Software Requirements Specification

for

Sweet Home Finder

Version 1.7 approved

Prepared by

TAN WU JI,
LAI BOON YI,
CHOW EN YAO,
LUYUN SEAN GABRIEL DE LA CRUZ,
OATES BENJAMIN HARRY,
ALYSSA NG YOKE SHIEN

SCSB-T2, Nanyang Technological University

20-4-2024

Table of Contents

Table of Contents ii
Revision History ii
1. Introduction 1
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience and Reading Suggestions 1
1.4 Product Scope 2
1.5 References 2
2. Overall Description 3
2.1 Product Perspective 3
2.2 Product Functions 3
2.2.1 Accounts 3
2.2.2 Logged In Functionality 3
2.2.3 Property Search 4
2.3 User Classes and Characteristics 4
2.3.1 Normal Users 4
2.3.2 Working Adults 4
2.4 Operating Environment 5
2.4.1 Production Environment 5
2.4.2 Development Environment 5
2.5 Design and Implementation Constraints 6
2.6 User Documentation 6
2.7 Assumptions and Dependencies 7
3. External Interface Requirements 7
3.1 User Interfaces 7
3.2 Hardware Interfaces 13
3.3 Software Interfaces 14
3.4 Communications Interfaces 14
4. System Features 14
4.1 User Account Registration 14
4.2 User Account Login 17
4.3 Search Home 19
4.4 Bookmark 21
4.5 TransportOption 22
4.6 ShowETA 26
4.7 FrequentAddress 25
4.8 Show nearby amenities 26
4.9 Reset Password 27
5. Other Nonfunctional Requirements 28
5.1 Performance Requirements 28
5.2 Safety Requirements 29
5.3 Security Requirements 29
5.4 Software Quality Attributes 29

5.5 Business Rules 29
6. Other Requirements 30
6.1 Legal and Compliance Requirements 30
Appendix A: Data Dictionary 31
Appendix B: Analysis Models 32
Appendix C: To Be Determined List 39

Revision History

Name	Date	Reason For Changes	Version
TAN WU JI	18/4/2024	Initial write up for introduction	1.0
TAN WU JI	19/4/2024	Initial write up for Overall Description	1.1
CHOW EN YAO	19/4/2024	Initial write up for External Interface Requirements	1.2
LUYUN SEAN	19/4/2024	Initial write up for System Features	1.3
ALYSSA NG	19/4/2024	Initial write up for Other Nonfunctional Requirements	1.4
LAI BOON YI	19/4/2024	Initial write up for Other Requirements	1.5
BENJAMIN HARRY	19/4/2024	Complete for Appendix A	1.6
LAI BOON YI	19/4/2024	Complete for Appendix B	1.7

1. Introduction

1.1 Purpose

This Software Requirement Specification (SRS) document is prepared for the *Sweet Home Finder* web application. The purpose of this SRS document is to describe the requirements specifications for application in detail and to facilitate the development process for all relevant stakeholders. This document covers all aspects of the web application, including but not limited to system features, functional and non-functional requirements, user interface design and limitations.

1.2 Document Conventions

This document follows standard typographical conventions which are listed in this section. All readers should pay attention to these standards to better understand this document when reading it.

Font:	Times New Roman
Heading:	Bold, Size 18
Sub-heading:	Bold, Size 14
Content:	Size 12
Technical Standards:	IEEE 830-1998

Please refer to Appendix A: Data Dictionary for the definition of special terms used throughout this document.

1.3 Intended Audience and Reading Suggestions

This document is prepared for all stakeholders, which includes the development team, testing team, project managers, marketing team and also the end users of *Sweet Home Finder* web application.

The development team should proceed with Section 2, *Overall Description*, to have a holistic understanding of the functionalities, design and also constraints of the web application. After that, Section 4, *System Features*, will provide developers with detailed explanations for each system feature in the web application. Lastly, Section 3, *External Interfaces Requirements*, and Section 5, *Other Non-functional Requirements*, will provide developers with an overall understanding of the requirements specified for the application to function as desired.

The testing team, project managers, marketing team and the end users of the application are encouraged to read this document in sequential order to gain a better understanding of this web application.

1.4 Product Scope

Sweet Home Finder is an innovative property finder web application poised to transform the real estate search experience in Singapore. Its primary aim is to empower users with an intuitive platform for discovering, exploring, comparing, and evaluating properties.

The application's core objective is to furnish users with comprehensive property information, enabling them to make well-informed decisions about their potential future homes. By offering a user-friendly interface, *Sweet Home Finder* seeks to simplify the property search process, thereby saving users valuable time and effort.

One of the key features of *Sweet Home Finder* is its ability to display property locations and nearby amenities simultaneously on an interactive map interface. This feature enhances the user experience by providing a holistic view of the property's surroundings, eliminating the need for users to physically visit each property during their search.

For users with specific preferences and considerations, *Sweet Home Finder* offers additional functionalities. Users can input their frequently visited addresses, allowing them to visualise the distances between properties and their commonly frequented locations. Furthermore, users can bookmark properties of interest for convenient access in the future.

Overall, *Sweet Home Finder* is dedicated to providing users with a smooth and user-friendly experience throughout their property search journey. By integrating useful features and prioritising user convenience, the application aims to revolutionise the way users navigate the real estate market.

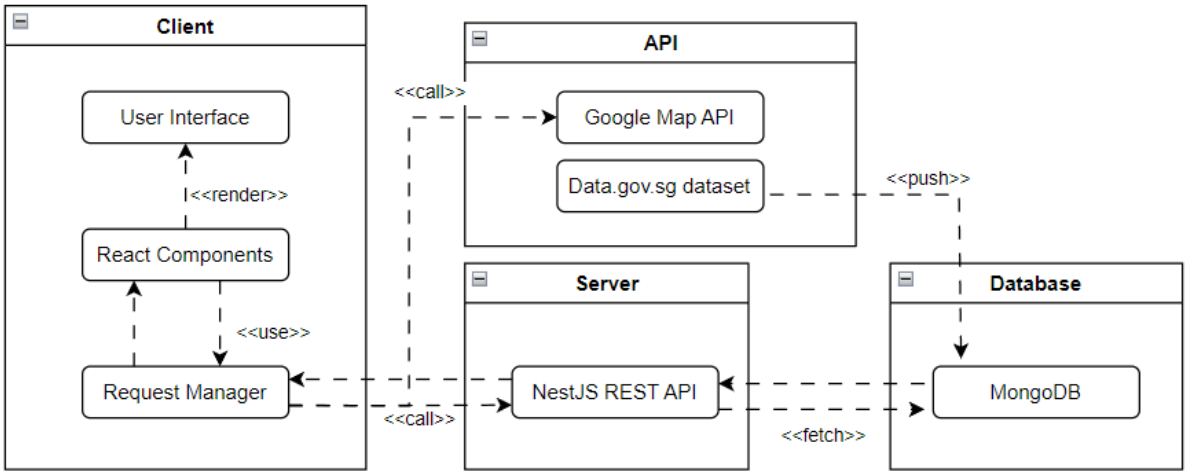
1.5 References

i. IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications. IEEE Standards Association. (n.d.). Retrieved April 17, 2024, from https://standards.ieee.org/ieee/830/1222/
ii. ReactJS Documentation. React. (n.d.). Retrieved March 12, 2024, from https://reactjs.org/docs/getting-started.html
iii. NestJS Documentation. NestJS. (n.d) Retrieved March 12, 2024, from https://docs.nestjs.com/
iv. MongoDB Documentation. MongoDB. (n.d) Retrieved March 12, 2024, from https://www.mongodb.com/docs/manual/
v. Google Maps Platform Documentation. Google Maps. (n.d) Retrieved March 12, 2024, from https://developers.google.com/maps/documentation

2. Overall Description

2.1 Product Perspective

Sweet Home Finder is a new, self-contained web application designed to serve as a user-friendly property finder platform. A simplified overall system architecture diagram is shown to introduce a brief overview of the operation of the *Sweet Home Finder* application.



2.2 Product Functions

The *Sweet Home Finder* application contains various functionalities. These functionalities can be categorised into 3 main categories: *Accounts*, *Logged In Functionalities* and *Property Search*. This section provides a breakdown list of the functionalities that Sweet Home Finder offers. For more detailed information regarding each functionality such as activity flows, conditions, assumptions and functional requirements, please refer to Section 4. System Features in this documentation.

2.2.1 Accounts

i. Account Registration
ii. Account Login
iii. Account Logout
iv. Account Reset Password

2.2.2 Logged In Functionalities

i. Bookmark Property
ii. Frequently Visited Address
iii. Update Account Information
iv. Routing

2.2.3 Property Search

i. Search Bar Filter
ii. Search Property
iii. View Property

2.3 User Classes and Characteristics

The *Sweet Home Finder* web application targeted different anticipated user classes shown in this section.

2.3.1 Casual Browsers

Attributes	**Descriptions**
Frequency of use	Low
Subset of functions used	Search Property category, bookmark listings
Technical expertise	Low
Characteristics	Individuals who are not currently looking to buy or rent but like to stay informed about the real estate market or dream about future purchases.

2.3.2 Home Buyers

Attributes	**Descriptions**
Frequency of use	High
Subset of functions used	All
Technical expertise	Medium
Characteristics	An individual who is actively searching for the property which best suits their preferences, according to location, nearby amenities, price and size. They may even want to take into consideration the distance between the property and their frequently visited addresses when searching for property. They can bookmark the potential properties that they would like to find more conveniently in the future.

2.3.1 Real Estate Agents

Attributes	**Descriptions**
Frequency of use	High
Subset of functions used	All
Technical expertise	Low
Characteristics	Individuals or entities investing in real estate to generate rental income or capital appreciation. They are interested in market trends, investment returns, and risk assessment.

2.4 Operating Environment

This section describes the production and development environment of *Sweet Home Finder* web application.

2.4.1 Production Environment

The web application requires an internet browser which supports at least:

- HTML5 or above
- CSS3 or above
- Javascripts

2.4.2 Development Environment

The web application should be developed or built under the following settings:

Front-end development: ReactJS is an open-source front-end JavaScript library maintained by Meta. The development team uses ReactJS to build all user interfaces of the web applications as it features reusable components which drastically speed up the development process.

Edition: ReactJS (version 18.2.0)

Back-end development: NestJS is a powerful open-source framework developed by NestJS team for building efficient, reliable, and scalable server-side applications with TypeScript. The development team utilises NestJS to construct the backend infrastructure of web applications due to its robust features and seamless integration with TypeScript. NestJS

simplifies backend development by providing a modular and structured approach, allowing developers to create reusable modules and components.

Edition: NestJS (version 10.3.8)

Database: MongoDB is a versatile NoSQL database solution utilised by the development team for storing and managing data in web applications. MongoDB's flexible document-based data model and scalable architecture make it an ideal choice for handling diverse data requirements. The development team integrates MongoDB into the backend infrastructure of web applications to store and retrieve data efficiently.

Edition: MongoDB (version 7.0)

2.5 Design and Implementation Constraints

The constraints for the development team are as follows:

- The application must meet performance requirements, including fast loading times, responsive user interface interactions, and efficient data retrieval from the database.
- The application must be designed to accommodate potential future growth in user base and property listings.
- The application must be optimised to run efficiently on a variety of devices, including desktops, laptops, tablets, and smartphones.
- The application must integrate with mapping services (e.g., Google Maps) for displaying property locations and nearby amenities, necessitating adherence to specific APIs and communication protocols.

2.6 User Documentation

Users can refer to the demo video, which serves as a guide for them to use and operate Sweet Home Finder web application. The video can be accessed at: Sweet Home Finder Demo

2.7 Assumptions and Dependencies

Assumptions:

- The availability and reliability of third-party APIs for mapping services (Google Maps) and property data sources (data.gov.sg) are assumed to be consistent throughout the development and operation of the application.
- The accuracy and completeness of property data, including listings and nearby amenities information, are assumed to be maintained by third-party data providers and are not under direct control of the application.
- Users will have consistent access to the internet and the application's performance and functionality will not be significantly impacted by fluctuations in network connectivity.

