Software Requirements Specification

for

HomeFinder

Version 3.0 approved

Prepared by Chia Dion Yi, Wang Ruiyun, Murugappan Venkatesh, Ma Gaoyuan, Srishakti Neduchelian, Ian Koh Jin

Nanyang Technological University

3 April 2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Wang Ruiyun	3/4/2025	Introduction, overall description and appendix	1.0
Murugappan	3/4/2025	Use Case Descriptions	1.0
Shakti	3/4/2025	Non-functional requirements	1.0
Ian Koh Jin	5/4/2025	Product Perspective	1.0
Murugappan	5/4/2025	Functional Requirements	1.0
Ian Koh Jin	5/4/2025	Product Functions	1.0
Ian Koh Jin	6/4/2025	User Classes and Characteristics	1.0
Murugappan	8/4/2025	Revised Use Case Descriptions	2.0
Dion	9/4/2025	Added External Interface Requirements	3.0
Murugappan	10/04/2025	Added 2 use cases	3.0
Murugappan	12/04/.2025	Revised functional requirements and added 2 functional requirements for 2 new use cases	3.0
Shakti	12/04/2025	Revised Non-functional requirements and added other requirements	3.0
Shakti	13/04/2025	Final revision and amendments to Functional and Non-functional requirements	3.0
Dion	13/04/2025	Revised External Interface Requirements	3.0
Dion	13/04/2025	Finalised External Interface Requirements	3.0

1. Introduction

1.1 Purpose

This software Requirements Specification (SRS) document serves to provide a comprehensive and detailed description of the requirements and specifications for the HomeFinder software application. It will serve as the foundation for the design, development, testing, and implementation of the HomeFinder system.

The primary aims of the HomeFinder web application are as follows:

1. Viewing and Exploring Location: Provide information on the 55 planning districts in Singapore, with details of the district on 5 categories: Resale Price, Crime Rate, Nearby schools, malls, and public transport.
2. Comparing Locations: Provide relevant district information to users side by side for comparison.
3. Favourite Location: Providing notification when the data of a saved location for a user has changed.

This SRS will define the functional and non-functional requirements, as well as the system architecture, user interfaces, and interfaces, and other technical specifications essential for the successful implementation of the HomeFinder application. The information contained in this document will ensure that all stakeholders have a clear understanding of the system's intended capabilities, constraints, and design.

The intended audience for this SRS includes the project sponsor, project manager, software developers, quality assurance team, and any other key stakeholders involved in the HomeFinder project.

1.2 Document Conventions

This SRS document will adhere to the standard typographical conventions for clarity and consistency. **Bold text** is used to indicate GUI elements, user inputs, or to emphasize key points. *Italicized text* highlights newly introduced terms, important notes, or areas that require special attention. Monospace font is used for source code, API endpoints, and database queries. Unless specified otherwise, the priorities assigned to high-level requirements are carried over to their corresponding detailed requirements.

Font: Times New Roman
Colour: Black
Content: Font Size 12
Heading 1: Font Size 18
Heading 2: Font Size 14
Spacing: Single line spaced

Further conventions on special terms used throughout this document are described in Appendix A: Data Dictionary.

1.3 Intended Audience and Reading Suggestions

This document is primarily intended for the **Product Manager** overseeing the development and maintenance of HomeFinder. It also provides critical context for **Software Engineers** involved in implementation.

Guidance for different roles:

Product Managers should:
1. Begin with 2.2 Product Functions to understand the core features.
2. Review 3.1 User Interfaces for design principles.
3. Study 4. System Features for detailed capability breakdown of HomeFinder.
4. Refer to Appendix B: Analysis Models for system visualizations.

Software Engineers should:
1. Focus on 4. System Features to scope individual functionalities (e.g., district comparison, favorites).
2. Update 2.6 User Documentation and 4. System Features as the application develops.

1.4 Product Scope

Existing platforms like PropertyGuru or 99.co provide district-level housing data focusing primarily on resale prices but lack comprehensive and integrated comparisons of key livability metrics (e.g., crime rates, nearby amenities) or personalised recommendations. Users must manually compile data from disparate sources, leading to inefficiencies in decision-making.

Existing platforms like PropertyGuru or 99.co provide district-level housing data but lack integrated comparisons of key livability metrics (e.g., crime rates, school proximity) or personalized recommendations. Users must manually compile data from disparate sources, leading to inefficiencies in decision-making. Hence, there is a lack of a personalized and comprehensive website that assists new couples in finding suitable homes based on their individual preferences, needs, and budget constraints. With all these concerns in mind, we aim to produce an all-in-one web application that makes finding the ideal neighbourhood convenient.

HomeFinder addresses this gap by:

- Centralizing Data: Aggregating real-time metrics (resale prices, crime rates, amenities) from data.gov.sg APIs.

- Enabling Comparisons: Allowing side-by-side analysis of districts via an intuitive interface.

- Personalizing Insights: Letting users set preferences and rank the category by order of importance, as well as allowing them to save favorites and receive notifications on changes (e.g., price drops).

For the frontend, we used a React-based UI with Mapbox for interactive district maps. For the backend, we used a Flask server to handle user accounts, preferences, and API calls. For our data, we used Government APIs (HDB resale prices, crime statistics, school/mall/MRT locations) and Mapbox GL JS for geospatial rendering.

1.5 References

[1] 'IEEE Recommended Practice for Software Requirements Specifications', IEEE Standards Association. Accessed: Apr. 13, 2025. [Online]. Available: https://standards.ieee.org/ieee/830/1222/

2. Overall Description

2.1 Product Perspective

The HomeFinder web application is a new, self-contained product designed to help users find suitable housing in Singapore. It provides a complete solution through its own authentication service, user interface, and housing recommendation system. Users can create an account (or sign in if they already have one), explore a wide range of housing options, and compare them based on their specific constraints to select the best possible home.

To deliver accurate, real-time information, the application integrates with several external APIs:

- data.gov.sg
    - Retrieves up-to-date metrics about housing in various districts, including HDB resale prices, districts of kindergartens, MRT stations, malls, and crime data.
- Mapbox
    - Renders an interactive map on the client side to allow users to visually explore housing districts.

The following diagram (Figure 1) illustrates how these external APIs are integrated into the overall system architecture.

