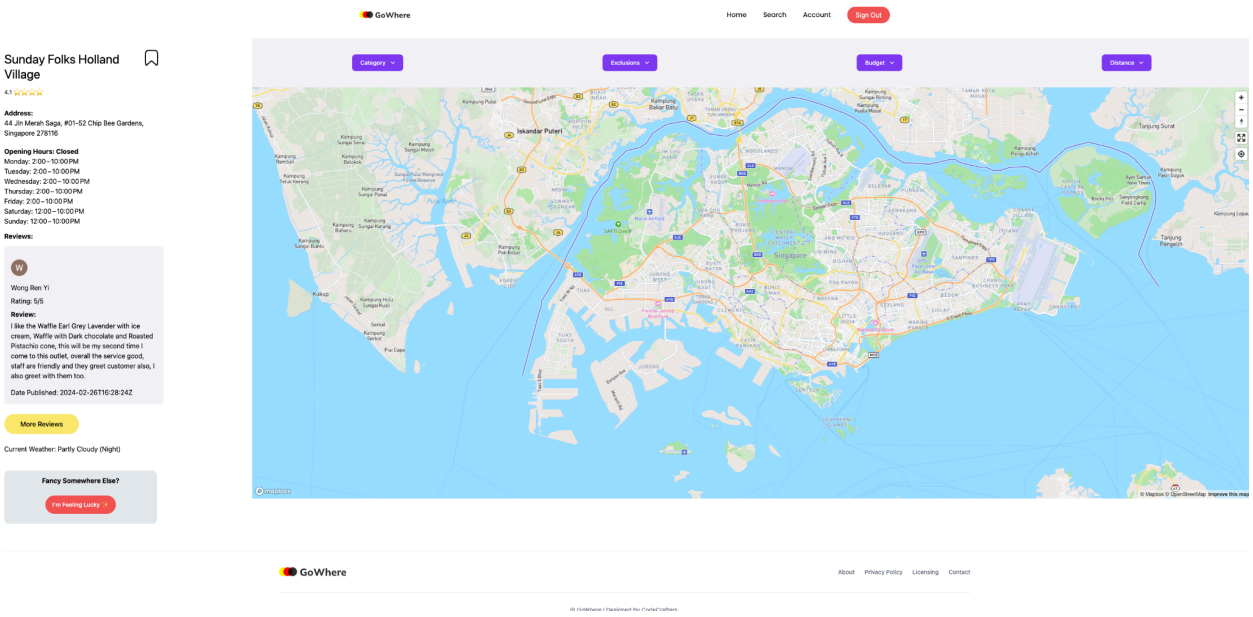
```jsx
import React from 'react'
import { Link } from 'react-router-dom'

export const SavedLocations = () => {

  return (
    <div className='px-16 py-6 w-full h-screen '>
      <h1 className='py-6 md:text-4xl font-bold text-black-500'>
        Saved Locations
      </h1>
      <div className='bg-gray-200 rounded-lg overflow-hidden shadow-md relative flex'>
        <img src='img/testimg.jpg' class='w-64 h-32 sm:h-48 object-cover'></img>
        <div class = 'py-6 m-4 justify-center'>
            <span class = 'px-6 py-6 font-bold text-xl'>Location 1 Name</span>
            <span class = 'px-6 py-6 block text-gray-500'>Location 1 Address</span>
        </div>

          <button class='border-slate-500 border-2 bg-white text-slate-500 text-xs uppercase font-bold rounded-full p-2 absolute top-14 right-
                    <svg class='w-5 inline-block' xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke='
              <path strokeLinecap="round" strokeLinejoin="round" d="m3 3 1.664 1.664M21 21l-1.5-1.5m-5.485-1.242L12 17.25 4.5 21V8.742m.164-4.
            </svg>
          </button>
      </div>
      <div className='py-3'></div>
      <div className='bg-gray-200 rounded-lg overflow-hidden shadow-md relative flex'>
        <img src='img/testimg.jpg' class='w-64 h-32 sm:h-48 object-cover'></img>
        <div class = 'py-6 m-4 justify-center'>
            <span class = 'px-6 py-6 font-bold text-xl'>Location 2 Name</span>
            <span class = 'px-6 py-6 block text-gray-500'>Location 2 Address</span>
        </div>

          <button class='border-slate-500 border-2 bg-white text-slate-500 text-xs uppercase font-bold rounded-full p-2 absolute top-14 right-
                    <svg class='w-5 inline-block' xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke='
              <path strokeLinecap="round" strokeLinejoin="round" d="m3 3 1.664 1.664M21 21l-1.5-1.5m-5.485-1.242L12 17.25 4.5 21V8.742m.164-4.
            </svg>
          </button>
      </div>

      <div className='next'>
        <Link to='/accountdetails'><button className='bg-red-500 w-[200px] rounded-full font-medium my-6 mx-auto py-3 text-white'>Return to Ac
      </div>
    </div>
  )
}

```

9.7 Questionnaire Page




```
42    return (
43      <div className='mt-36 max-w-[900px] w-full h-screen mx-auto text-center flex flex-col justify-center'>
44          <div className='grid grid-cols-1 grid-row'>
45              <div className='mt-[-80px]'>
46                  <h1 className='md:text-4xl text-l font-bold text-black-500'>
47                      Before we explore...
48                  </h1>
49              </div>
50          </div>
51          <div className='bg-gray-200 rounded-lg py-4'>
52              <form>
53                  <div>
54                    {handleStep()}
55                  </div>
56                  <br />
57                  <div className="flex justify-end my-3 mx-24">
58                    <div className="mx-3">{step!=1 && <button onClick = {back} className='bg-white mx-3 w-[150px] rounded-full font-medium my-6
59                    <div>{step!=4 ? <button onClick = {next} className='bg-red-500 w-[150px] rounded-full font-medium my-6 mx-auto py-3 text-whi
60                  </div>
61              </form>
62          </div>
63
64
65        {/* <div className='next'>
66          <Link to='/map'><button className='bg-red-500 w-[150px] rounded-full font-medium my-6 mx-auto py-3 text-white'>Go to Map</button></Lin
67        </div> */}
68      </div>
69
70    )
71  }
```

9.8 Map Page


GoWhere
Home
Search
Account
Sign Out
Category
Exclusions
Budget
Distance
Sunday Folks Holland Village
4.1
Address:
44 Jln Merah Saga, #01-52 Chip Bee Gardens, Singapore 278116
Opening Hours: Closed
Monday: 2:00 – 10:00 PM
Tuesday: 2:00 – 10:00 PM
Wednesday: 2:00 – 10:00 PM
Thursday: 2:00 – 10:00 PM
Friday: 2:00 – 10:00 PM
Saturday: 12:00 – 10:00 PM
Sunday: 12:00 – 10:00 PM
Reviews:
W
Wong Ren Yi
Rating: 5/5
Review:
I like the Waffle Earl Grey Lavender with ice cream, Waffle with Dark chocolate and Roasted Pistachio cone, this will be my second time I come to this outlet, overall the service good, staff are friendly and they greet customer also, I also greet with them too.