Dependencies:

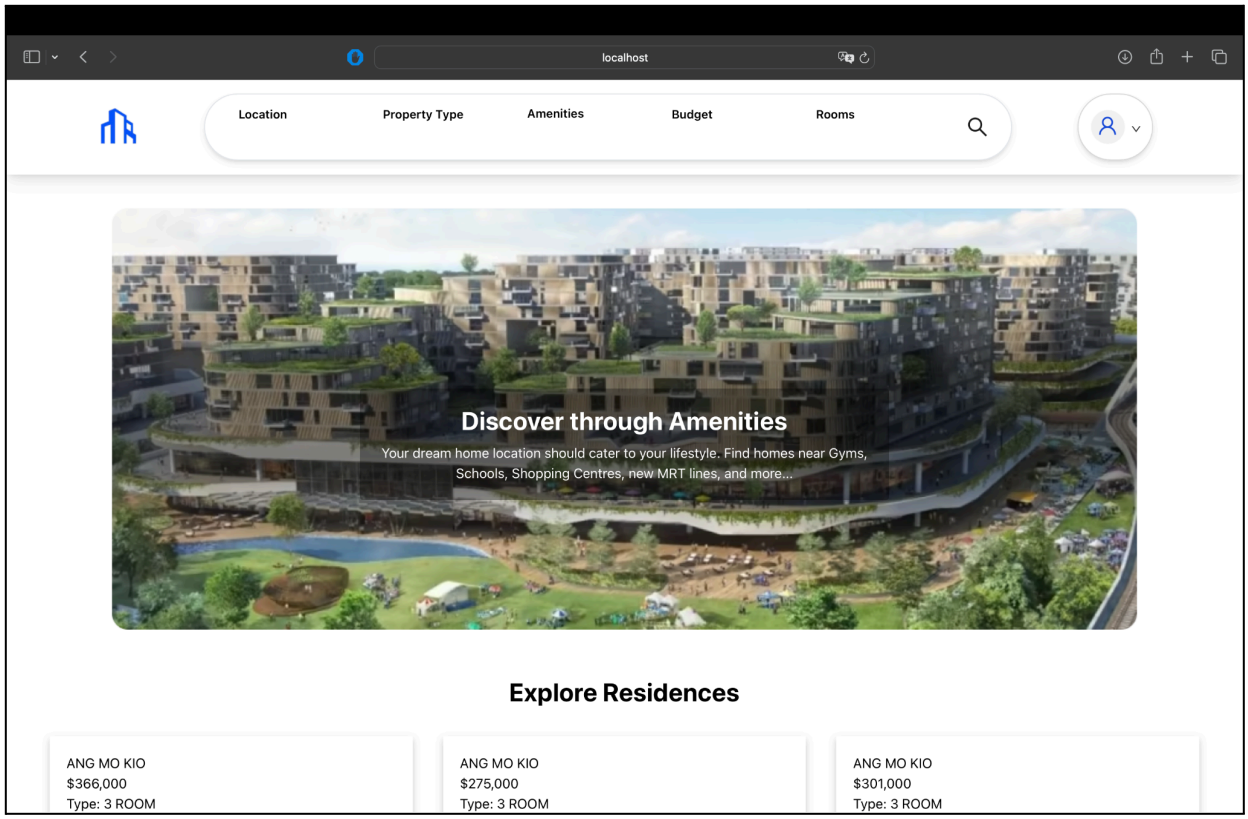
- The application depends on reliable access to third-party APIs for mapping services and property data sources.
- The application relies on MongoDB as the primary database solution for storing property data, user information, and application state, requiring consistent availability and performance of the MongoDB database server.
- The front-end development depends on the use of ReactJS for building user interfaces, requiring compatibility with ReactJS libraries and components.

3. External Interface Requirements

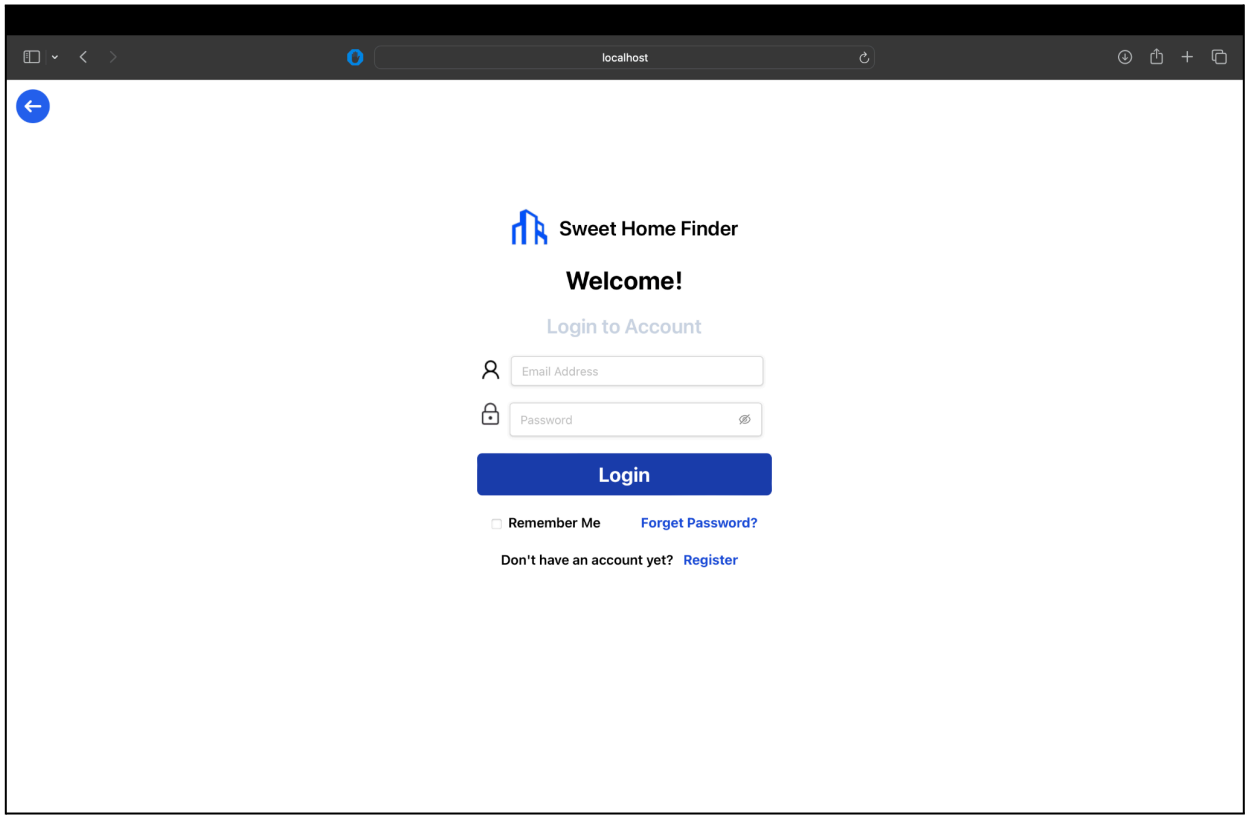
3.1 User Interfaces

The *Sweet Home Finder* web application is designed to be viewed with an aspect ratio that matches a desktop browser.