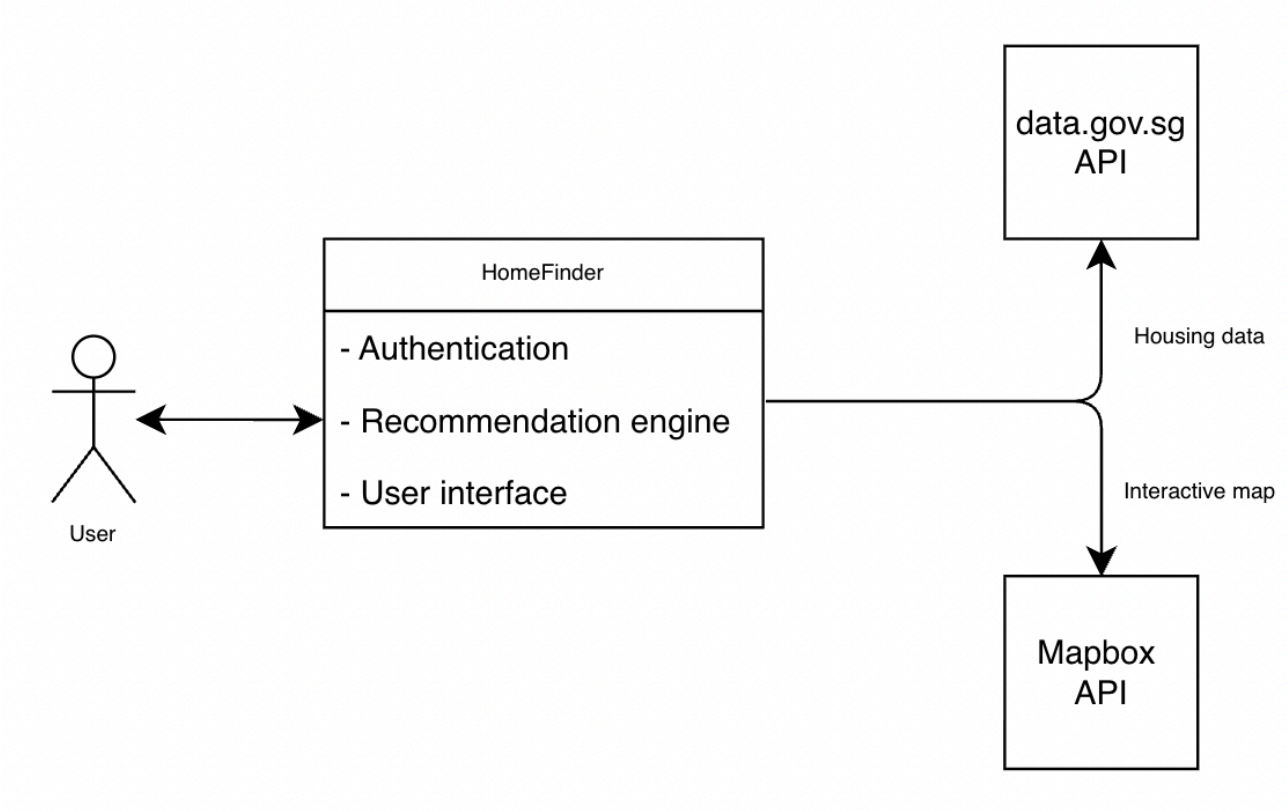


Figure 1. Product perspective illustration

The system architecture diagram below (Figure 2) maps out the entire HomeFinder system, showing both hardware and software components and how everything connects.

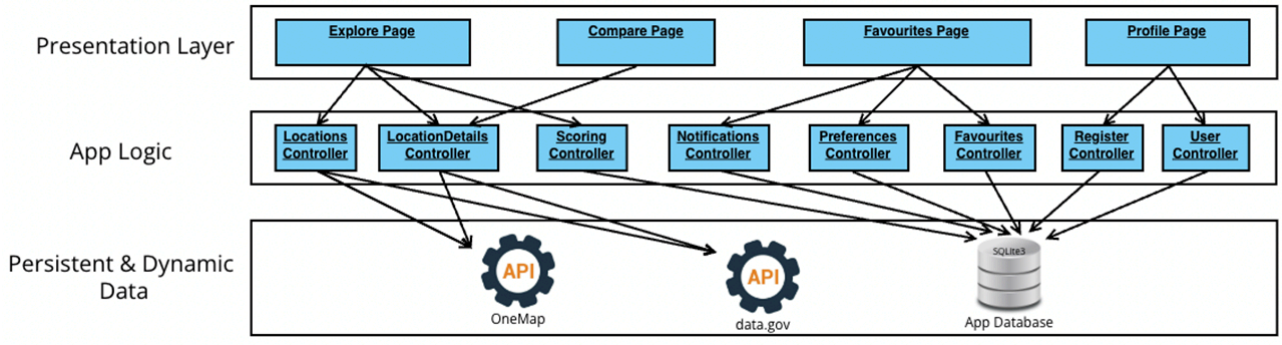


Figure 2. System architecture diagram

2.2 Product Functions

The HomeFinder web application is designed to empower potential homebuyers with comprehensive location evaluation by integrating real-time district data, comparative analytics, and

personalised recommendation features. Below is a high-level summary of the product's core functions:

2.2.1 Account Management

1. User Registration & Authentication
    - Create account via email/password (Figure 3)
    - Log in/Log out securely (Figure 4)
    - Delete accounts (with data purge confirmation)

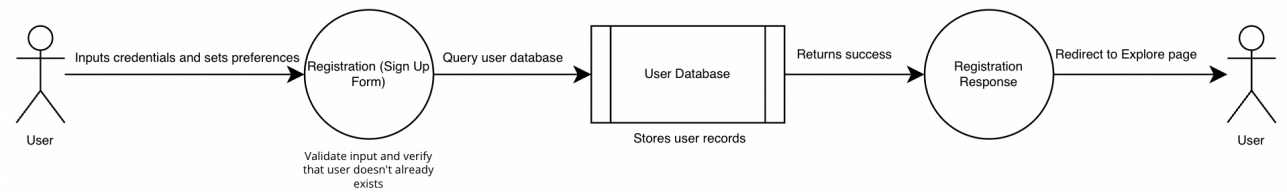


Figure 3. User creates an account under the profile page

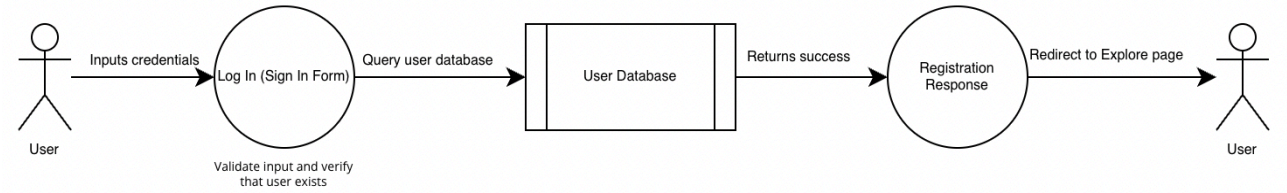


Figure 4. User has an existing account and signs in under the profile page

2. Preference Customisation
    - Set/update preferred value for the 5 categories – Resale price, Crime rate, Nearby schools, malls, and public transport
    - Set/update priority ranking for the 5 categories
    - Store preferences for “For You” recommendations

2.2.2 View and Explore Districts

1. An interactive map that visualises all 55 Singapore districts.
2. Provides a filtered view of the map with the following filter buttons: Resale Price, Crime Rate, Schools, Public Transport, Shopping Malls, For You (personal preferences when the user registers an account)
3. Supports searching for the district names and zooms in to selected districts on the map.
4. Provide the details of the district selected (information on the 5 categories) in the Compare page, and allow for side-by-side comparison of 2 selected districts.