Date Published: 2024-02-26T16:28:24Z
More Reviews
Current Weather: Partly Cloudy (Night)
Fancy Somewhere Else?
I'm Feeling Lucky
Iskandar Puteri
Singapore
GoWhere
About
Privacy Policy
Licensing
Contact

```
export const MapPage = () => {

    return (
    <div className='grid grid-rows-6 grid-columns-10 grid-flow-col py-3'>
        <div className='row-span-6 pl-4 pt-4'>
            <div>
                <div className='flex flex-row place-content-between'>
                    <h1 className='md:text-3xl text-black-500 py-4 justify-content align-items'>{name}</h1>
                    <button className='p-2 size-16'><img src={bookmarkicon}></img></button>
                </div>

                <h2 className='md:text-sm text-black-500 pb-6'>{rating} {stars}</h2>
                <h2 className='md:text-md font-bold text-black-500 pt-4'>Address:</h2>
                <h2 className='md:text-md text-black-500 pb-10'>{address}</h2>

                {/* <h2 className='md:text-md font-bold text-black-500 py-4'>Opening Hours:</h2>
                <h2 className='md:text-md text-black-500 pb-10'>{}</h2> */}
                <div>
                    <h2 className='md:text-md text-l font-bold text-black-500 pb-3'>
                    Reviews:
                    </h2>
                    <ReviewCard />
                </div>
                <Link to='/reviews'>
                    <button className='bg-gray-200 w-[180px] rounded-full font-medium my-6 py-3 text-black'>
                        More Reviews