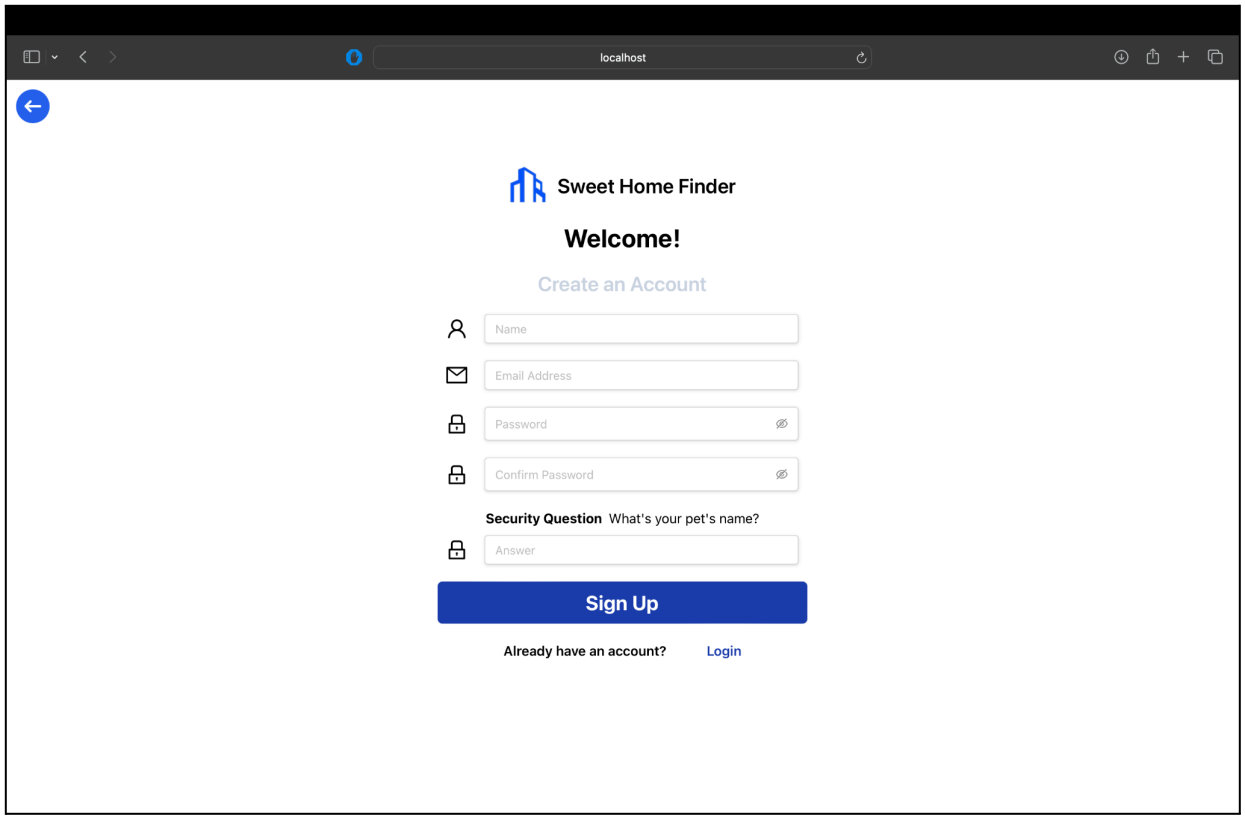
3.1.1 Landing Page



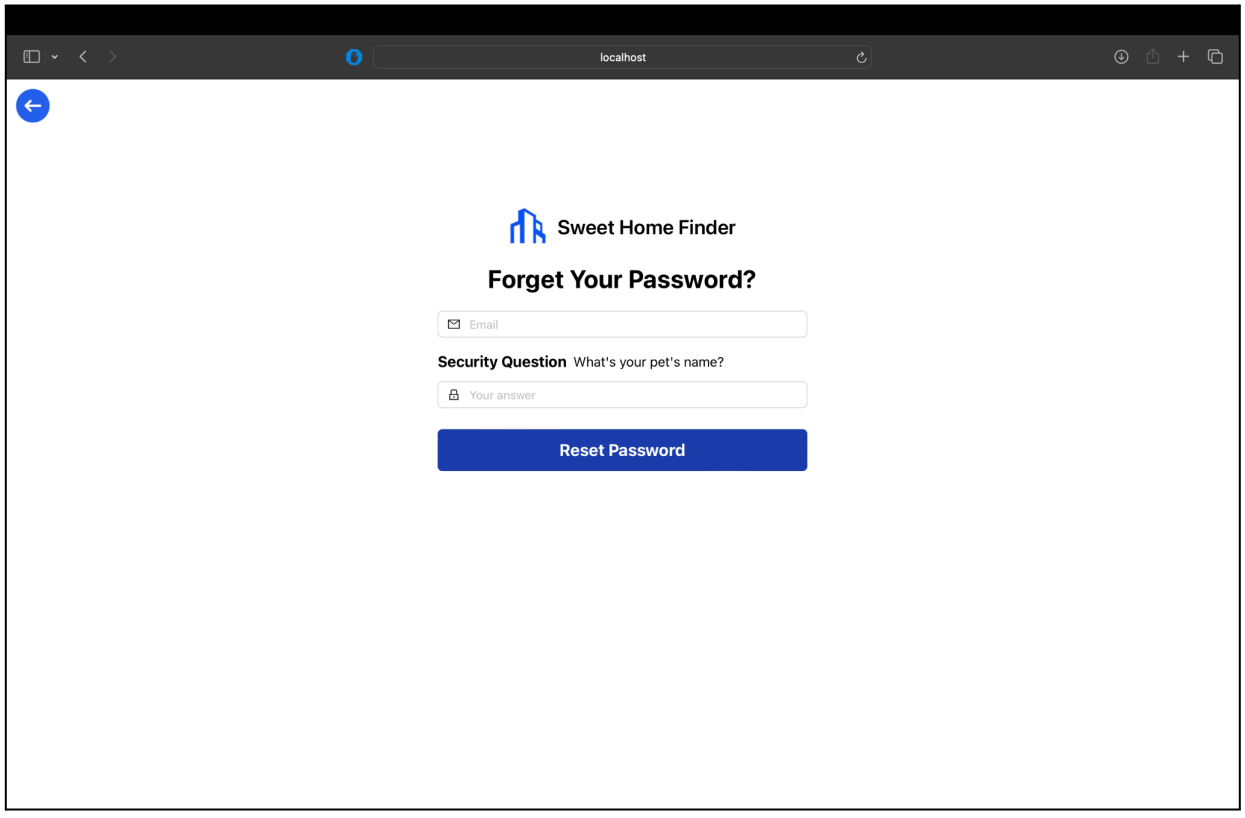
3.1.2 Login Page



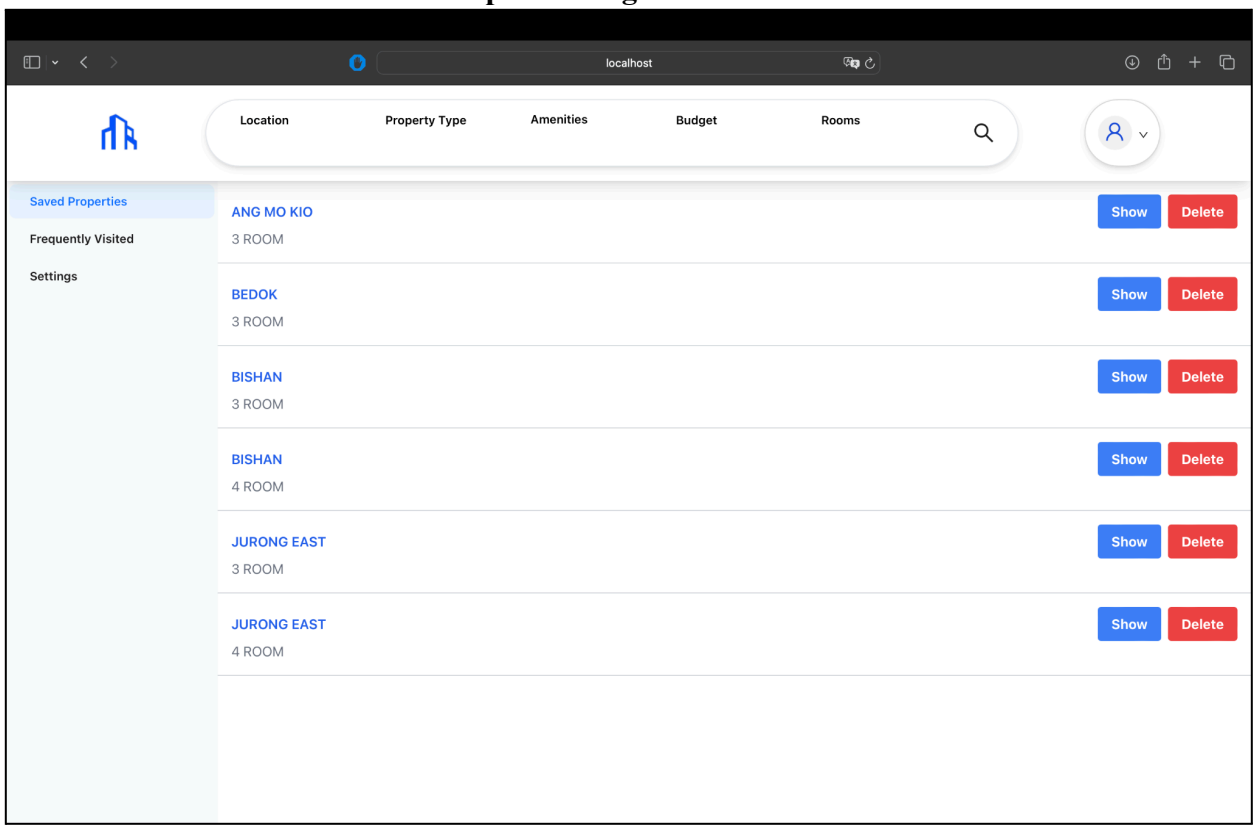
3.1.3 Signup Page



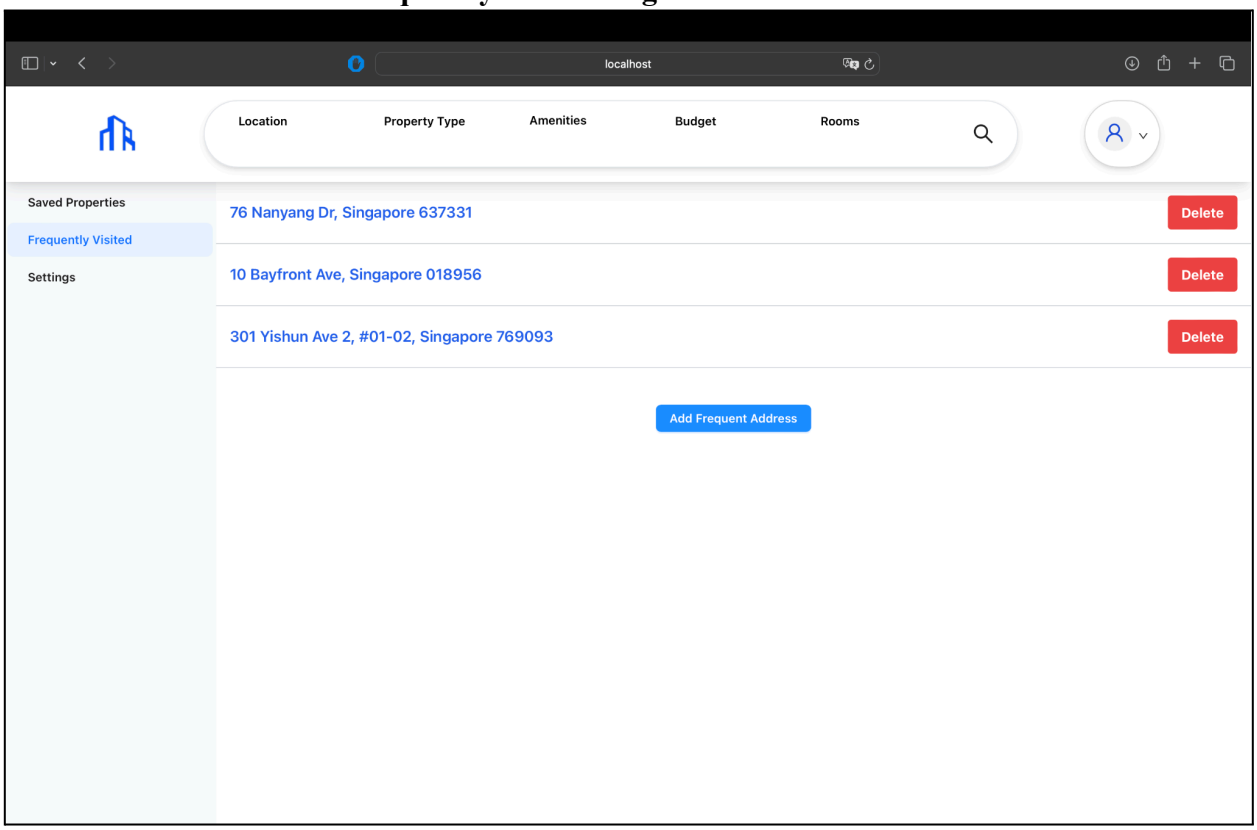
3.1.4 Forget Password Page



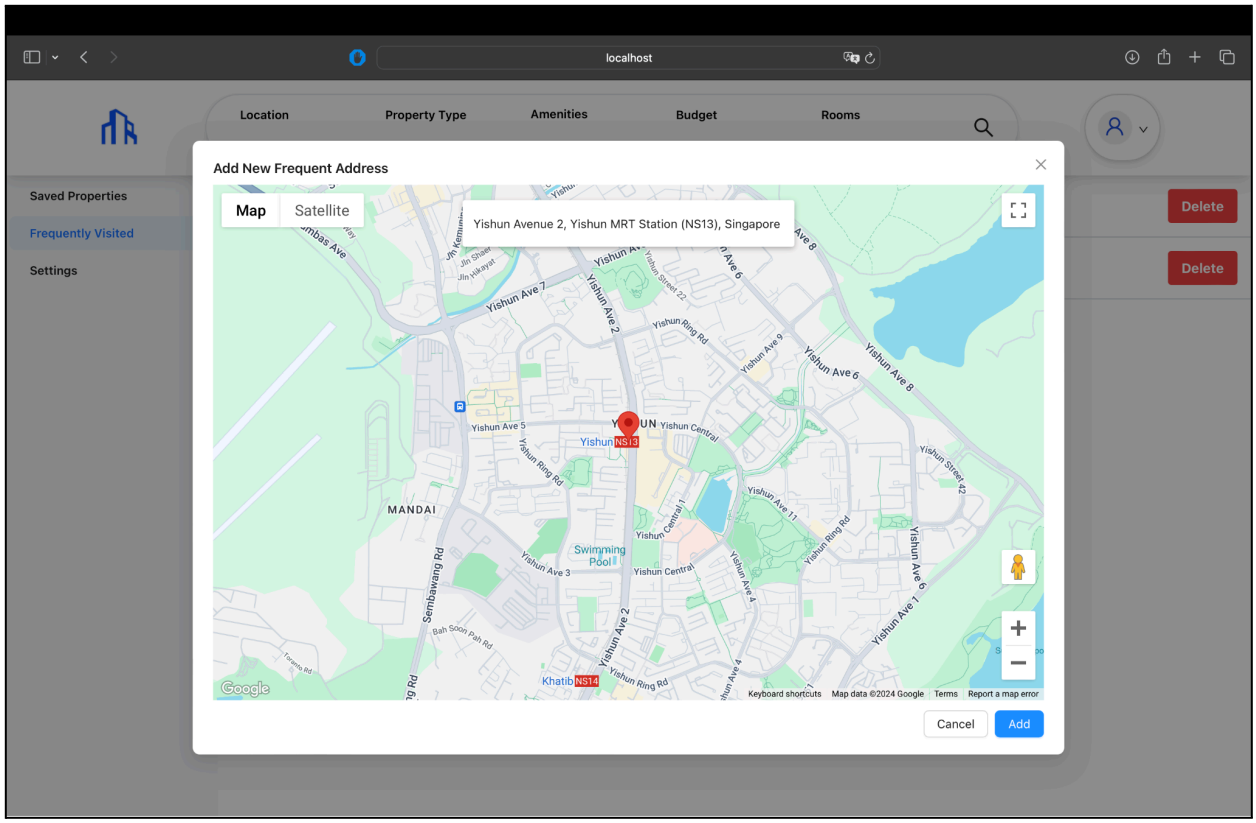
3.1.5 Dashboard - Saved Properties Page



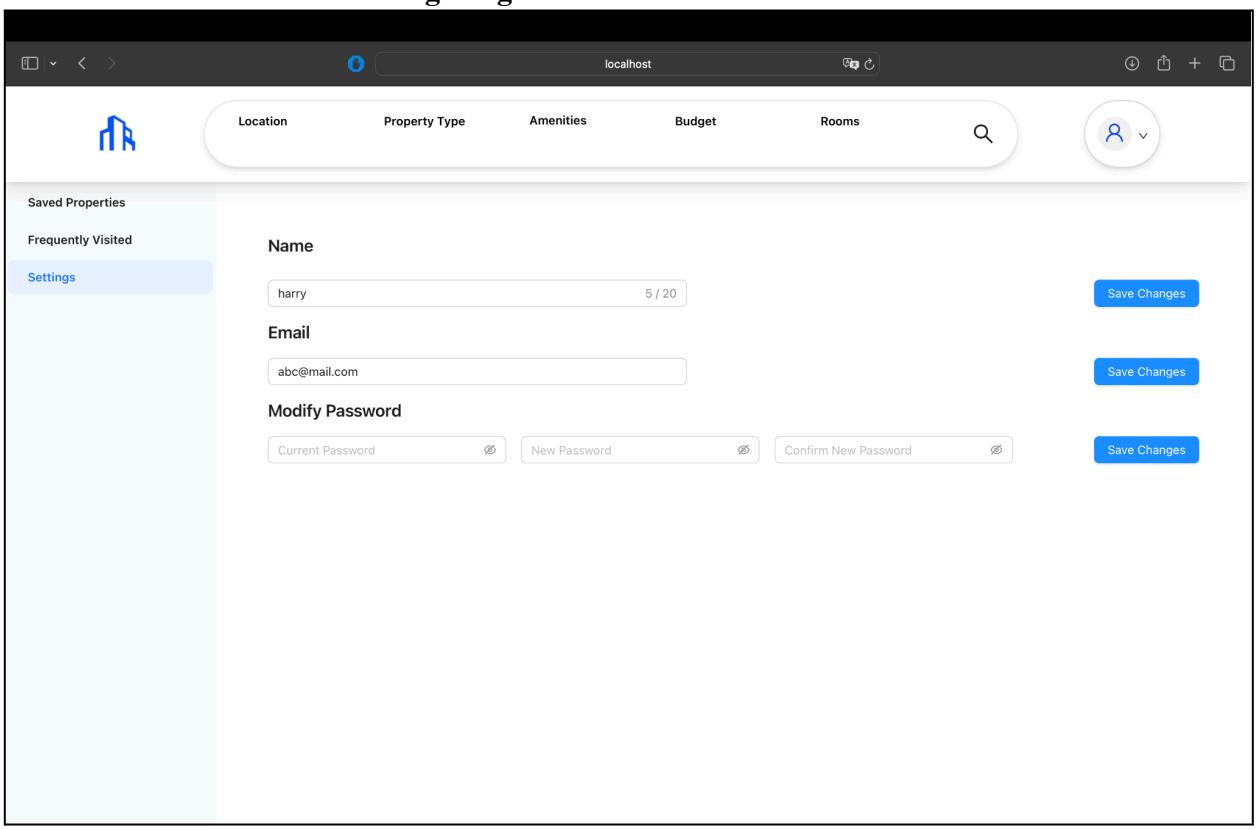
3.1.6 Dashboard - Frequently Visited Page