2.2.3 Favourites Management

1. Allows users to save districts to the favourites page (user has to have an existing account)
2. Alerts users with notifications when information in the district has changed

2.3 User Classes and Characteristics

2.3.1 Primary User Class

- **First-Time Homebuyers**

Aspect	**Description**
Frequency of Use	High. Usage will be frequent, especially during home search period.
Subset of Product Functions Used	All Primarily uses most of the core features (explore, filter, and compare of district functions) and will likely create an account to save districts to the favourites page for future reference as they search for their ideal home.
Technical Expertise	Comfortable with web apps; may lack housing market knowledge. Varying levels of technical expertise, but can be generally assumed to be above average in terms of tech-savviness (younger demographics are more familiar with technology).
Age	21-35
Characteristics	- Young couples applying for BTO flats or resale HDBs - Reliant on data-driven decisions

2.3.2 Secondary User Classes

- **Real estate agents (or other professions related to the housing industry)**

Aspect	**Description**
Frequency of Use	Moderate to High. Usage will be as per client's needs as they can leverage the application as a tool to supplement their listings and provide data-driven recommendations to their clients.
Subset of Product Functions Used	All Primarily uses most of the core features. Might create own account, but will most likely

	encourage clients to do so instead, as each account only has one set of preferences at any given point in time. This is a limitation for a real estate agent serving multiple clients simultaneously, as each client has their own set of unique preferences.
Technical Expertise	Comfortable with web apps**.** Varying levels of technical expertise can be generally assumed to have more contextual knowledge and thus expect more advanced search or filtering capabilities.
Age	28-60
Characteristics	- Uses HomeFinder to supplement client consultations - Requires accurate, up-to-date data for credibility - Limited by single-account preference settings (can't manage multiple client profiles)

- **Property Investors**

Aspect	**Description**
Frequency of Use	Low to Moderate
Subset of Product Functions Used	All. However, they may focus more on viewing details, price trends, and comparing locations.
Technical Expertise	Comfortable with web apps; familiar with market information. High (data-savvy)
Age	30-65
Characteristics	- Focused on ROI metrics (e.g., 10-year price growth) - Less interested in livability filters

	(schools/malls)

2.4 Operating Environment

The production and development environment of HomeFinder web application will be covered in this section:

Hardware platform
Server-side
- If cost is not a major issue, on-premise servers would be used. If funding is tight, hosting this web application on cloud services such as Azure or AWS is also an option. A hybrid approach of on-premise and cloud services can also be considered, as this is the industry standard.

Client-side
- Designed for laptops and desktops primarily. Adding tablets and smartphones is a future consideration.

Operating system
Server
- Primarily deployed on Linux-based operating systems for production, but can also function on Windows Server environments if the appropriate dependencies are installed and managed.

Client
- Any operating system can be used as long as it supports modern web browsers. Examples of modern web browsers include Google Chrome, Mozilla Firefox, Safari and Microsoft Edge.

External dependencies
- Data.gov.sg API
  - Provides real-time housing metrics from the Singapore government
- OneMap API
  - Provides real-time location geodata, school, transport and mall data from Singapore Land Authority.
- Mapbox API
  - Renders an interactive map on the client-side using Mapbox GL
- Frontend
  - Developed with React, utilising libraries such as react-router and react-beautiful-dnd
- Backend
  - Developed using Flask

Development Environment

Development	**Description**

Environment	
Version Control	**Git and GitHub** are used for version management and collaborative development. The repositories are maintained both locally and on GitHub, enabling seamless team collaboration and continuous integration/deployment (CI/CD) workflows**.**
Frontend	**React.js** is an open-source JavaScript library used to build HomeFinder's dynamic interface. Its component-based architecture enables: - Reusable UI elements for district cards, filters, and comparison tools Integration with: - Mapbox GL JS for interactive Singapore district maps - React Router for seamless page navigation - Tailwind CSS for responsive styling Key Advantages: - Real-time data display for housing metrics - Drag-and-drop functionality for district comparisons
Backend	**Flask** serves as HomeFinder's lightweight backend framework, providing: 1. RESTful API endpoints for: - User authentication and profile management - Data processing (e.g., district scoring based on preferences) 2. Integration with external APIs: - data.gov.sg (HDB prices, crime rates, amenities) - Mapbox (geocoding and distance calculations) Advantages:

	- Minimalist Design: Rapid development with customizable components - Scalability: Supports transition from SQLite (development) to PostgreSQL (production)
Database	**SQLite** is a lightweight, self-contained relational database management system that operates without a separate server process. Its embedded architecture makes it particularly well-suited for HomeFinder's development environment, where simplicity and rapid iteration are prioritized. Serverless Design: The entire database exists as a single file within the application directory, eliminating complex setup procedures Zero-Configuration: Requires no administration, matching HomeFinder's need for a low-maintenance development database Full SQL Support: Provides complete relational database capabilities for managing: - User profiles and authentication data - Saved favorite districts and preference settings - Cached API responses from data.gov.sg Technical Advantages: 1. ACID Compliance: Guarantees reliable transactions for critical operations like: - User account creation - District watchlist updates - Preference changes 2. Cross-Platform Portability: - Single database file works identically across Windows, macOS, and Linux development environments - Enables easy team collaboration and version control integration

	3. Resource Efficiency: - Minimal memory footprint (under 1MB) - Ideal for caching frequently accessed district metrics: - HDB resale price histories - Crime statistics - School malls and public transport locations Implementation in HomeFinder: During development, SQLite serves as the primary datastore for: - User session management - Temporary storage of district ranking calculations - Prototyping data models before production deployment

2.5 Design and Implementation Constraints

This section outlines the constraints that will impact the design and implementation of the system. These constraints may arise from organizational, technical, or regulatory factors.

Regulatory and Corporate Policies

- All data must be stored and transmitted securely.

Hardware Limitations

- The application must be optimized for use on devices with a minimum of 2GB RAM dual-core CPU.

Interfaces to Other Systems

- The system must integrate with data.gov.sg APIs.
- RESTful APIs must be used for all inter-service communication.

Technology Stack Constraints

- Frontend must be implemented using React.js with TypeScript.
- Backend must be developed using Flask.
- Database will use SQLite. No other database types may be used.

Programming and Design Standards

- Unit and integration testing is mandatory, with at least 80% test coverage.

Language and Localization

- The system must support English.

Maintenance Requirements

- The system must be designed in a modular way to allow future maintainers (internal team) to easily modify or extend the codebase.
- All documentation must be included inline and in external README or Wiki format.

2.6 User Documentation

The following user documentation components will be delivered alongside the software to ensure effective onboarding and usage:

2.6.1 User Manual (For User)

The User Manual is a comprehensive guide for end-users to use HomeFinder's core features. It can be accessed at the ReadMe file.

2.7 Assumptions and Dependencies

Assumptions

User Environment

- It is assumed that end-users will have access to a modern web browser (e.g., Chrome, Firefox, Safari, Edge) with TypeScript enabled.
- Users will have a stable internet connection during system usage.

Technology Stack Stability

- The core technologies (e.g., React.js, Flask, SQLite) will remain stable and continue to be maintained throughout the development lifecycle.
- Required SDKs/APIs (e.g. data.gov.sg, etc.) will not undergo breaking changes during the project timeline.

Client-Side Device Capabilities

- It is assumed that users' devices meet the minimum system requirements for running the application (e.g., sufficient memory and processing power).

Team Access and Tools

- The development team will have continued access to development tools, cloud services, and repository hosting (e.g., GitHub, AWS, etc.).