                    </button>
                </Link>
                <h2 className='md:text-md text-l text-black-500 pb-3'>Current Weather: {weather}</h2>
                <h2 className='md:text-md text-l text-black-500 pb-3'>{reminder}</h2>
            </div>
            <div className='text-center my-4 bg-gray-200 rounded-xl mr-4 py-3'>
                <h2 className='md:text-md text-l text-black-500 pb-3'>Somewhere Else?</h2>
                <button onClick={randomiseSearch} className='bg-red-500 w-[200px] rounded-full font-medium my-3 py-3 text-white'>
                    I'm Feeling Lucky ✨
                </button>
            </div>
        </div>
        <div className='col-span-9 bg-gray-100'>
            <div className='flex flex-row place-content-around'>
                <div>Category</div>
                <div>Exclusions</div>
                <div>Budget</div>
                <div>Distance</div>
            </div>
        </div>
        <div className='row-span-5 col-span-9'><Map /></div>
    </div>
  )
}✨
```

9.9 Database (Firebase.js)

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
import { getAuth } from "firebase/auth";
import { getStorage } from "firebase/storage";

const firebaseConfig = {
  apiKey: "AIzaSyBKdXOQeQCjWKo5EQZD8xHkmcoGkfjXI14",
  authDomain: "gowhere-e0d76.firebaseapp.com",
  projectId: "gowhere-e0d76",
  storageBucket: "gowhere-e0d76.appspot.com",
  messagingSenderId: "777994122660",
  appId: "1:777994122660:web:ea150eaae45acccf46fd31",