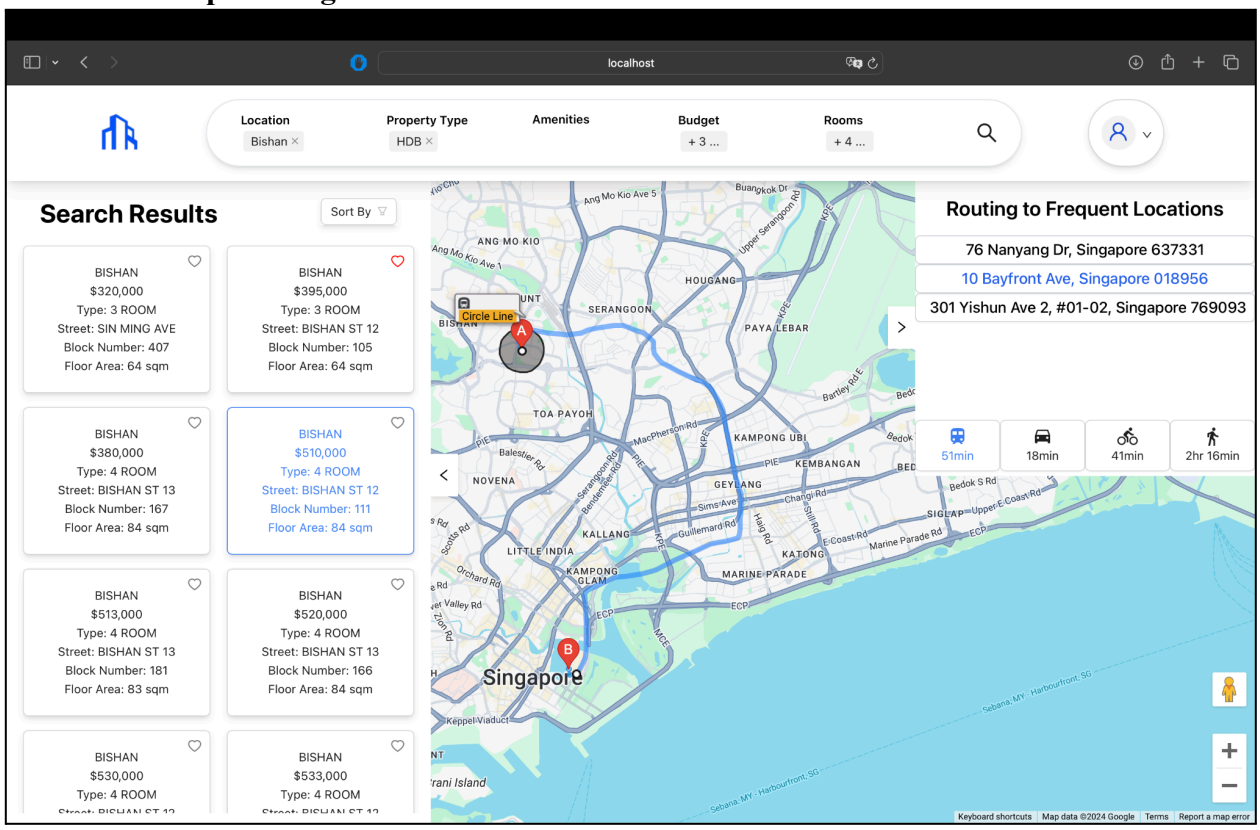
3.1.7 Add Frequently Visited



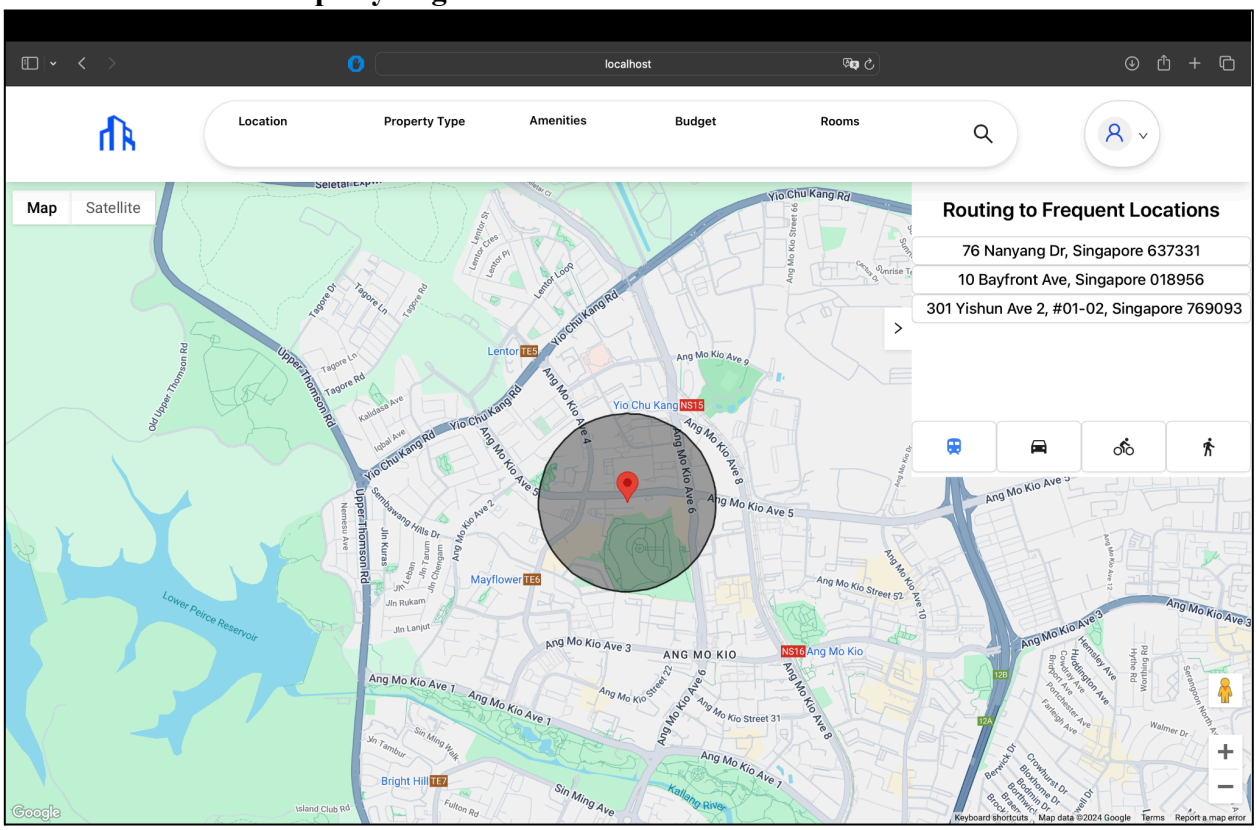
3.1.8 Dashboard - Settings Page



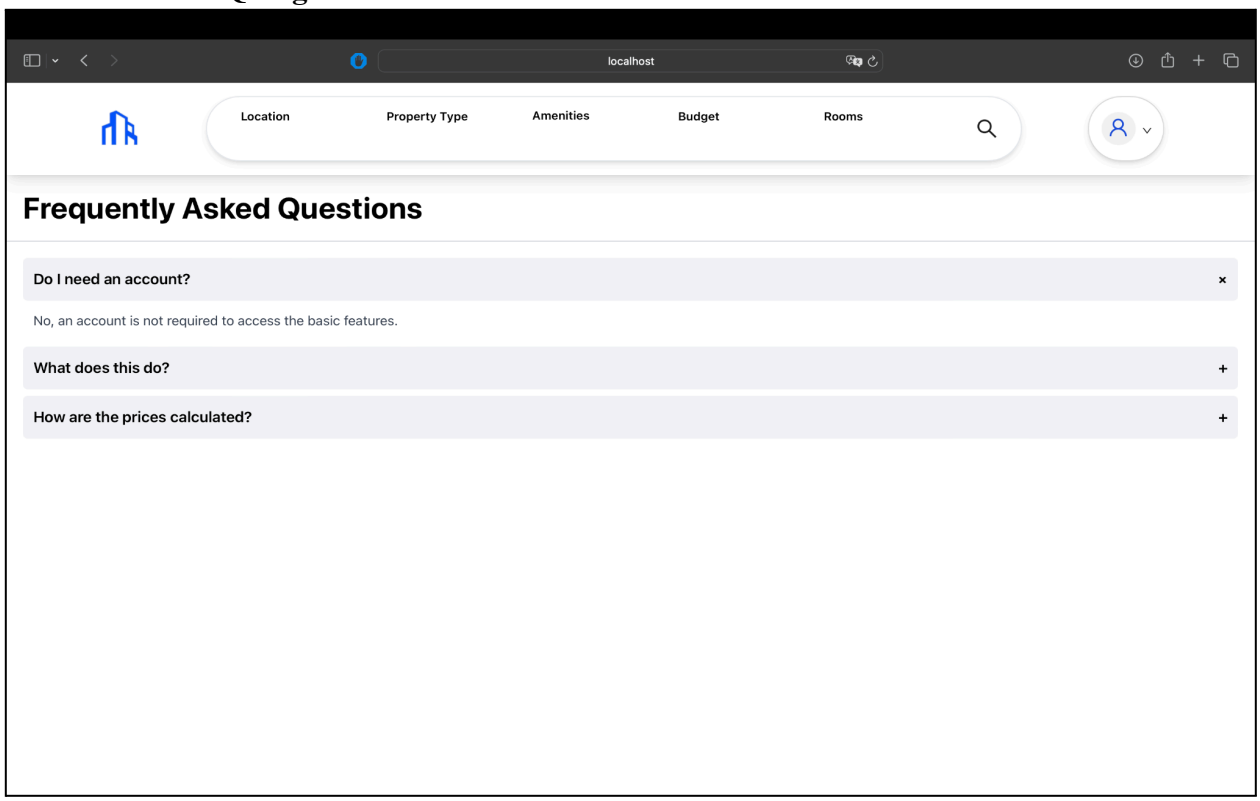
3.1.9 Explore Page



3.1.10 Show Property Page



3.1.11 FAQ Page



3.2 Hardware Interfaces

This section covers all hardware interface requirements for the *Home Sweet Finder* web application to achieve its desired functionalities. The requirements are divided into client-side requirements and server-side requirements to facilitate easier reading for the respective stakeholders.

3.2.1 Client-side Requirements

The *Sweet Home Finder* web application supports all desktop computers and laptops. The device must support the usage of a desktop browser which fulfils the requirements as indicated in *2.4.1 Production Environment of Sweet Home Finder.*

3.2.2 Server-side Requirements

The Sweet Home Finder back-end server must be hosted and run on a server-computer. The back-end server will perform Create, Read, Update, Delete (CRUD) operations on the back-end database. The database is hosted on a MongoDB server.

3.3 Software Interfaces

3.3.1 Software Components and Versions

The *Home Sweet Finder* web application utilises a back-end server and a database to handle all the system features implementation. The back-end server is built and implemented using the NestJS framework, version 10.3.8. The *Home Sweet Finder* development team adopts MongoDB, as the database of the web application.

3.3.2 Software Architecture

The *Home Sweet Finder* web application software architecture must follow the Model-View-Controller design pattern. The interface must be able to connect to a database to store persistent data in BSON format.