Dependencies

Third-Party Libraries/Frameworks

- The project depends on the availability and functionality of the following third-party components:
  - React.js and tailwind CSS for frontend development
  - Axios or Fast API for HTTP requests
  - SQLite Database

External APIs

- The system depends on the availability of external APIs, such as:
  - Mapping or location services (e.g. Onemap Singapore, mapbox)
  - External data APIs (e.g. data.gov.sg)

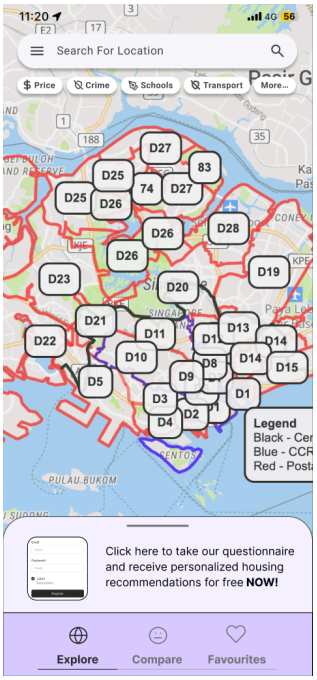
Hosting and Infrastructure

- Deployment depends on the availability of infrastructure (e.g., AWS, Vercel, or internal hosting resources).
- Downtime or policy changes from these providers may impact delivery or uptime.