  measurementId: "G-L2F6C2G9J3"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);

// Initialize Firebase Authentication and get a reference to the service
export const auth = getAuth(app);
export default app;
export const storage = getStorage(app);
```

10. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

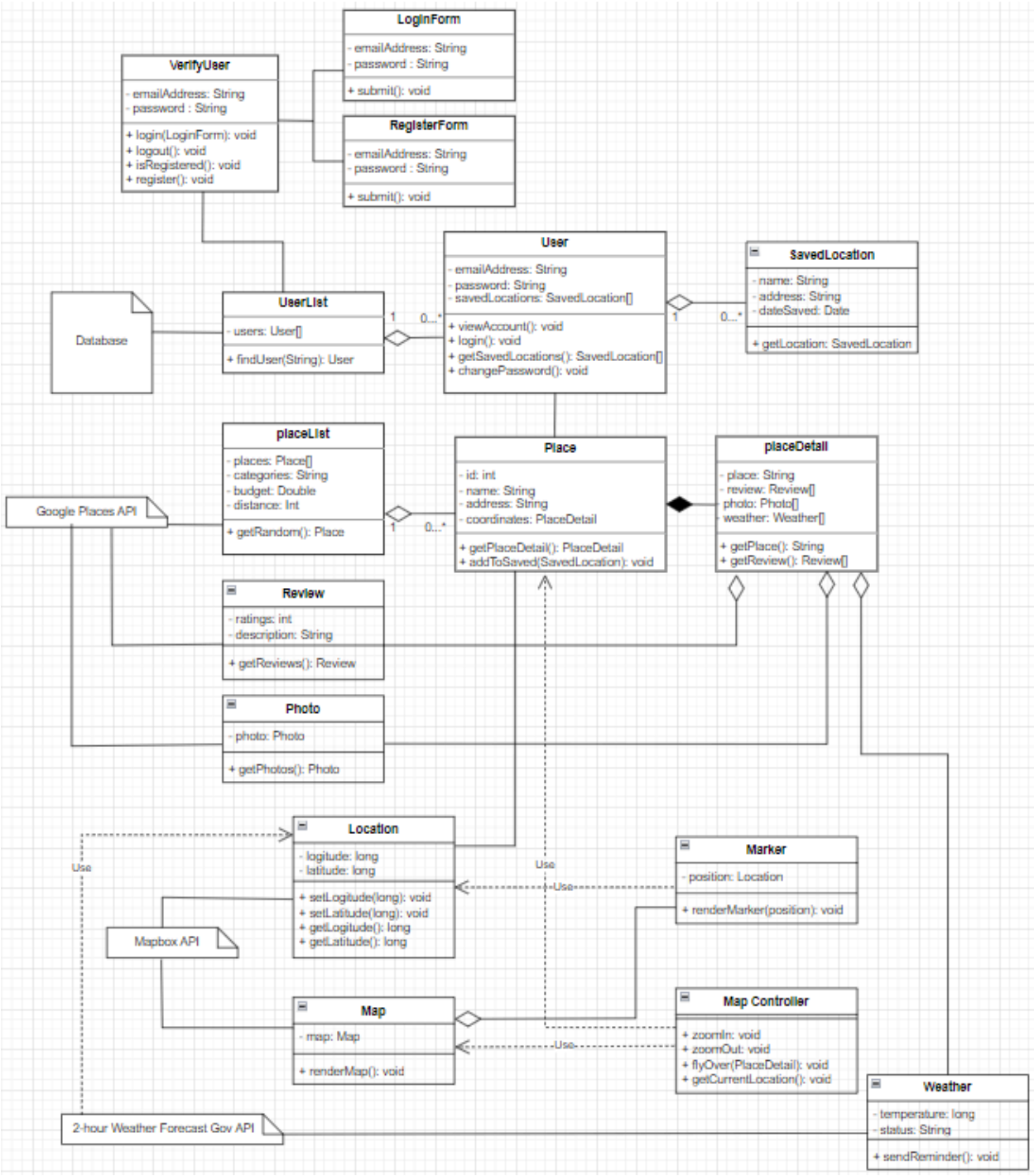
10.1 Data Dictionary

Term	Definition
Places of Interest	The places that the user often frequents.
Category	The group of places that serve similar purposes or services. E.g. restaurants, supermarkets, clothing stores, etc… Type.
Area	The circular area around the chosen location that the user is willing to travel to.
Rating	The rating of the place provided by users of google maps