3.4 Communications Interfaces

The *Home Sweet Finder* web application communication architecture must follow a client-server model. Each communication must go through a REST-styled Application Programming Interface (API) provided by the backend server. Each request must also be served over HyperText Transfer Protocol Secure (HTTPS).

Communication from client to server must invoke GET and POST requests. Communication from server to client must serve data standardised in JavaScript Object Notation (JSON) format.

4. System Features

4.1 User Account Registration

4.1.1 Description and Priority

New users of *Sweet Home Finder* can register for an account. Upon registration, the user's inputted email, username, and password are stored in the database. The password will be encrypted to protect the user's privacy. Further updates by the user, such as saving properties and frequently visited addresses, will use this registered account as a reference.

4.1.2 Stimulus/Response Sequences

Use Case ID:	001		
Use Case Name:	Register		
Created By:	Tan Wu Ji	**Last Updated By:**	Boon Yi
Date Created:	11/2/2024	**Date Last Updated:**	23/2/2024

Actor:	App User, Database

Description:	App User can choose to register for an account which will be stored in the Database
Preconditions:	1. Database must be up and online 2. App User must be connected to Internet
Postconditions:	App User has successfully registered an account with a unique username and password. OR App User is notified of the unsuccessful registration of an account
Priority:	
Frequency of Use:	
Flow of Events:	1. At the home page, App User clicks on “Sign up” and is redirected to the registration page. 2. App User inputs a valid email, a unique username, a password with at least 8 characters, 1 upper-case letter and 1 lower-case letter and the repeated password in the respective fields in the registration page. 3. App User checks the checkbox of “I agree to the Terms of Use and Private Policy” and clicks on ” “Sign Up” to register and create an account. 4. The system verifies the username is unique and the password satisfies the constraints. 5. The system stores App User’s information in the Database. 6. App User is notified that the registration is successful.

Alternative Flows:	AF-S2: App User left input field(s) blank. 1. When the App User clicks on "Sign Up", the system displays the message "Please fill in all input fields to register for an account!" above the registration page. 2. The system returns to Step 2 and waits for the App User inputs. AF-S3: App User did not check the checkbox of "I agree to the Terms of Use and Privacy Policy". 1. When the App User clicks on "Sign Up", the system displays the message "Please tick the checkbox for acknowledging the Terms of Use and Privacy Policy!" above the registration page. 2. The system returns to Step 3 and waits for the App User inputs. AF-S4: App User inputs a taken username. 1. The system displays the message "Username has been taken. Please input a new username!" above the registration page. 2. The system returns to Step 2 and waits for the App User inputs. AF-S4: App User inputs a password that does not satisfy the given requirements. 1. The system displays the message "Password must contain at least 8 characters, 1 upper-case letter and 1 lower-case letter!!!" above the registration page. 2. The system returns to Step 2 and waits for the App User inputs. AF-S4: App User inputs mismatched passwords. 1. The system displays the message "Passwords do not match! Please check again!" above the registration page. 2. The system returns to Step 2 and waits for the App User inputs.
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.1.3 Functional Requirements

1. The user must be able to register for an account in the system.

1.1. The system must display text fields for the user to enter his information.

1.1.1. A text field for the userID must be included.

1.1.2.A text field for the email address must be included.

1.1.3. A text field for the password must be included.

1.2. The system must display a checkbox stating, “I agree to the Terms of Use and Private Policy”.

1.3. The user must fill in all the text fields and check the checkbox before clicking the “Sign Up” button.

1.4. The system must verify the text fields filled in by the user before creating the account.

1.4.1. The userID must be unique for all users of the system.

1.4.2. The email address used must be new within the system.

1.4.3. The email address used must be valid in receiving emails.

1.4.3.1. The system must send a One Time Password (OTP) to the user’s email address.

1.4.4. The password used must be at least 8 characters in length, with 1 upper-case letter and 1 lower-case letter.

1.5. The system must store the user’s information in the user database.

1.6. The system must display a message if the registration is successful.

1.7. The system must log the user into the system.

4.2 User Account Login

4.2.1 Description and Priority

Existing users of *Sweet Home Finder* can login to their account using the email and password they registered with. Users must login to access *Sweet Home Finder*’s user-specific features.

4.2.2 Stimulus/Response Sequences

Use Case ID:	002		
Use Case Name:	Login		
Created By:	Tan Wu Ji	**Last Updated By:**	Boon Yi
Date Created:	11/2/2024	**Date Last Updated:**	23/2/2024

Actor:	App User, Database
Description:	App User can login to his/her account with their username and password which are stored in the Database. Logging in allows them to save their preferences.
Preconditions:	1. The Database must be up and online. 2. App User must be connected to the Internet. 3. App User has a registered account.
Postconditions:	App User has successfully logged into his/her account. OR App User is notified of the unsuccessful login to his/her account.
Priority:	
Frequency of Use:	
Flow of Events:	1. At the home page, App User clicks on “Log in” and is redirected to the login page. 2. App User inputs his/her username and password. 3. App User clicks on “LOGIN”. 4. The system verifies the username and password with the Database. 5. Once the information is verified, the App User is redirected to the home page.
Alternative Flows:	AF-S2: If the App User inputs an incorrect username or password. 1. When the App User clicks on “LOGIN”, the system displays the message “Invalid username and/or password!” above the registration page. 2. The system returns to Step 2 and waits for the App User inputs.
Exceptions:	EX-1: The App User did not register for an account. 1. The system displays the message “Please register an account in order to log in”. 2. The system directs the App User to the registration page.
Includes:	Register
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.2.3 Functional Requirements

2. The user must be able to log into the system.

2.1. The system must display text fields for the user to enter his information.

2.1.1. A text field for the userID must be included.

2.1.2. A text field for the password must be included.

2.2. The user must fill in all the text fields before clicking the “Login” button.

2.3. The system must verify that the fields filled in by the user are within the user database.

2.3.1. The system must be able to find the userID inputted by the user.

2.3.2. The system must verify that the password entered matches the password associated with the userID.

2.4. The system must log the user into the system.

4.3 SearchHome

4.3.1 Description and Priority

Users of *Sweet Home Finder* will be able to search for properties with keywords such as location regardless of whether they are logged in or not. The system will retrieve information from the API data and display the result for the App User.

4.3.2 Stimulus/Response Sequences

Use Case ID:	003		
Use Case Name:	SearchHome		
Created By:	Tan Wu Ji	**Last Updated By:**	Boon Yi
Date Created:	11/2/2024	**Date Last Updated:**	23/2/2024

Actor:	App User, API data
Description:	App User will be able to search for properties with keywords such as location or the properties’ name. The system will retrieve information from the API data and display the result for the App User.
Preconditions:	1. Database is up and online 2. App User is connected to the Internet.
Postconditions:	App User obtained a list of searched properties based on the keywords. OR App User is unable to obtain a search result based on the keywords.

Priority:	
Frequency of Use:	
Flow of Events:	1. At the home page, the App User clicks on the search bar, types in keywords in the search bar and clicks on the "Search" icon/button. 2. The system retrieves a list of properties from the API data and searches for the relevant properties based on the keywords. 3. The system displays information of the relevant properties such as property name, price and address.
Alternative Flows:	AF-S1: App User inputs nothing and clicks on the search icon. 1. The system displays a list of 20 random properties. 2. The system returns to Step 1 and waits for the App User actions.
Exceptions:	EX-1: The keyword input by the App User does not match any searched items. 1. The system displays the message "No relevant properties are found!". 2. The system returns to Step 1 and waits for the App User actions.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.3.3 Functional Requirements

3. The user must be able to view an interactive map of Singapore.

3.1. The system must display a map of Singapore with the available properties highlighted with distinct markers.

3.2. The user must be able to click on a property to view detailed information on it.

3.3. The system must display information for each property displayed.

3.3.1. The system must display the name of the property.

3.3.2. The system must display the price of the property.

3.3.3. The system must display the address of the property.

3.4. The user must be able to zoom in and zoom out of the map.

3.5. The user must be able to pan through the map.

4. The user must be able to search for properties based on keywords.

4.1. The user must be able to click on a search bar.

4.1.1. The user must be able to fill text in the search bar.

4.2. The system must return a list of properties that are relevant to the keywords entered in the search bar.

4.3. The system must display information for each property displayed.

4.3.1. The system must display the name of the property.

4.3.2. The system must display the price of the property.

4.3.3. The system must display the address of the property.

5. The user must be able to select a specific property to view more details of.

4.4 Bookmark

4.4.1 Description and Priority

Logged in Users can bookmark their interested properties. The system saves the bookmarked properties in the database. App User is able to quickly find their properties when they use the app later on.

4.4.2 Stimulus/Response Sequences

Use Case ID:	004		
Use Case Name:	Bookmark		
Created By:	Tan Wu Ji	**Last Updated By:**	Boon Yi
Date Created:	11/2/2024	**Date Last Updated:**	23/2/2024

Actor:	App User, Database
Description:	App User can bookmark their interested properties. The system saves the bookmarked properties in the database. App User is able to quickly find their properties when he/she uses the app later.
Preconditions:	1. App User is connected to the Internet. 2. App User registered for an account with the Database. 3. App User has logged in to his/her account.
Postconditions:	App User has successfully added the property into their bookmark and saved into the database. OR App User has successfully removed the property from their bookmark and updated the database.
Priority:	
Frequency of Use:	

Flow of Events:	1. When the App User clicks the property, the system directs App User to the detailed information page. 2. The "bookmark" icon located at the top right corner displayed in grey colour. 3. App User clicks the "bookmark" icon. 4. The system saves the property into the App User's database. 5. The "bookmark" icon turns yellow, indicating that this property is successfully added into the bookmark database.
Alternative Flows:	AF-S2: "Bookmark" icon is in yellow colour 1. When the App User clicks the yellow colour bookmark, the system removes the property from the database. 2. The system changes the "bookmark" icon to grey colour, indicating that the property is successfully removed from the database. 3. The system returns to Step 1 and waits for the App User actions. AF-S2: App User does not log into his/her account 1. The system displays the message "Please log into your account to bookmark this property". 2. The system directs the App User to the login page. 3. App User uses the LOGIN use case to log into his/her account. 4. The system returns to Step 1 and waits for the App User input.
Exceptions:	
Includes:	Login
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.4.3 Functional Requirements

5.1. The user must be able to bookmark the selected property.

5.1.1. The system must store the selected property within a database.

4.5 TransportOption

4.5.1 Description and Priority

Logged in Users can select transport options such as Public Transport, Car, Bicycling, and Walking.

4.5.2 Stimulus/Response Sequences

Use Case ID:	005		
Use Case Name:	TransportOption		
Created By:	Tan Wu Ji	**Last Updated By:**	Boon Yi
Date Created:	11/2/2024	**Date Last Updated:**	23/2/2024

Actor:	App User
Description:	System prompts App User to choose the transport options such as MRT, CAR, BUS.
Preconditions:	1. App User is connected to the Internet. 2. App User is in the detailed information page.
Postconditions:	App User successfully input the transportation option. OR App User failed to input the transportation option.
Priority:	
Frequency of Use:	
Flow of Events:	1. The system prompts App User to select the transportation option. 2. App User chooses the corresponding option and clicks “Select”. 3. The system saves the App User’s choice.
Alternative Flows:	AF-S2: App User does not choose any option and clicks “Select” 1. The system displays the message “Please choose your transportation option”. 2. The system returns to Step 1 and waits for the App User input.
Exceptions:	EX-1: App User clicks “Cancel” button 1. The system returns to the detailed information page.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.5.3 Functional Requirements

5.2. The user must be able to search for routes from the selected property to a location inputted by the user.

5.2.1. The system must display the time taken and the transportation used for the route.

4.6 ShowETA

4.6.1 Description and Priority

Logged in Users can see the Estimated Time of Arrival (ETA_ based on their input of transportation options.

4.6.2 Stimulus/Response Sequences

Use Case ID:	006		
Use Case Name:	ShowETA		
Created By:	Tan Wu Ji	**Last Updated By:**	Boon Yi
Date Created:	11/2/2024	**Date Last Updated:**	23/2/2024

Actor:	App User, API data
Description:	The system shows the Estimated Time of Arrival (ETA) based on the App User's input of the transportation option.
Preconditions:	1. App User is connected to the Internet. 2. App User is in the detailed information page. 3. API data is up and connected.
Postconditions:	The system successfully showed the ETA corresponding to the transport option. OR The system failed to show the ETA.
Priority:	
Frequency of Use:	
Flow of Events:	1. App User clicks "Show ETA". 2. App User uses the TransportOption use case to select the transportation option. 3. The system retrieves the property's address from the API data. 4. The system makes API request calls to calculate the ETA between the addresses. 5. The system displays the ETA.
Alternative Flows:	AF-S4: The system failed to calculate the ETA 1. The system displays the message "Calculating the ETA again…". 2. The system returns to Step 3.
Exceptions:	EX-1: The system failed to calculate ETA in 30 seconds 1. The system uses the FrequentAddress use case to wait for App User to input the address again.
Includes:	TransportOption
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.6.3 Functional Requirements

5.2. The user must be able to search for routes from the selected property to a location inputted by the user.

5.2.1. The system must display the time taken and the transportation used for the route.

4.7 FrequentAddress

4.7.1 Description and Priority

Logged in Users can input the addresses of places they frequent. The system will show the ETA and route from their properties of interest to the places they frequent.