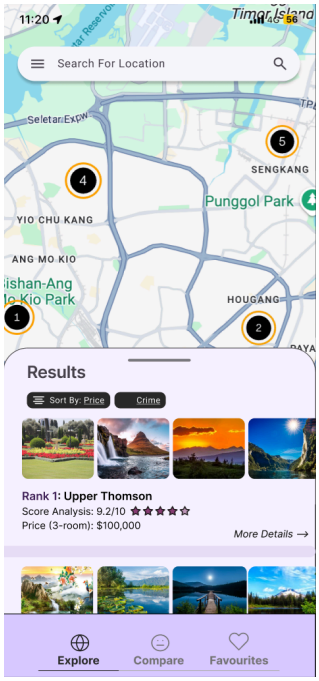
3. External Interface Requirements

3.1 UI Mockups

3.1.1 Explore Page

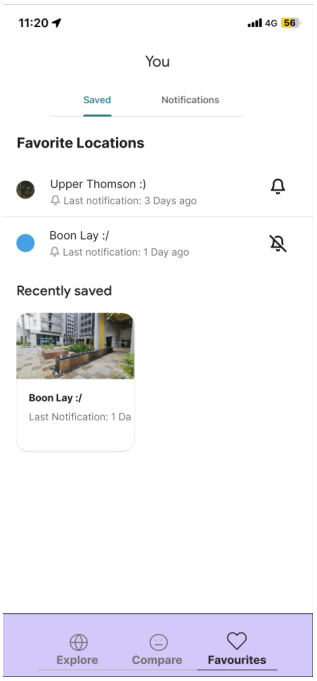


3.1.1.1 **Explore Page (Explore Location)**



3.1.2 **Compare Page**

3.1.3 **Favourites Page**

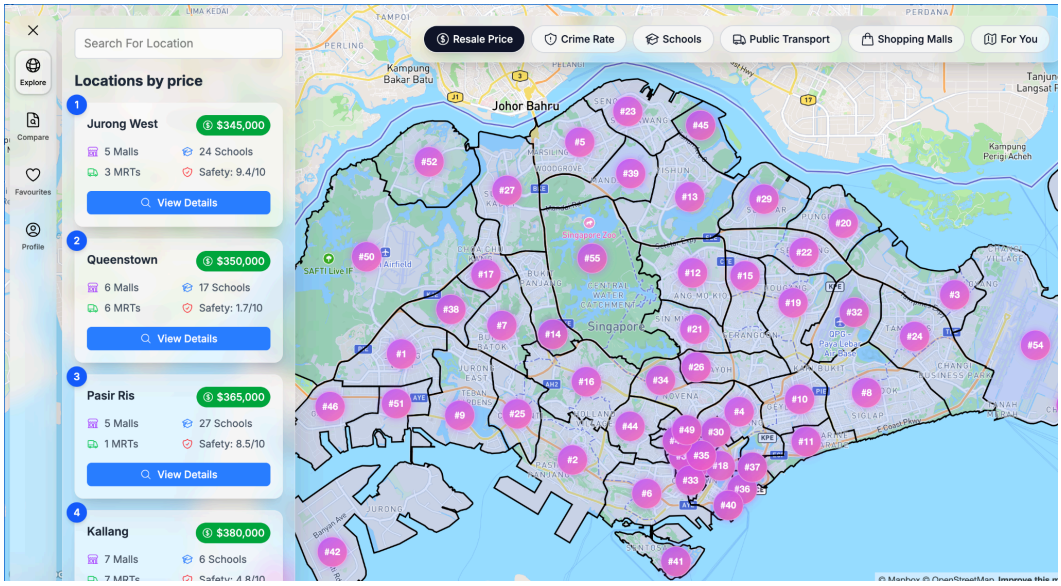


3.1.4 **Profile Page (Register Preferences)**

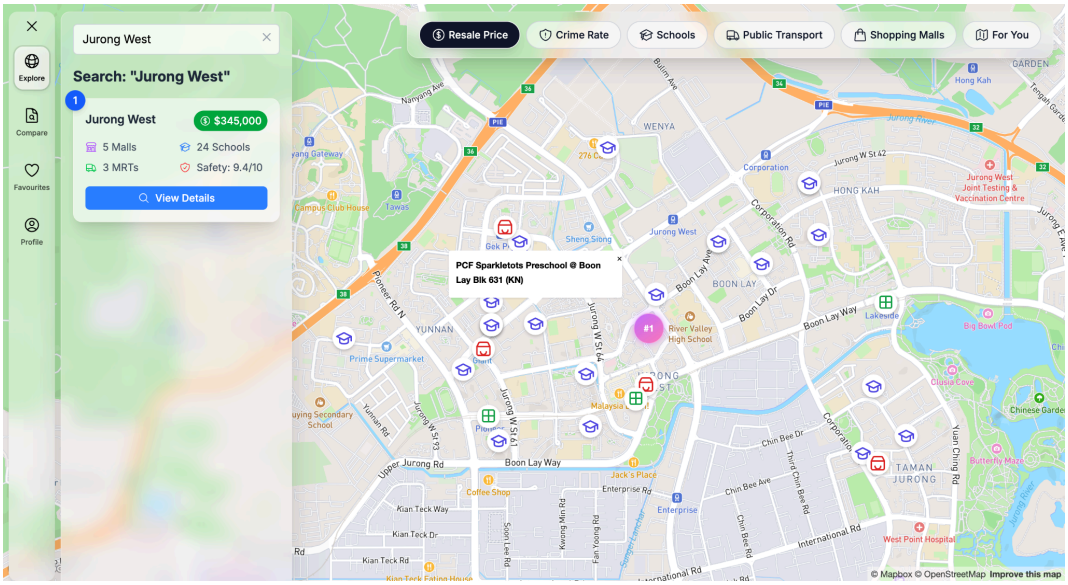
3.2 User Interfaces

3.2.1 Explore Page

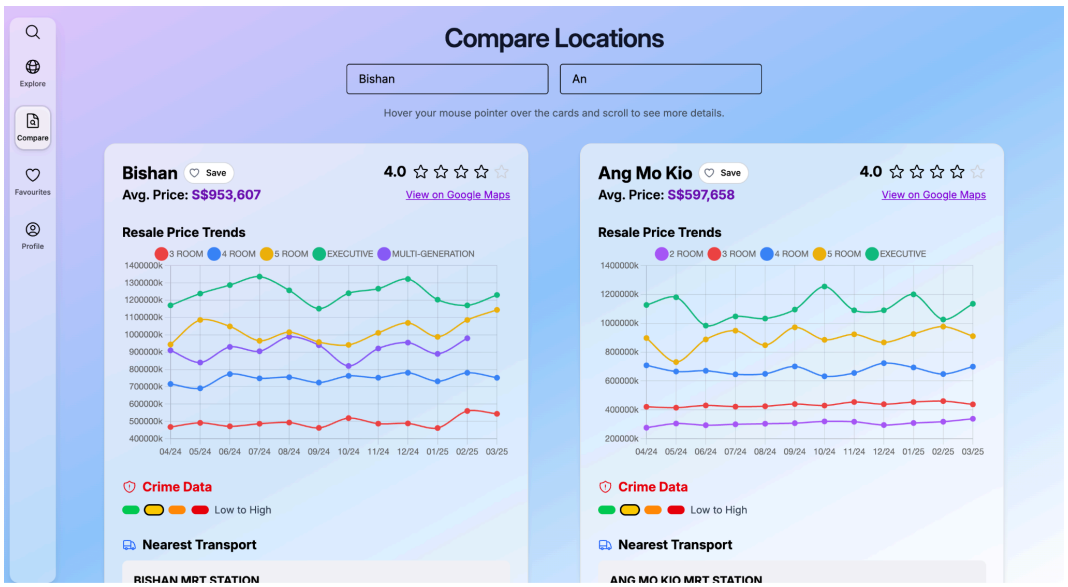
3.2.1.1 Explore Page (Overview)