Budget	The amount of money that the user is willing to spend.
User	The person who is currently using the application.
Stakeholders	The organizations or people who have a vested interest in the application.
System	The application that is being used.
Login System	The part of the application that focuses on logging in.
Food Places	Places that serve food. Eateries such as Restaurants, Cafes, Hawker Centres, Food Courts, Canteens
Requirements	The baseline necessities to be met by the application.
Homepage	The page of the application that is first viewed by the user upon logging in.

Appendix B: Analysis Models

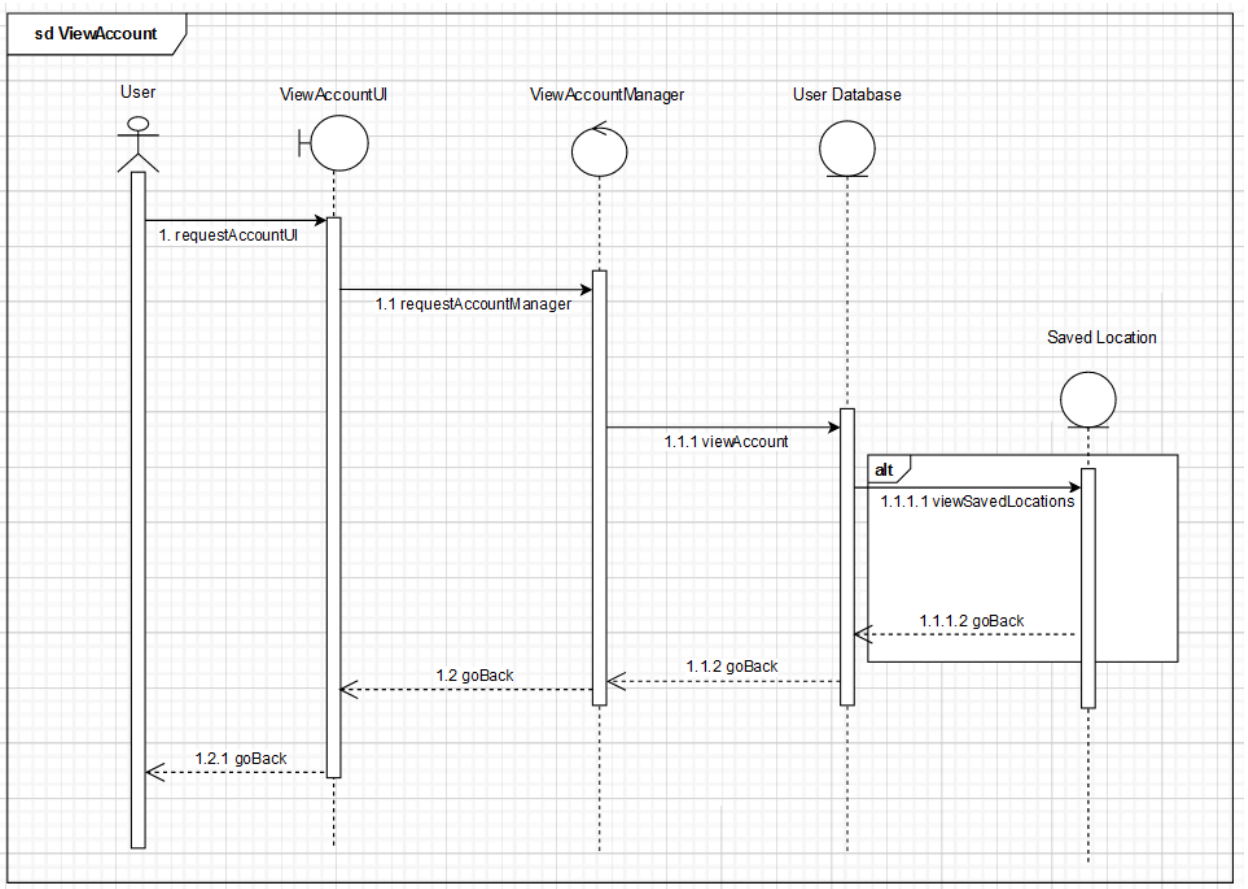
<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

10.2 Class Diagram (Conceptual Model)