4.7.2 Stimulus/Response Sequences

Use Case ID:	007		
Use Case Name:	FrequentAddress		
Created By:	Boon Yi	**Last Updated By:**	Boon Yi
Date Created:	12/02/2024	**Date Last Updated:**	23/02/2024

Actor:	App User, Database
Description:	App User can input the addresses of places they frequent. The system will show the ETA from their properties of interest to the places they frequent.
Preconditions:	1. App User is connected to the internet 2. App User is in the detailed information page
Postconditions:	The system successfully showed the ETA from their property of interest to their input address. OR The system failed to show the ETA from their property of interest to their input address.
Priority:	
Frequency of Use:	
Flow of Events:	1. App User click on the property of interest 2. System directs App User to the detailed information page 3. The “ShowETA” icon is located at the top right hand corner of the page 4. App User click on the “ShowETA” icon 5. System prompt the App User to input their address 6. System then calculate ETA using ShowETA function 7. System display ETA calculated
Alternative Flows:	AF-S5: App User input invalid address 1. System display “Unable to find address” 2. System return to step 5 and wait for App user input AF-S6: Unable to calculate ETA with corresponding transport option 1. System display “Transport Option not available” 2. System use TransportOption use case and wait for App user to select alternate transport option.

Exceptions:	EX-1: App User clicks "Cancel" button 1. The system returns to the detailed information page.
Includes:	ShowETA
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.7.3 Functional Requirements

5.2. The user must be able to search for routes from the selected property to a location inputted by the user.

5.2.1. The system must display the time taken and the transportation used for the route.

4.8 Show nearby amenities

4.8.1 Description and Priority

The system will display nearby amenities within 500 metres from Users' property of interest.

4.8.2 Stimulus/Response Sequences

Use Case ID:	008		
Use Case Name:	Show nearby amenities		
Created By:	Boon Yi	**Last Updated By:**	Boon Yi
Date Created:	12/02/2024	**Date Last Updated:**	23/02/2024

Actor:	App User, API data
Description:	The system will display nearby amenities from App User property of interest.
Preconditions:	1. App User is connected to the internet 2. App User is in the detailed information page
Postconditions:	The system successfully displays nearby amenities from App User property of interest. OR The system failed to display nearby amenities from App User property of interest.
Priority:	
Frequency of Use:	
Flow of Events:	1. App User click on the property of interest 2. System directs App User to the detailed information page 3. The "Amenities" icon is located at the top right hand corner of the page 4. App User click on the "Amenities" icon

	5. System display nearby amenities from property of interest
Alternative Flows:	AF-S5: No amenities nearby 1. System display “Unable to find any amenities” 2. System return to step 2
Exceptions:	EX-1: App User clicks “Cancel” button 1. The system returns to the detailed information page.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.8.3 Functional Requirements

5.3. The user must be able to view nearby amenities to the selected property.

5.3.1. The system must display amenities within a 500-meter radius from the selected property.

4.9 Reset Password

4.9.1 Description and Priority

The system will reset password to default password after the user successfully input valid email and the correct answer for the security question.

4.9.2 Stimulus/Response Sequences

Use Case ID:	009		
Use Case Name:	Reset Password		
Created By:	Wu Ji	**Last Updated By:**	Wu Ji
Date Created:	12/02/2024	**Date Last Updated:**	23/02/2024

Actor:	App User, API data
Description:	The system will reset password to default password
Preconditions:	1. App User is connected to the internet 2. App User is in the Login page
Postconditions:	The system successfully reset the user's password to default password such as 123456. OR The system failed to reset the user's password to default password such as 123456.
Priority:	

Frequency of Use:	
Flow of Events:	1. App User click on the reset password 2. System directs App User to the reset password page 3. App User inputs valid email and answers for security questions. 4. App User click on the "Reset Password" button 5. System display "Password is reset to default password 123456"
Alternative Flows:	AF-S4: Wrong email or security answer 1. System display "Wrong email or security answer" 2. System return to step 3
Exceptions:	EX-1: App User clicks "Back" button 2. The system returns to the Login page.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.9.3 Functional Requirements

5.4. The user must be able to reset the password in case he/she forgets the password.

5.4.1. The system must prompt the user to enter valid email and the correct security answer to authenticate the user before resetting the password to default password.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

1. The system must be able to successfully add an account to the database within 15 seconds after successful registration.
2. The system must be able to retrieve and display the search results of the user (property details and map data) within 20 seconds.
3. System updates (e.g., property listings or amenities information) must propagate to all users in real-time or near-real-time, with no more than a 60-second delay.
4. The system must support simultaneous access by up to 10,000 users without degradation of performance.

5.2 Usability Requirements

1. The system must be able to display an FAQ message of the functions the user can use.
2. The system must provide accessibility features in the future for users with disabilities, such as screen reader support and high-contrast visual options.
3. The system should offer multi-language support in the future to cater to Singapore's diverse population, including English, Mandarin, Malay, and Tamil.
4. The system should be compatible with major browsers including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.
5. The system should have an intuitive user interface that is accessible to users with varying levels of technical expertise.
6. The application must be responsive in the future, accessible on multiple devices, including desktops, tablets, and smartphones, and optimised for each.

5.3 Security Requirements

1. The system must use a hashing algorithm to encrypt the user’s password within the user database.
2. The system must implement secure HTTPS connections for all data transactions to prevent unauthorised data interception.
3. Regular security audits must be conducted at least once every six months to ensure the integrity and security of the user data.

5.4 Reliability Requirements

1. The system must not be down for more than 2 hours in 1 month.
2. Data backups should be performed daily to prevent data loss and ensure data recovery within 2 hours in the event of a system failure.
3. In the future, backups of user data and system databases should be performed daily, with the ability to restore data from any point within the last 30 days.
4. In the future, automated failover mechanisms should switch to a standby system in the event of the failure of the primary system to ensure service continuity in the event of a hardware or software failure.

5.5 Maintainability Requirements

1. The software should be built in a modular fashion, allowing for easy updates and maintenance without affecting the entire system.
2. Comprehensive documentation of the system’s architecture, code, and APIs must be maintained and regularly updated
3. Establish traceability links to maintain consistency across development, ensure requirements are addressed, and understand relationships between parts of the programme.
4. A system for managing and deploying updates should be established, including security patches and feature improvements.

6. Other Requirements

6.1 Legal and Compliance Requirements

1. The software must comply with the Personal Data Protection Act (PDPA) of Singapore in handling personal information.
2. The use and distribution of third-party software components such as APIs, data, and libraries must adhere to their respective licences.

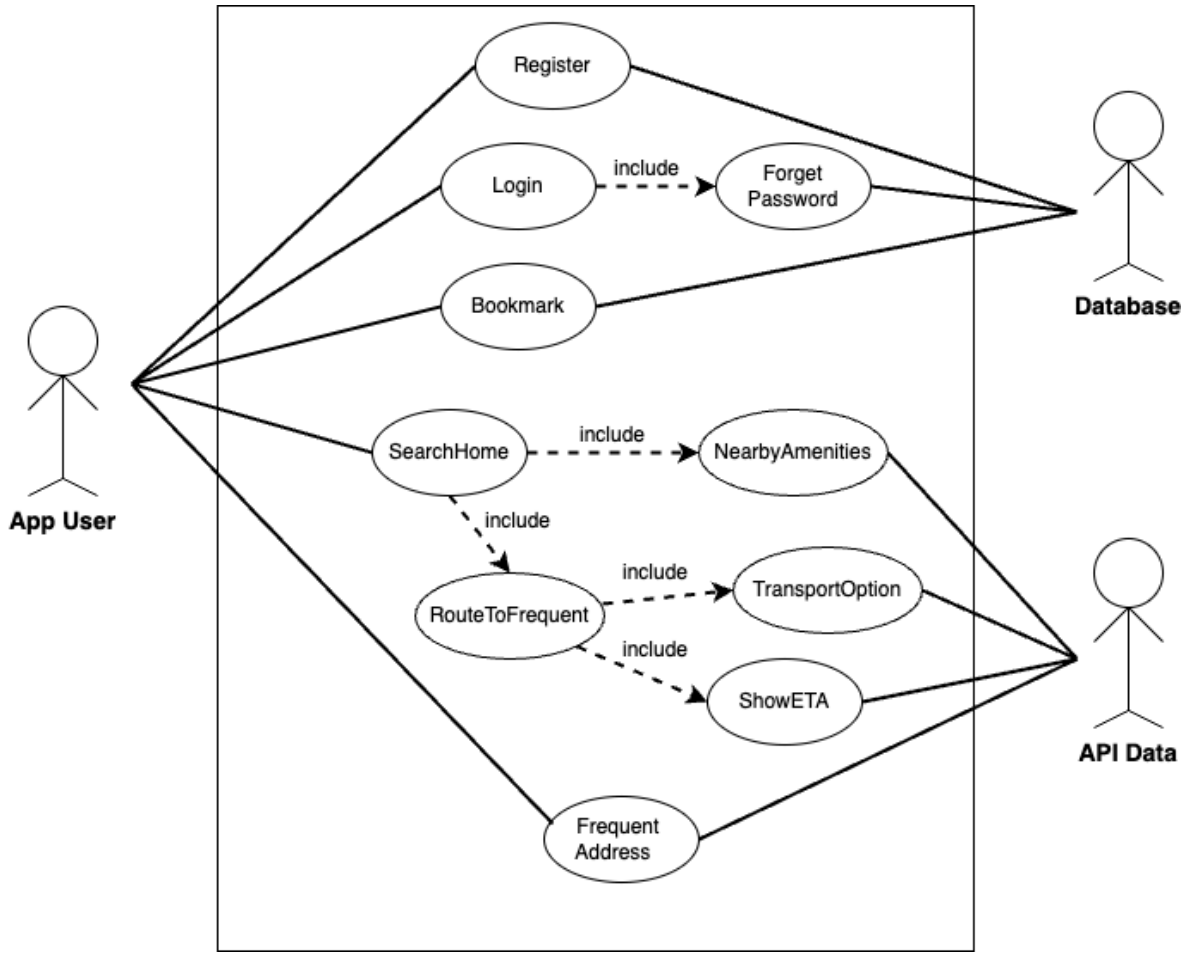
Appendix A: Data Dictionary

Created by:	Benjamin Harry Oates	Updated by:	Benjamin Harry Oates
Created Date:	11 February 2024	Updated Date:	19 April 2024