3.2.1.2 **Explore Page (Explore Location)**

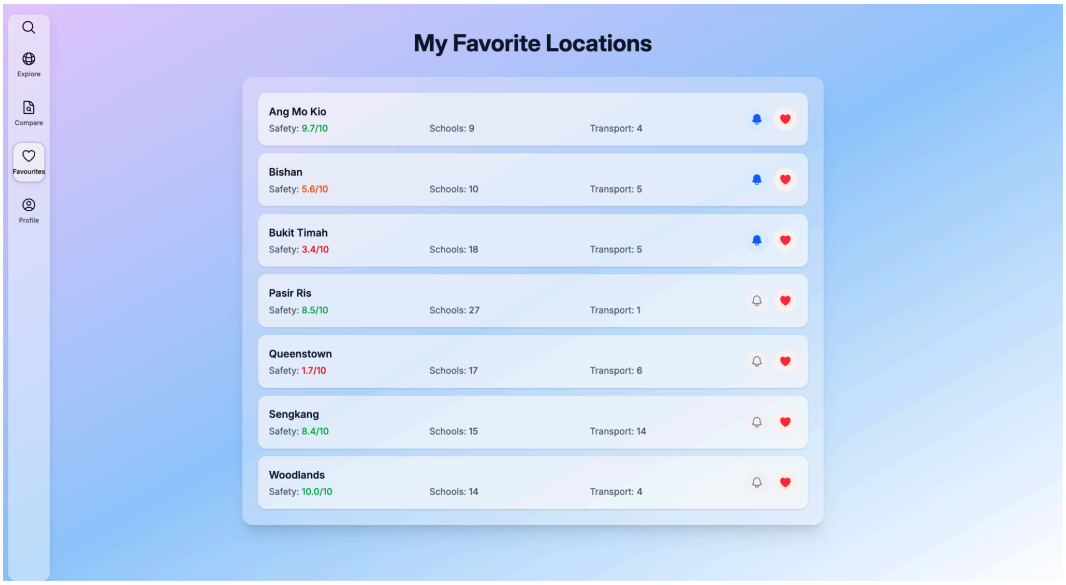


3.2.2 **Compare Page**



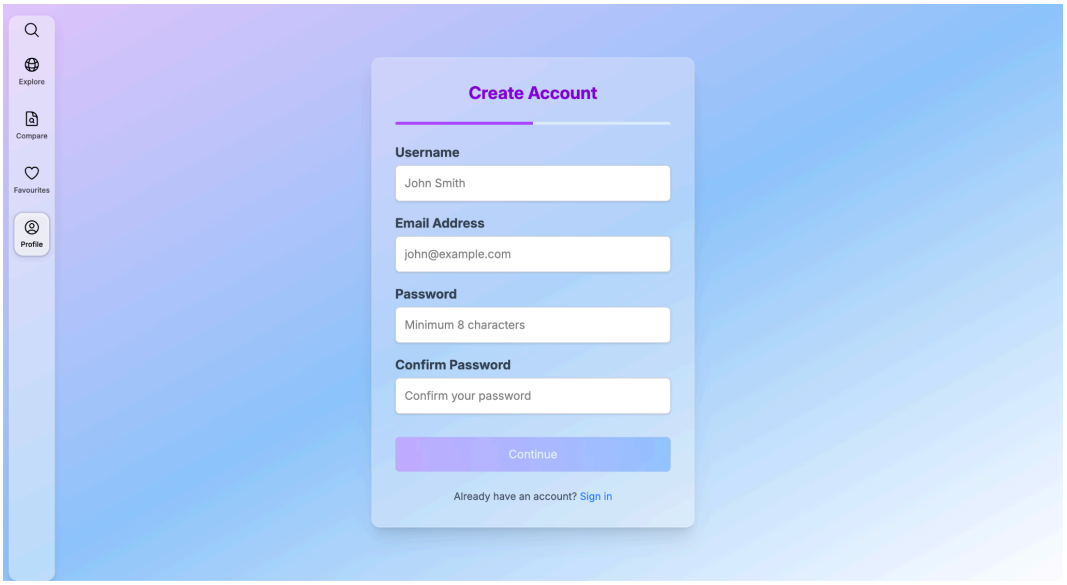
3.2.2.1 **Compare Page (Expanded)**

3.2.3 **Favourites Page**

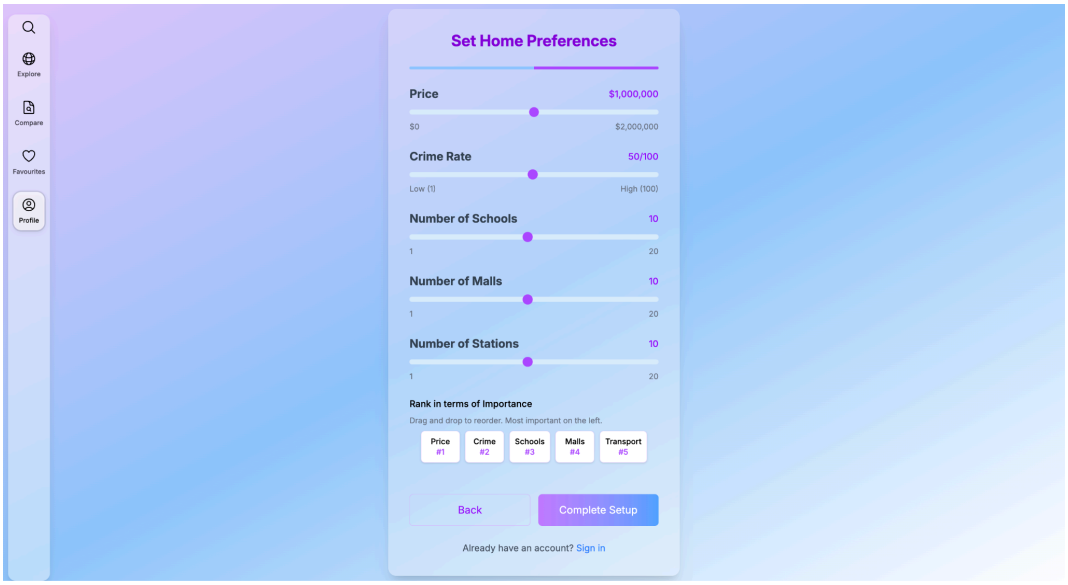


3.2.4 **Profile Page**

3.2.4.1 **Profile Page (Sign UP)**



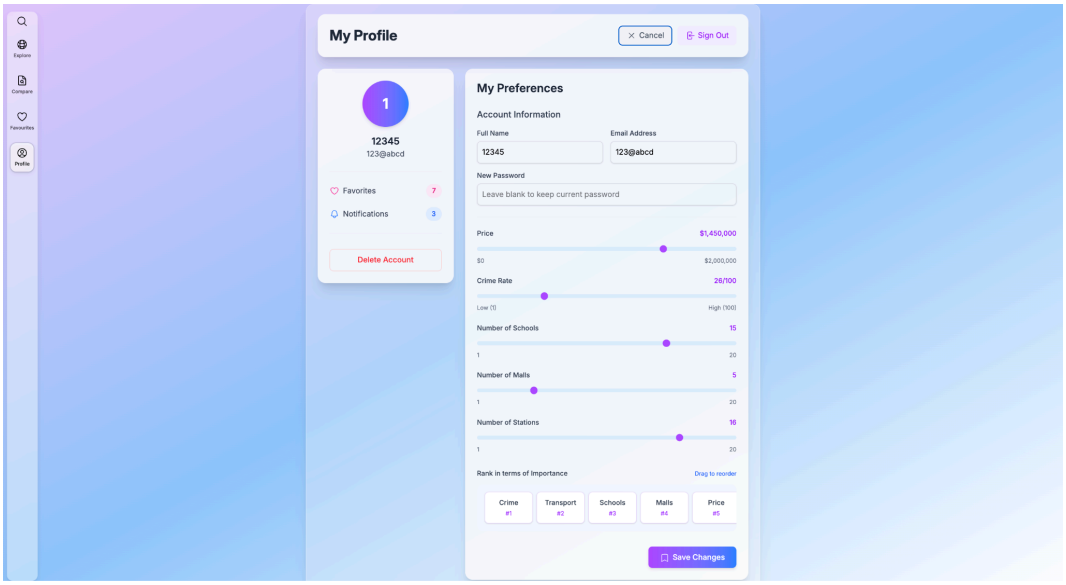
3.2.4.2 **Profile Page (Register Preferences)**



3.2.4.3 **Profile Page (Sign IN)**

3.2.4.4 **Profile Page (User Profile, Signed IN)**

3.2.4.5 **Profile Page (User Profile, Edit Preferences)**



3.3 Hardware Interfaces

This section describes the hardware interfaces required for the HomeFinder web application to perform effectively and reliably.

Requirements	Specifications
Operating System	- For Windows: Windows 8 and later versions. - For macOS : macOS 10.10 and later version. - For Linux: Ubuntu, Debian, Fedora
Network Connection	Wireless Network Interface Card (WNIC) or a modern chip with a cellular modem.
Interaction	A functional touchpad or mouse to navigate through the website.

3.4 Software Interfaces

OS: macOS 10.10
Tools : React JS, Flask, Python, SQLite

APIs:

- Mapbox API

  The Mapbox API provides developers with a powerful suite of tools to integrate interactive maps, geolocation, and navigation features into their applications. It offers access to high-quality, customizable map data and services, enabling developers to create dynamic user experiences tailored to specific geographic needs.

- OneMap API

  The OneMap API is Singapore's authoritative national map API service, developed by the Singapore Land Authority (SLA). It offers a wide range of geospatial data and services, including location search, routing, reverse geocoding, and thematic layers such as MRT stations, schools, shopping malls, and planning areas. The API provides developers with access to reliable and up-to-date geographic information for integration into web and mobile applications.

- Data.Gov API

  The Data.gov.sg API is a public data platform developed by the Singapore Government that provides programmatic access to a wide range of datasets from various government agencies. It enables developers to access timely and authoritative data for use in data analytics, visualization, and application development. The API covers domains such as population, housing, transport, education, and public safety.

4. System Features

4.1 User Registration

4.1.1 Description and Priority

User registration is the process of registering an account for a new user using a valid email address, username and password.

Overall Priority	Description
High	A user must have a registered account in order to access the services the app provides. A new user will need to register an account using a valid email, username and password of minimum 8 characters in length.

4.1.2 Stimulus/Response Sequences

Use Case ID:	001		
Use Case Name:	Registration		
Created By:	Ian	Last Updated By:	Dion
Date Created:	12/03/2025	Date Last Updated:	02/04/2025

Actor:	1. User 2. Data APIs
Description:	A new user creates an account to access the application.
Preconditions:	The user is using the application for the first time with an unregistered email.
Postconditions:	The user's username, password and email address are registered into the database, and the user can log in.
Priority:	High
Frequency of Use:	Once per user

Flow of Events:	1. The user is directed to the registration page after they click the ‘Profile’ tab on the Navigation Bar. 2. The system prompts the user to input an email, username, password of at least 8 characters in length and password confirmation. 3. The system validates the username. 4. The system validates the email. 5. The system validates the password. 6. New account created successfully. 7. A new page appears, prompting the user to set his/her home preferences for price, crime rate, number of schools, number of malls and number of stations and also rank the preferences in terms of importance. 8. User keys in the information and clicks on the “Complete Setup” button. 9. System displays the user’s name, email, his/her preferences, favourites and notifications.
Alternative Flows:	AF-S2: User does not fill in all sections. The system will disable the 'Continue' button when there is at least one section not filled up, preventing the user from proceeding to register. AF-S3: Username already registered. The system will display an error message notifying the user the username is already used and prompt the user to register using another username. AF-S4: Email already registered. The system will display an error message notifying the user the email is already used and prompt the user to register using another email. AF-S5: Password is less than 8 characters long.

	The system will disable the 'Continue' button when the password is less than 8 characters long, preventing the user from proceeding to register. AF-S8: User can click on the “Back” button. User is redirected to the registration page.
Exceptions:	NA
Includes:	Password Confirmation
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

4.1.3 Functional Requirements

1. User Registration
    1.1. The user must be able to register for an account if they do not have an account.
        1.1.1. The user must set their account username during registration.
        1.1.2. The user must set their account email during registration.
        1.1.3. The user must set their password during registration.
            1.1.3.1. The user’s password must be at least 8 characters..
        1.1.4. The user must confirm their password during registration.

4.2 User Login

4.2.1 Description and Priority

User login is a process where a user will login to their existing account using their valid email and password.

Overall Priority	Description

High	The user needs to be verified to be a registered user before gaining access to the website's services. A user who types in an invalid email and/or password will be denied access.