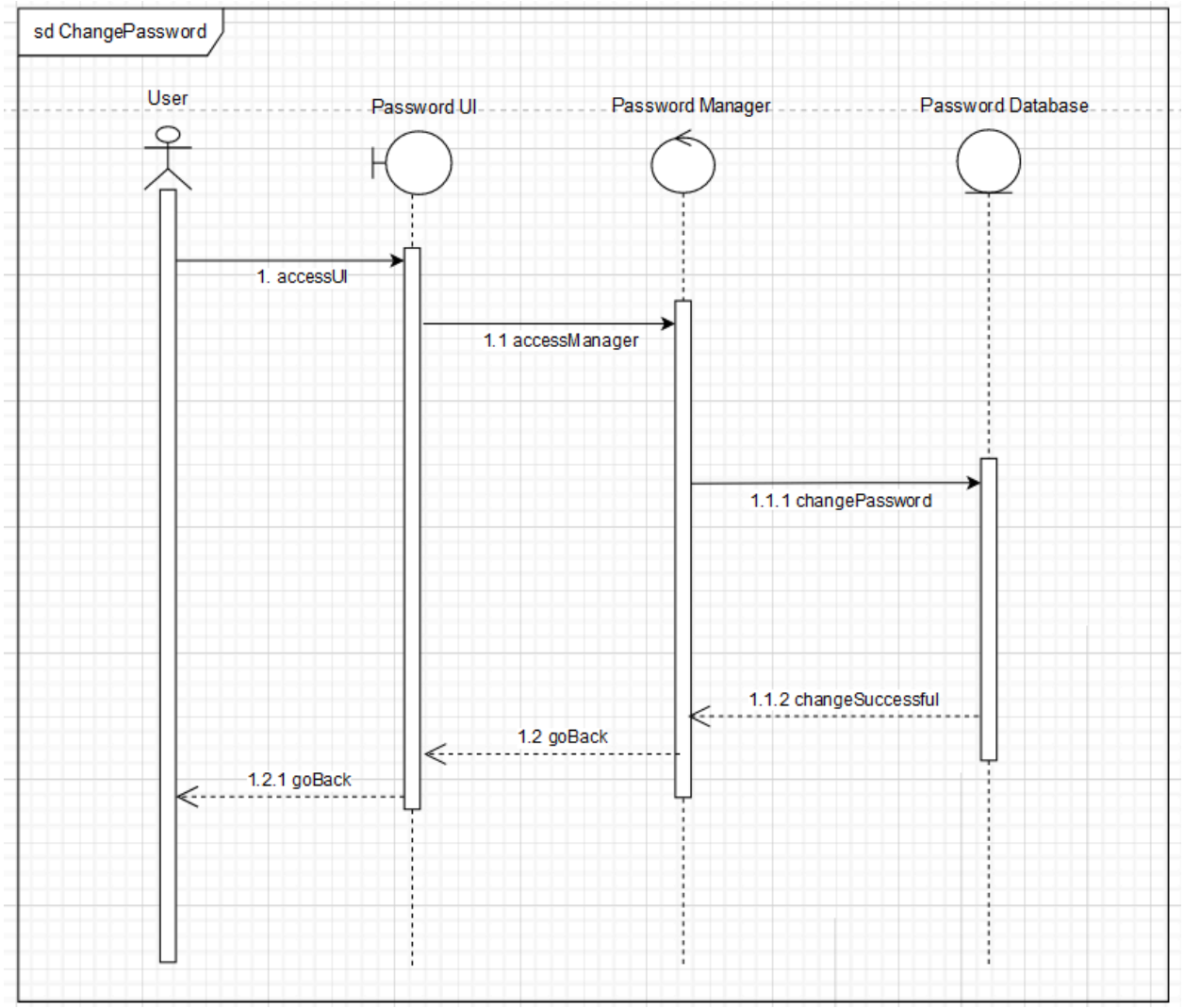


10.3 Sequence Diagram (Dynamic Model)

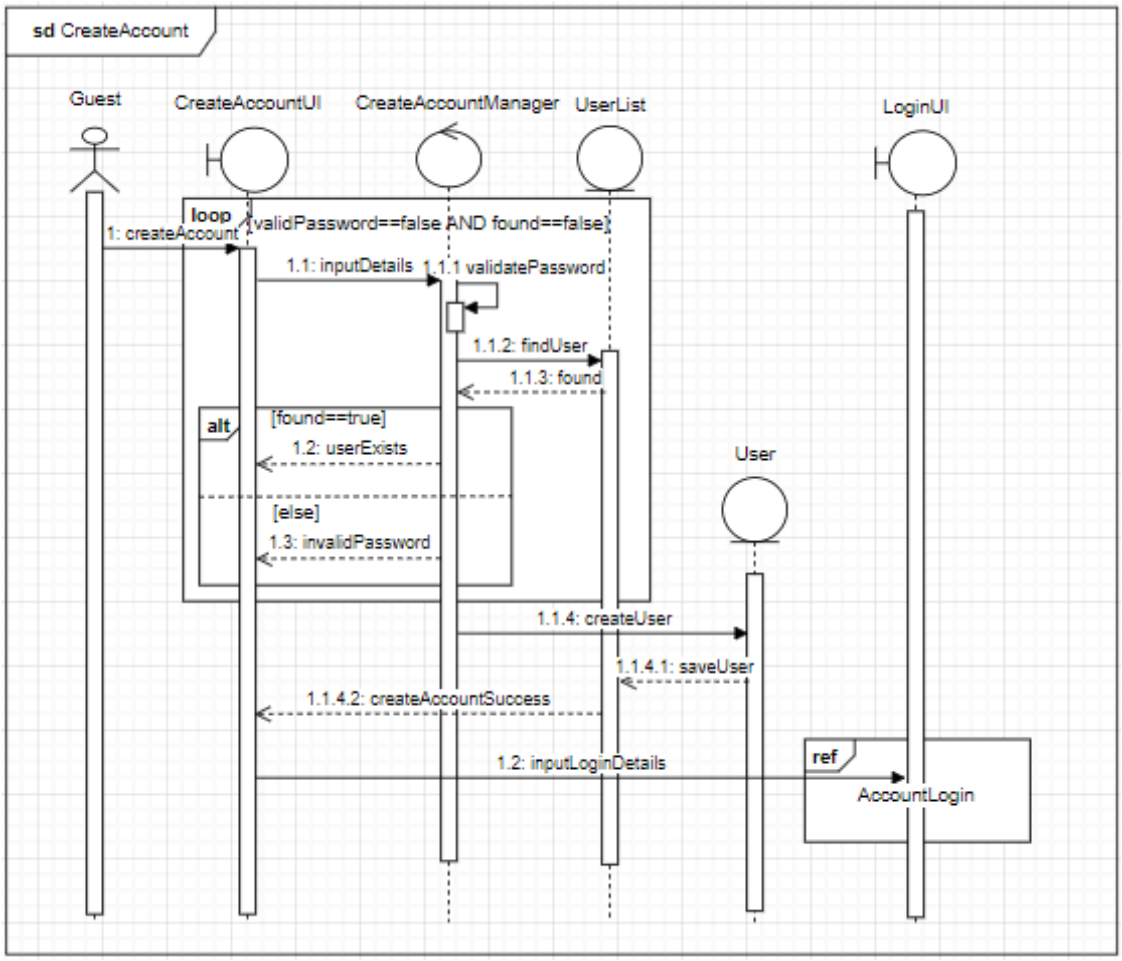
10.3.1 View account/View saved locations



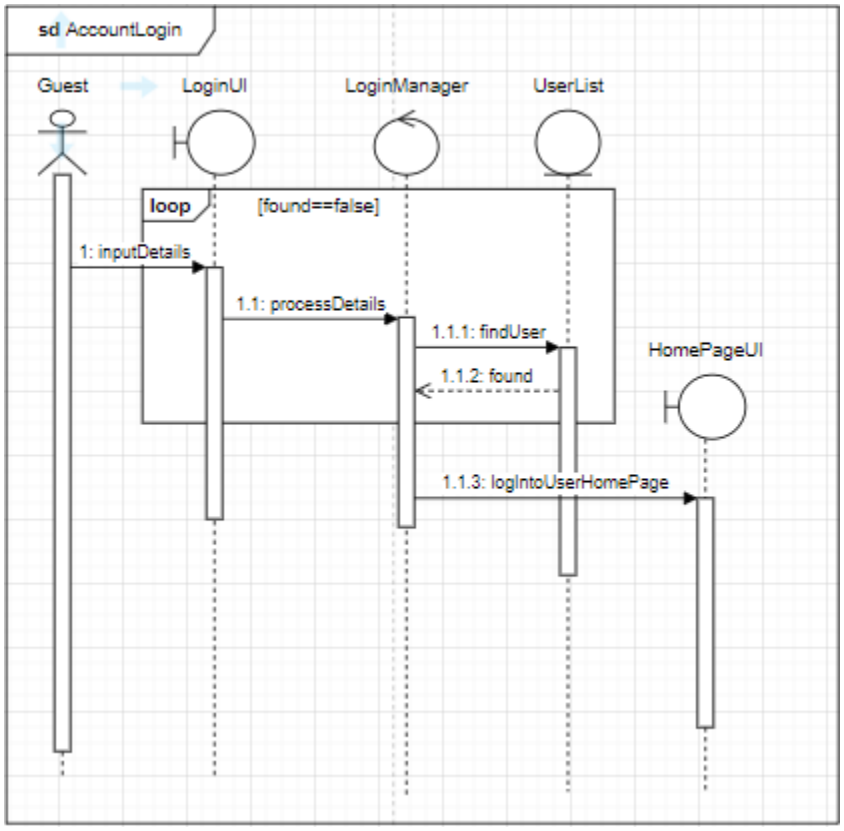
10.3.2 Change password



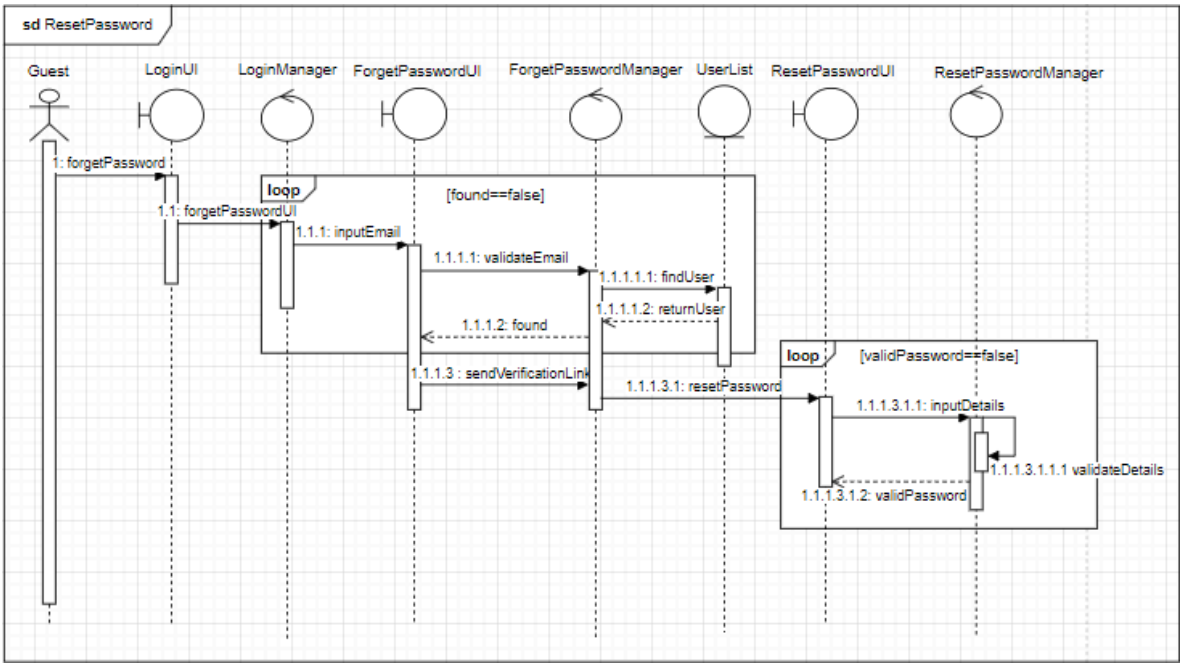
10.3.3 Create account



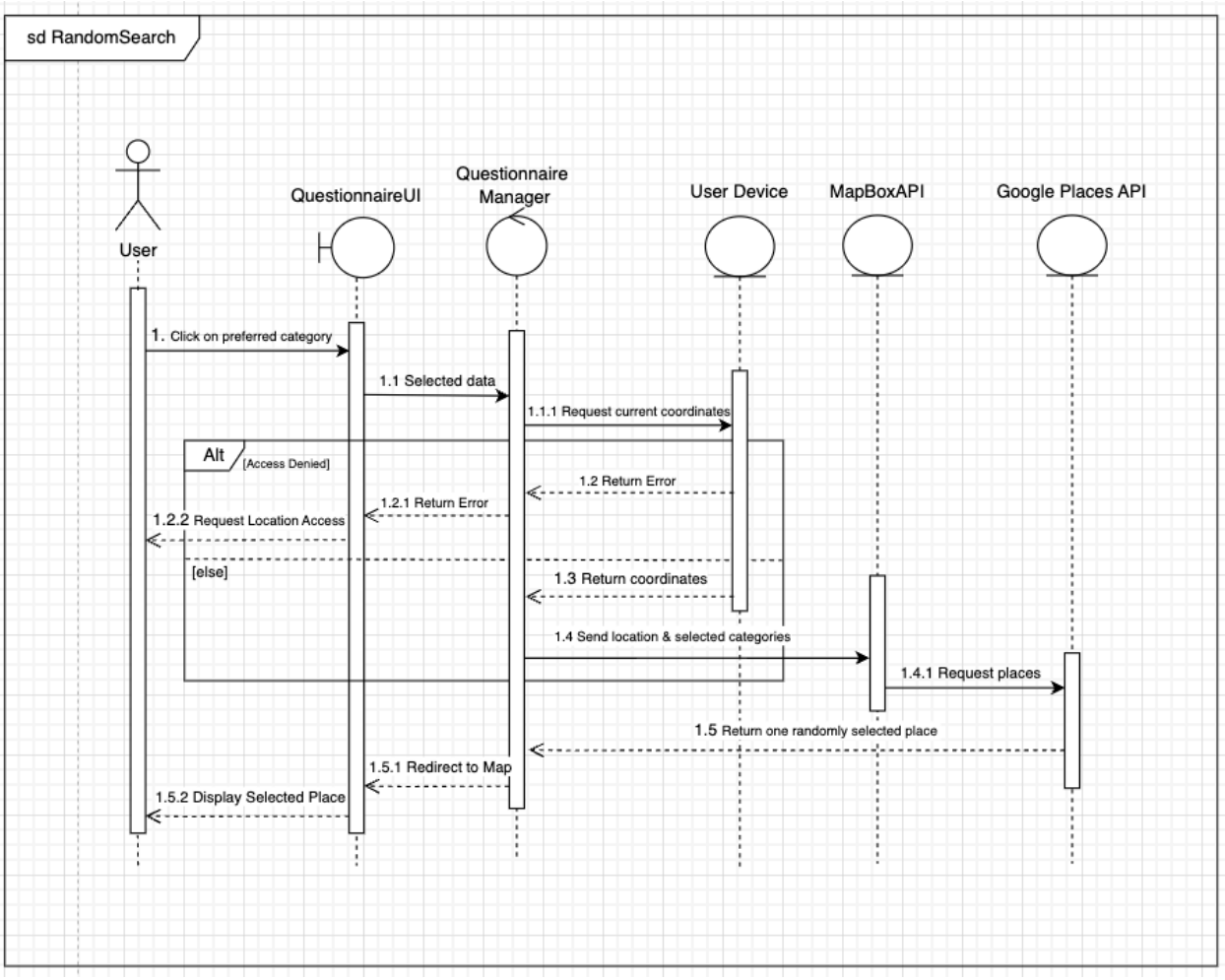
10.3.4 Account login



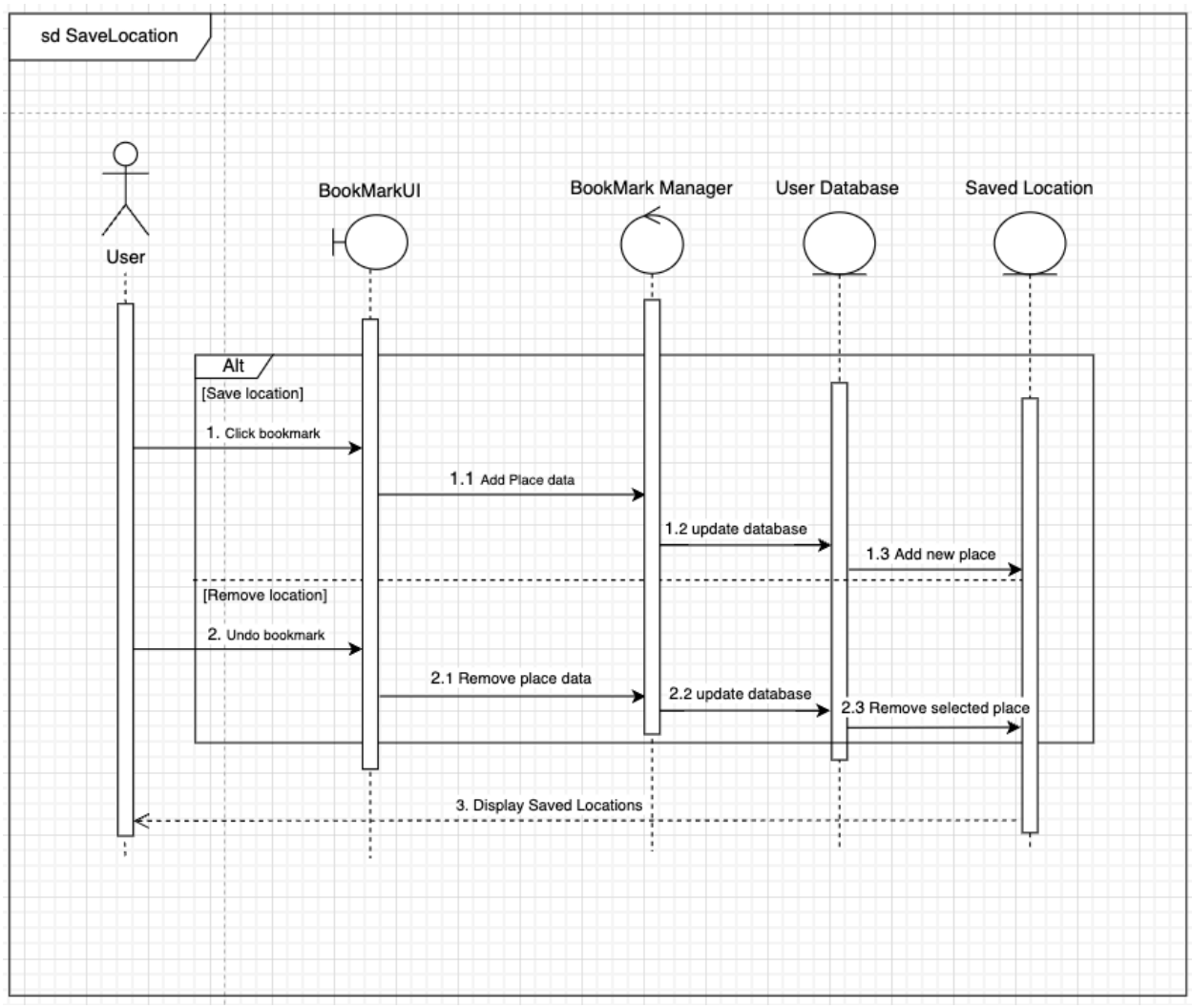
10.3.5 Reset password



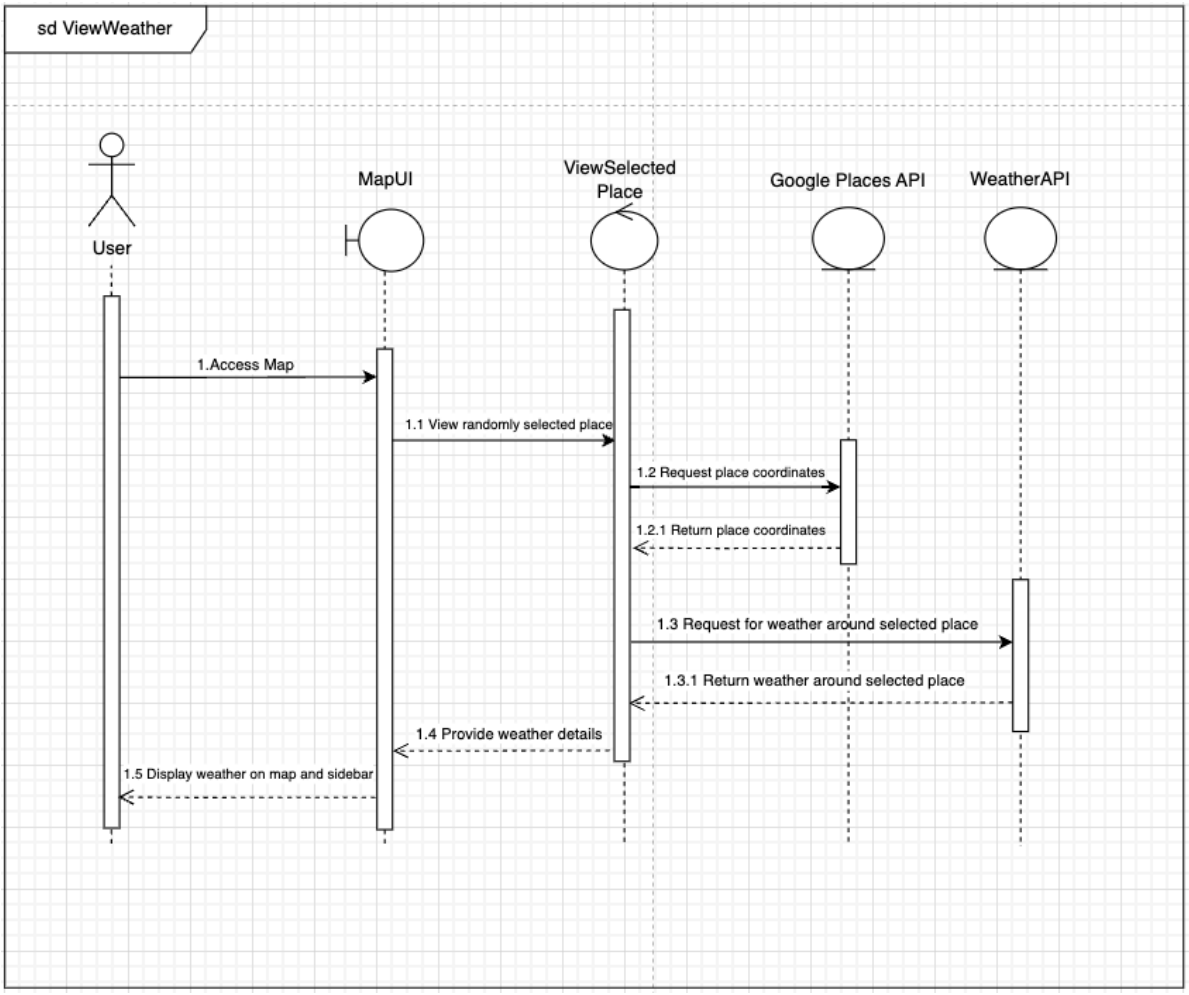
10.3.6 Random Search



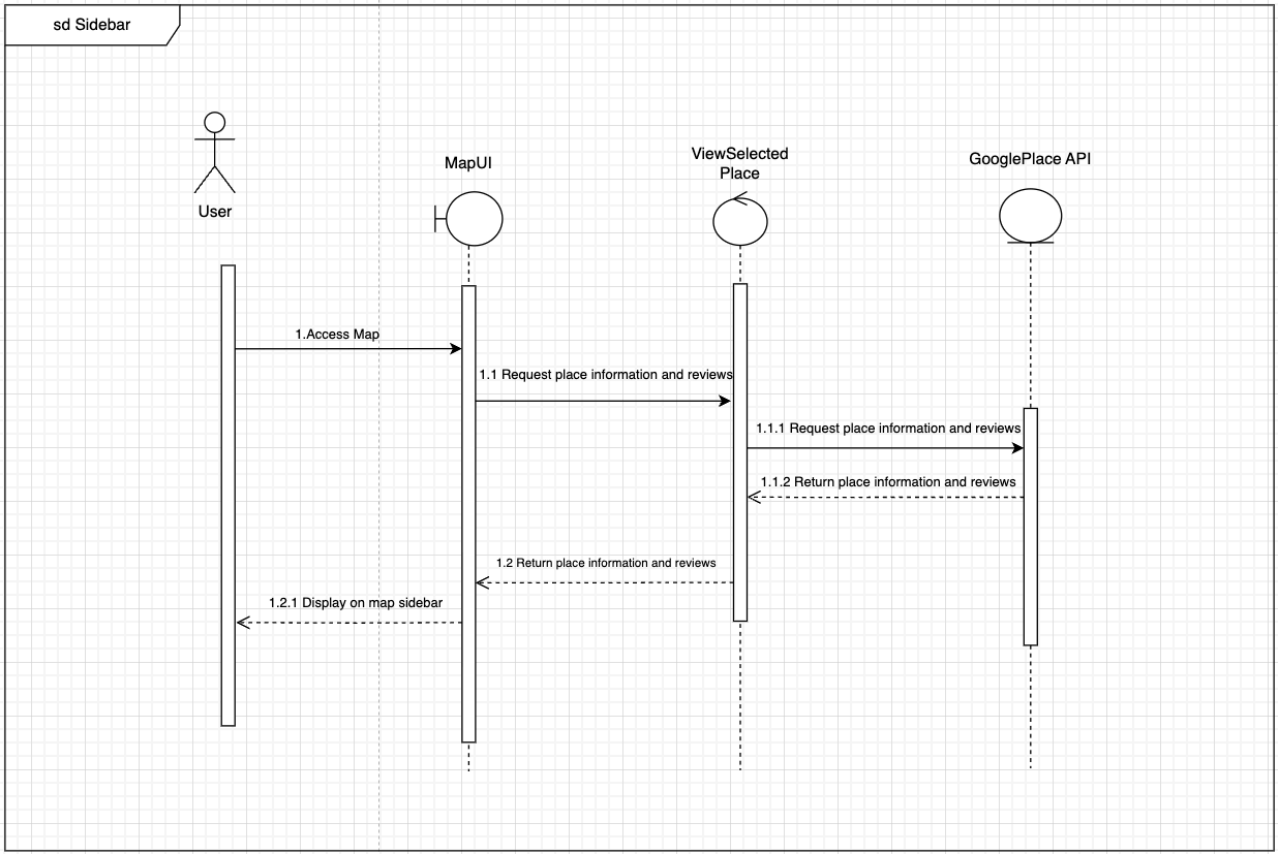
10.3.7 Save Locations



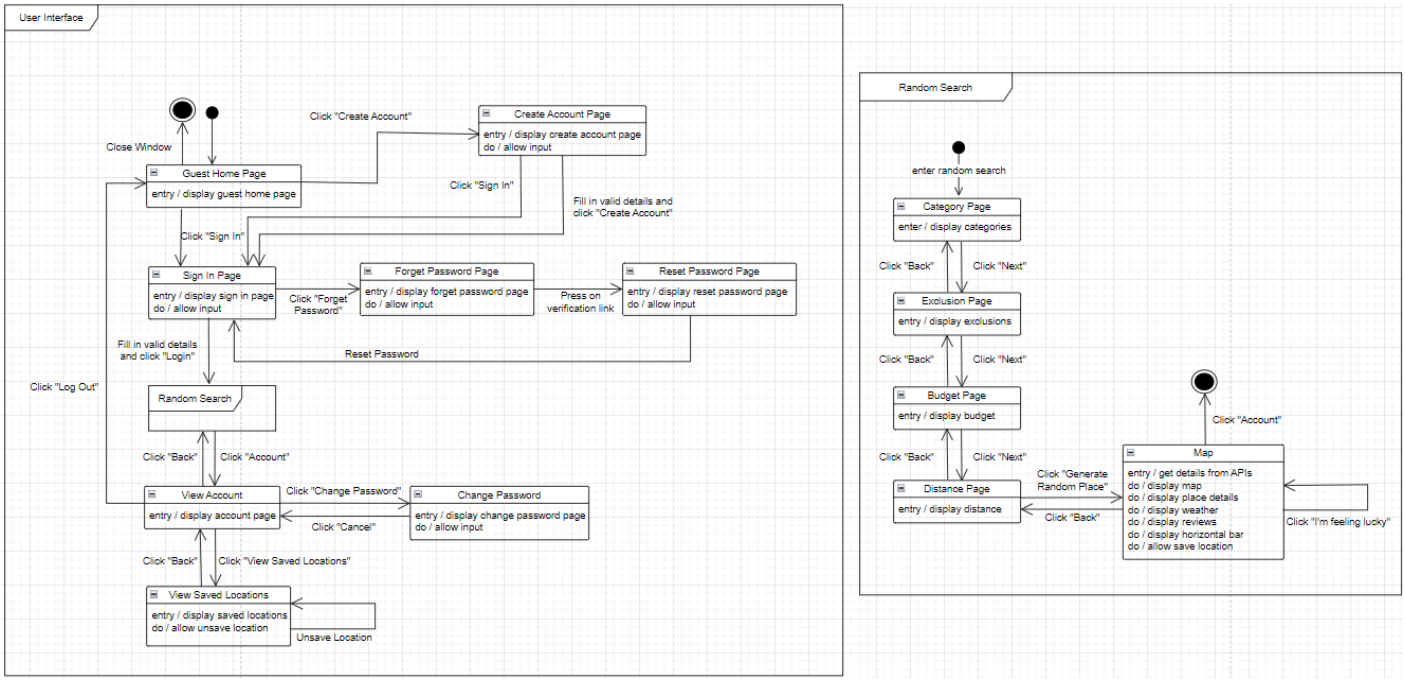
10.3.8 View Weather Information



10.3.9 View Place Information on Sidebar



10.4 Dialog Map (Dynamic Model)



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