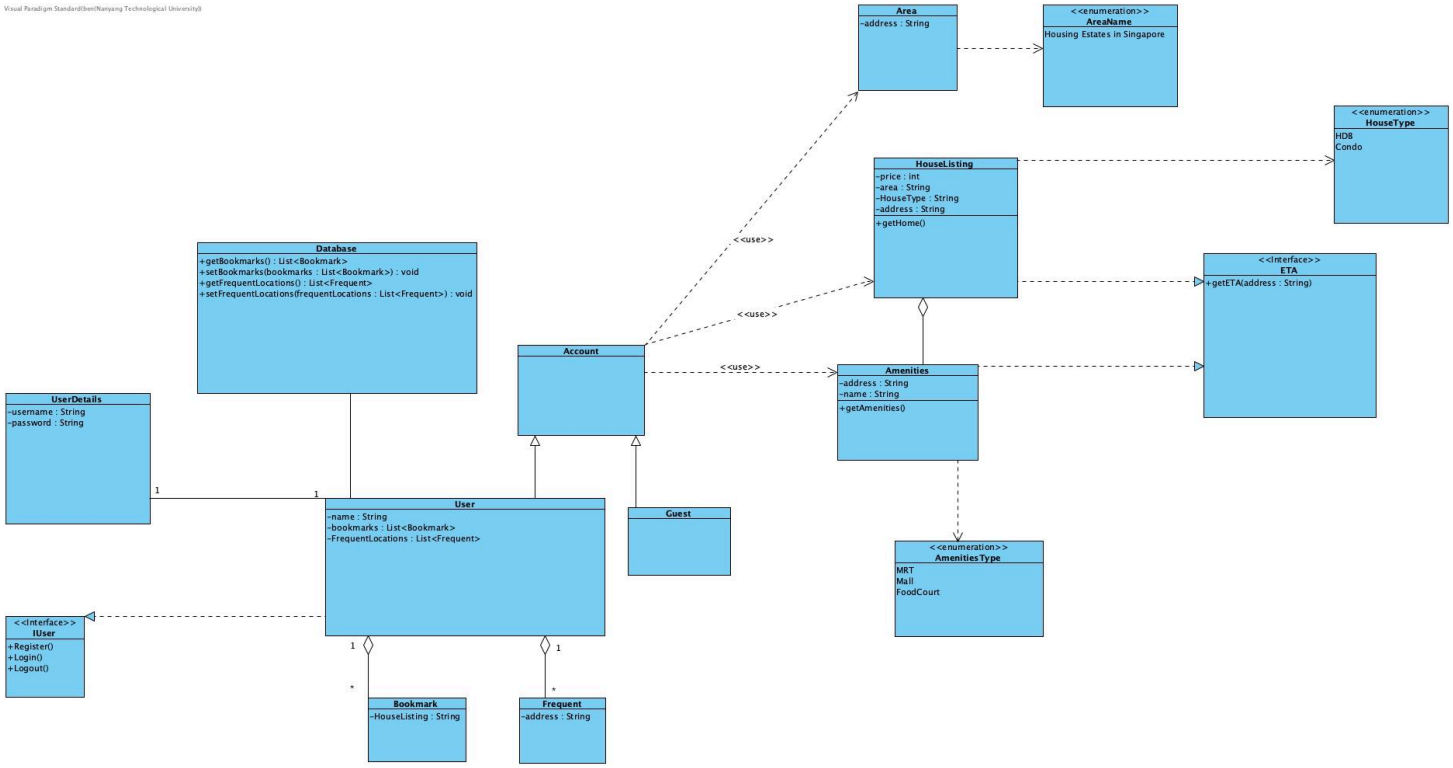
User	An individual who uses Sweet Home Finder application. Logged-in users will be granted permission to use additional services provided by the application. Users will be able to view the properties listed and also bookmark them for future reference. Users will be able to view the travel time from the viewing property to their frequent locations.
Bookmark	A list which contains all the users saved properties that they have liked and saved for them to refer to in the future.
Username	A unique identifier (user email) for their future log in.
Property	A resale HDB property that is available for purchase and currently in the database.
Frequently Visited Address	A location that the user frequently travels to such as their parents home or their work. The user will be given the option to input their frequent locations and see the ETA from the property they are viewing to these locations.
Estimated Time of Arrival (ETA)	The predicted time that it will take a user to travel to one of their frequent locations.
FAQ	Frequently asked questions that users may have and answers to it, enables users to better understand the application.
Nearby Amenities	Amenities such as food, malls and parks that are within 500 metres of the property.
Google Maps API	An API which enables users to use maps and the components/functions related to the maps.