4.2.2 Stimulus/Response Sequences

Use Case ID:	002		
Use Case Name:	Login		
Created By:	Dion	Last Updated By:	Ruiyun
Date Created:	12/03/2025	Date Last Updated:	02/04/2025

Actor:	1. User 2. Database
Description:	Users log into the website using their registered email and password.
Preconditions:	The user must be registered with an account.
Postconditions:	The user is authenticated and granted access to the website.
Priority:	High
Frequency of Use:	Once per use of application

Flow of Events:	1. The user enters their email or username and password. 2. The system checks the entered email or username against the stored email or username in the database. 3. The system checks the entered password against the stored password in the database. 4. The user is logged in and directed to the Explore page.
Alternative Flows:	AF-2: User does not fill in all sections. The system will disable the 'Sign in' button, and user is unable to log in. AF-3 : Invalid email The system displays an error message stating that the user is not verified. AF-4 : Invalid password The system displays an error message stating that the user is not verified.
Exceptions:	NA
Includes:	NA
Special Requirements:	Checking the database for user information.
Assumptions:	NA
Notes and Issues:	NA

4.2.3 Functional Requirements

1. The user must be able to log in to their account.
   1.1. The user must be able to enter their email during login.
   1.2. The user must be able to enter their password during login.

4.3 Viewing and Exploring districts

4.3.1 Description and Priority

Users can click on the bubble representing each district and view the amenities in the district, like schools and MRT stations.

Overall Priority	Description
High	The system will display all the districts in Singapore, each represented by a bubble. It displays the various schools, MRTs, malls, the average price of the houses in the district and the overall score calculated based on the user's preferences. This is an essential feature of the app since it provides the necessary information for the user to choose a house in his/her preferred district.

4.3.2 Stimulus/Response Sequences

Use Case ID:	003		
Use Case Name:	Viewing and Exploring districts		
Created By:	Dion	Last Updated By:	Dion
Date Created:	17/03/2025	Date Last Updated:	01/04/2025

Actors	User (Initiating), Data APIs
Description	Users can view districts on the map or the sidebar and explore detailed information about a chosen district.

PreConditions	1. User selects a district from the map or list of locations in the sidebar.
PostConditions	1. User is rerouted to the Compare Page and detailed information about the district is displayed.
Priority	High
Frequency of Use	High – Used when users explore specific districts.
Flow of Events	1. User selects a district from the map or searches a district via the search bar. 2. User clicks on the "View Detail" button. 3. System fetches detailed information, including: ○ Average price of houses in the district. ○ Resale price trends of each type of house over the past 12 months(displayed in monthly format). ○ Neighbourhood crime rate ○ Nearest MRT stations ○ Nearest schools ○ Nearest malls ○ Past Major Crimes 4. User views the details and optionally add the district to their favourites by clicking on the "Save" button.
Alternative Flows	AF-S2: Data is unavailable for some metrics.
Exceptions	1. API failure or timeout for fetching detailed data.

	○ System retries or shows a fallback summary.
Includes	Data visualization for scores and trends.
Special Requirements	Easy-to-navigate interface for viewing district details.
Assumptions	NA
Notes and Issues	NA

4.3.3 Functional Requirements

1. The system must display a map of all districts represented as bubbles.
    1.1. The system must allow users to select a district by clicking on a district bubble or from the sidebar list.
    1.2. Upon selecting a district, the system must navigate the user to the Compare Page.
    1.3. The system must display the district information.
        1.3.1. The system must display the average resale price of houses in the district.
        1.3.2. The system must display the resale price trends of each type of house over the past 12 months in a monthly format.
        1.3.3. The system must display the neighbourhood crime rate.
        1.3.4. The system must display the nearest MRT stations.
        1.3.5. The system must display the nearest schools.
        1.3.6. The system must display the nearest malls.
        1.3.7. The system must display past major crimes.
        1.3.8. If some district information is unavailable, the system must not display the corresponding district information.
        1.3.9. If there is an API failure or timeout for fetching district information, the system must retry or show a fallback summary.
    1.4. The system must provide a “Save” button to allow the user to add the district to their favourites.

4.4 Comparing districts

4.4.1 Description and Priority

A feature to compare selected districts side-by-side based on the 5 factors namely resale price trends and crime data. nearest transport options, nearest schools and nearest malls.

Overall Priority	Description
Medium	A user needs to enter the 2 districts that the user wants to compare based on the 5 categories. This is an optional yet highly useful feature for users to make more informed decisions about purchasing a house.

4.4.2 Stimulus/Response Sequences

Use Case ID:	004		
Use Case Name:	Comparing districts		
Created By:	Shakti	Last Updated By:	Dion
Date Created:	17/03/2025	Date Last Updated:	05/04/2025

Actor	User (initial actor), Data APIs
Description	Users can compare shortlisted districts side by side for informed decision-making.
Preconditions	User enters 2 districts at the 2 search bars respectively or clicks on 2 district bubbles on the interactive map.

Postconditions	A comparison table with the average price of the houses, overall score, price chart showing the resale price trends over the past year, crime data bar with 4 categories, top 3 nearest MRTs, top 3 nearest schools and top 2 nearest malls is displayed.
Priority	Medium
Frequency of use	Medium – Used when users need to compare 2 districts.
Flow of events	1. User enters two districts in the 2 search bars or clicks on 2 district bubbles on the interactive map in the Explore page. 2. System generates a comparison table, showing: ○ Average Price of the houses in the district ○ Overall scores ○ Resale Price Chart ○ Crime Data with 4 levels ○ 3 nearest MRTs ○ 3 nearest schools ○ 2 nearest malls
Alternative flows	AF-S2: If the user changes or removes a district during comparison. 1. If the user removes one district, only the information for the other district is displayed. If the user changes district(s), the system dynamically updates to show the information of the respective district(s). AF-S2: If the district information is unavailable, 1. The system will not display any district information.

Exceptions	EX 1: If the user enters invalid names of districts 1. The system displays “Loading…” and is unable to show the information.
Includes	Dynamic comparison table generation
Special requirements	Clear, visually appealing table format.
Assumptions	Users can interpret the comparison metrics easily.
Notes and Issues	-

4.4.3 Functional Requirements

1. Users must be able to compare 2 districts.
    1.1. The system must allow the user to input 2 district names in separate search bars or click on 2 district bubbles on the map.
    1.2. The system must display a comparison table comparing 2 districts based on the district information.
        1.2.1. The system must display the average resale price of houses in the district for each district.
        1.2.2. The system must display the resale price trends of each type of house over the past 12 months in a monthly format for each district.
        1.2.3. The system must display the neighbourhood crime rate for each district.
        1.2.4. The system must display the nearest MRT stations for each district.
        1.2.5. The system must display the nearest schools for each district.
        1.2.6. The system must display the nearest malls for each district.
        1.2.7. The system must display the past major crimes for each district.
        1.2.8. The system must dynamically update the comparison table when the user changes one or both districts.
        1.2.9. If the user enters only one valid district, the system must display information for that district only.
        1.2.10. If the district information is unavailable, the system will not display any district information.
        1.2.11. If the user enters an invalid district name, the system must display a “Loading…” message and fail to load data.

4.5 Sorting and listing districts based on individual categories

4.5.1 Description and Priority

Users can view districts sorted based on a specific category.