Appendix B: Analysis Models

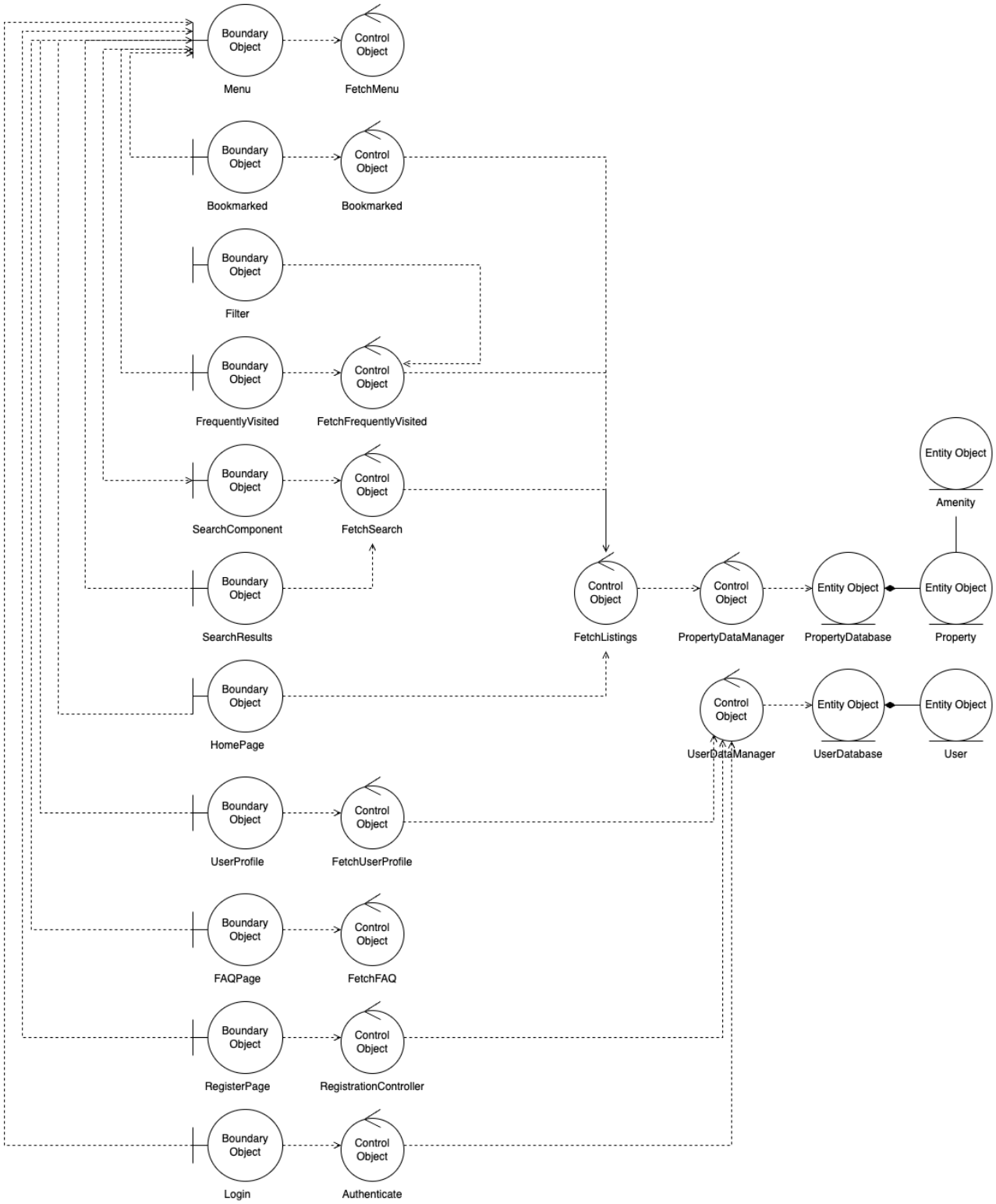
B.1 Use Case Model



B.2 UML Diagram

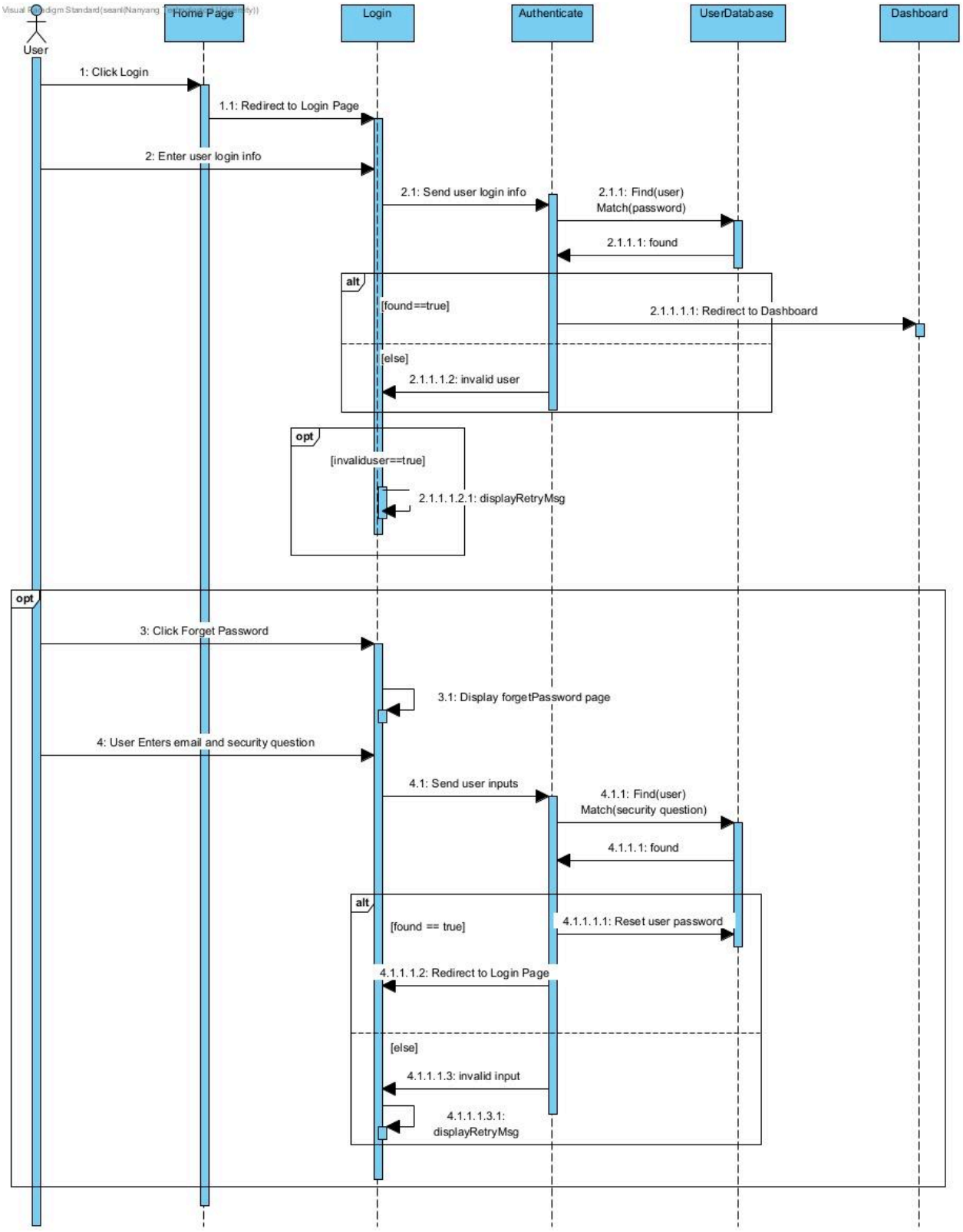


B.3 Entity, Control and Boundary Class Diagram

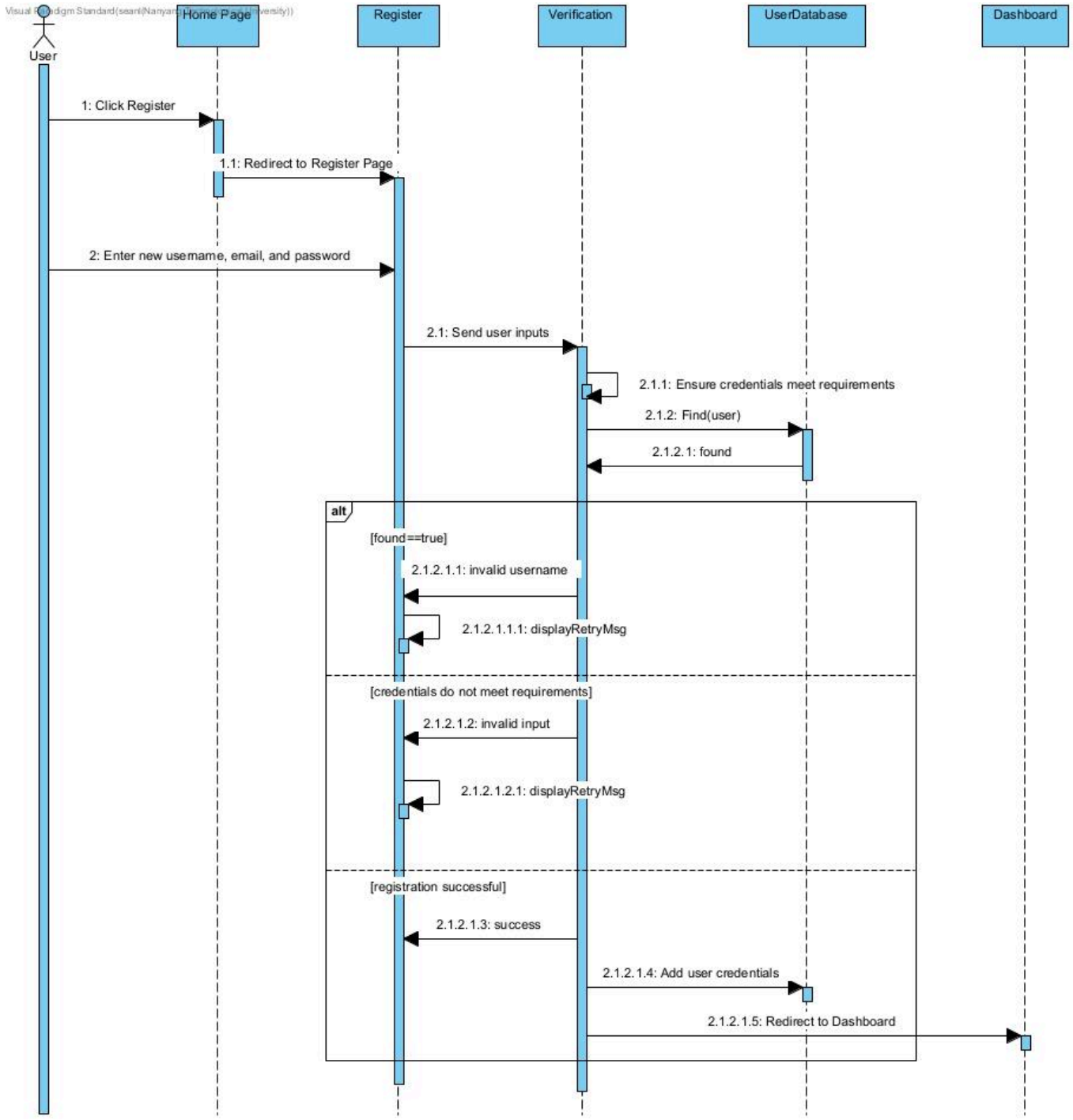


B.4 Sequence Diagram

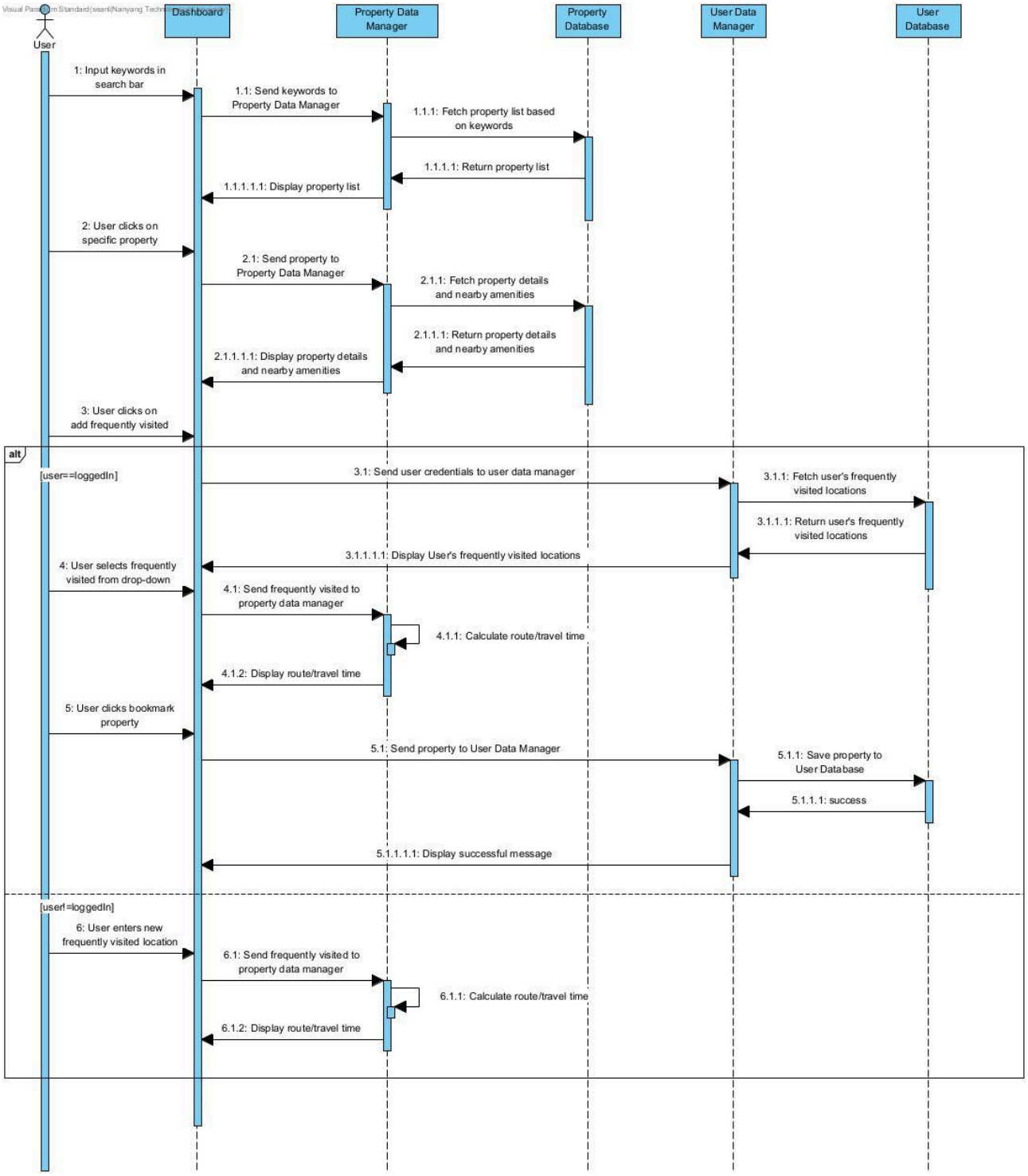
User Login



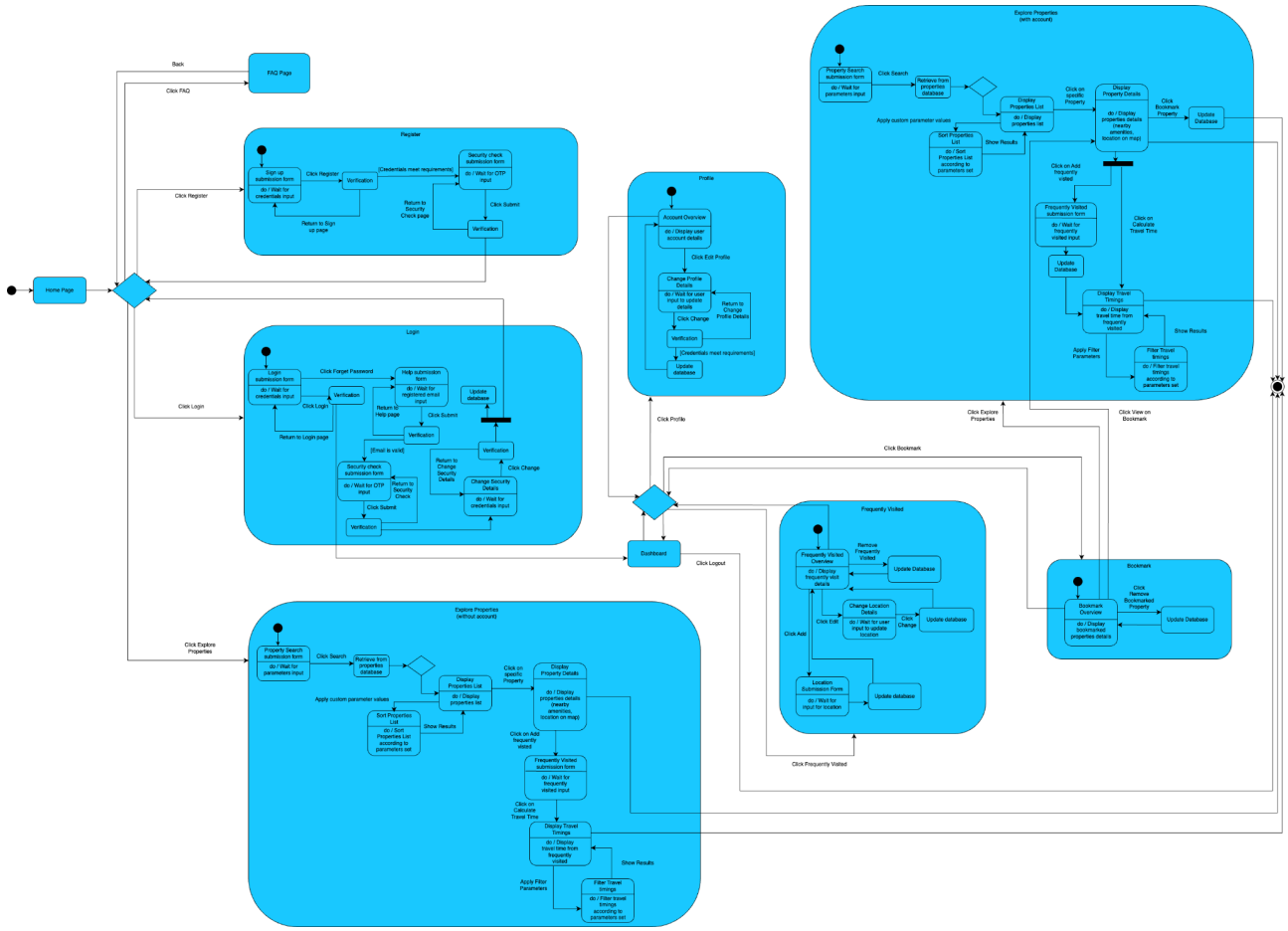
User Registration



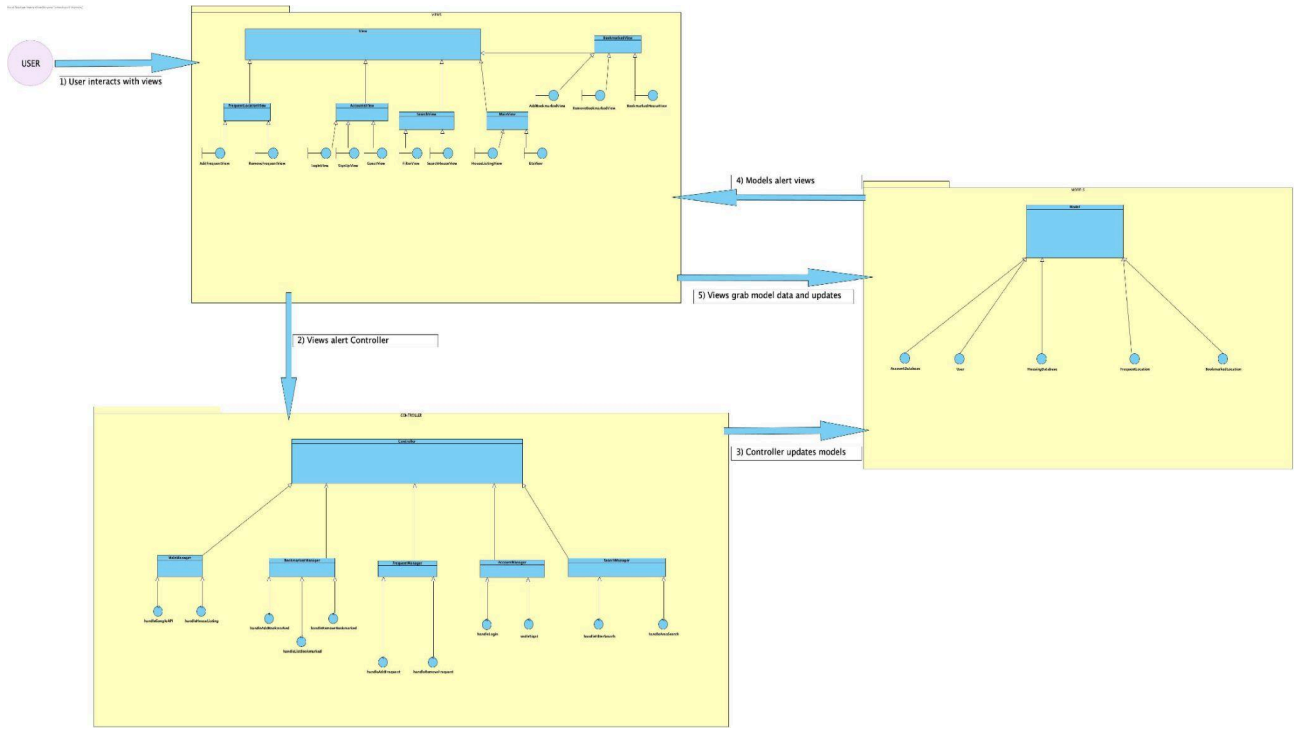
SearchHome Function



B.5 Initial Dialog Map



B.6 System Architecture 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.>