Overall Priority	Description
High	A user can click the respective button for each category to view districts sorted based on a certain category. The system displays a bubble for each district, with each bubble displaying the rank of the district sorted in ascending order of the category. The system also lists the districts in ascending ranks.

4.5.2 Stimulus/Response Sequences

Use Case ID:	005		
Use Case Name:	Sorting and listing districts based on individual categories		
Created By:	Dion	Last Updated By:	Muru
Date Created:	15/03/2025	Date Last Updated:	04/4/2025

Actor	User (Initiating), Data APIs
Description	Users can view a list of districts sorted in ascending order based on specific categories such as resale price, crime rate, number of schools,

	number of malls and number of MRTs. This helps them to get category-specific information about each district for each category.
Preconditions	1. The system can access real estate pricing, crime statistics, and geographical data. 2. The user has selected a category to rank the districts.
Postconditions	1. The system displays the districts on the map ranked based on the chosen category and lists the districts.
Priority	High
Frequency of use	High – Users will use this feature multiple times throughout their home search.
Flow of events	1. The user navigates to the Explore page. 2. The system presents 5 buttons each labelled with a category as follows: • Resale price • Crime rate • Schools • Public Transport • Shopping Malls 3. The user clicks on the button with the respective category.

	4. The system retrieves district information for the selected category, ranks districts in the map and lists districts in ascending order of rank. 5. The user can then save shortlisted districts to their favourites for future reference or click on a particular district bubble to view more information about the district.
Alternative flows	•AF-S4 : The user changes filtering criteria(clicks on another category button) and re-runs the search.
Exceptions	NA
Includes	Filtering and sorting algorithms to refine results. • Integration with real estate, crime, and public infrastructure databases. • Interactive UI with list and map-based navigation.
Special requirements	The filtering interface should be user-friendly and responsive. • The system should handle large datasets efficiently. • The ranking algorithm should balance multiple criteria effectively.
Assumptions	Users want to refine their home search based on specific categories. • Reliable and relatively up-to-date information is available for real estate, crime rates, and amenities.

Notes and Issues	-

4.5.3 Functional Requirements

1.1 The system must display 6 category buttons, namely Resale price, Crime rate, Schools, Public Transport and Shopping Malls and one personalised button, “For You”.

1.1.1. When a category is selected, the system must retrieve and rank all districts based on that category.

1.1.2. The system must display each district as a bubble with its corresponding rank number on the map.

1.1.3. The system must list each district in ascending order of rank in the sidebar.

1.1.4. The system must display a summary of the district information.

1.1.4.1. The system must display the average resale price of houses in the district for each district.

1.1.4.2. The system must display the safety rate for each district.

1.1.4.3. The system must display the number of MRT stations for each district.

1.1.4.4. The system must display the number of schools for each district.

1.1.4.5. The system must display the number of malls for each district.

1.1.5. The system must allow users to click on a district bubble to view more information.

1.1.6. The system must allow users to click on the “View details” button in the sidebar for a district to view more information about it.

1.1.7. If a category is re-selected, the system must update and re-rank districts accordingly.

4.6 Adding district to Favourites

4.6.1 Description and Priority

A process where a user can add their preferred home to the favourites list.

Overall Priority	Description
Medium	A user can add a home they like. Users can then access the information for their favourite districts directly in the Favourites page, without needing to use a list-based or map-based search. Priority is low, considering that this is an optional feature, but it

	is highly useful and beneficial as it decreases users' search time for a particular district

4.6.2 Stimulus/Response Sequences

Use Case ID:	006		
Use Case Name:	Adding to Favourites		
Created By:	Ruiyun	Last Updated By:	Ruiyun
Date Created:	21/03/2025	Date Last Updated:	08/04/2025

Actor	User (initial actor), Data APIs
Description	Users can add districts to a watchlist of favorite districts.
Preconditions	1. The user has a stable internet connection. 2. The user has selected at least one district.
Postconditions	1. Users have added a district to the favourites list.
Priority	Medium
Frequency of use	Medium Users can opt to use this function for ease in finding the district again in the future.

Flow of events	1. User clicks on the district bubble in the map. 2. System redirects the map to the district and displays the district information in the search bar. 3. The user then clicks the “View Details” button to obtain more district information. 4. If the user likes the district, the user clicks on the “Save” button. 5. System saves the district added by the user to the server-side database, dynamically updates the button to “Saved” and displays the new district in the Favourites page on the UI.
Alternative flows	AF-S5: If the user clicks on the “Saved” button, the system updates its database and removes the district from the Favourites page
Exceptions	1. The user has not registered an account yet or has not logged in.
Includes	-
Special requirements	-
Assumptions	-
Notes and Issues	-

4.6.3 Functional Requirements

1. Adding district to Favourites

1.1. The system must provide a “Save” button for each district in the Compare page to add the district to favourites.

1.2. The system must update the button text to “Saved.”

1.2.1 The system must not update the button text and display an error if the user is not logged in or registered.

1.3. The system must add the saved district to the Favourites page on the UI.

1.4. If the “Saved” button is clicked, the system must remove the district from the favourites list.

4.7 Viewing favourite districts

4.7.1 Description and Priority

A process where a user can view their favourite districts.

Overall Priority	Description
Medium	A user can view his/her favourite districts. The user can click on the district, which provides detailed information.

4.7.2 Stimulus/Response Sequences

Use Case ID:	007		
Use Case Name:	Viewing favourite districts		
Created By:	Ruiyun	Last Updated By:	Dion
Date Created:	21/03/2025	Date Last Updated:	05/04/2025

Actor:	User, Data APIs
Description:	The user is provided a brief summary of information about their favourite district. User can view more information by clicking on the district.
Preconditions:	1. The user must have an account. 2. The user must be logged in.

	3. The user has added district(s) to the Favourites page
Postconditions:	The detailed information of the district is displayed
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. The user clicks on the Favourites page. 2. The user clicks on 1 district in the Favourites page 3. The system redirects to the Compare Page and shows detailed information about the district, including average price, resale price trends for each type of housing, crime rate, nearest transport, nearest schools, nearest malls and crime incidents
Alternative Flows:	AF-S2: If no favourite districts present 1. The user is not able to click on any district.
Exceptions:	-
Includes:	NA
Special Requirements:	NA
Assumptions:	The user has a list of favourite district(s).
Notes and Issues:	NA

4.7.3 Functional Requirements

1. Favourites Page

1.1 The system will display the saved districts by the user and the respective district information.

1.1.1. The system must display the safety rate for each district.
1.1.2. The system must display the number of schools for each district.
1.1.3. The system must display the number of MRT stations for each district.

1.2 The system will display buttons for user options.
1.2.1 The system will display the favourites button.
1.2.2 The system will display the notifications button.

4.8 Receiving Notifications from Favourites

4.8.1 Description and Priority

Users receive notifications about districts once they have added districts to the favourites and enabled notifications for the favourite district(s).

Overall Priority	Description
Medium	Assigned medium priority as it allows users to be updated in real-time about their preferred districts to make crucial purchasing decisions, although it is not the main functionality of the app

4.8.2 Stimulus/Response Sequences

Use Case ID:	008		
Use Case Name:	Receive Notifications about favourite districts		
Created By:	Dion	Last Updated By:	Dion
Date Created:	28/03/2025	Date Last Updated:	08/04/2025

Actor:	User

Description:	The user receives notifications about districts once they have added districts to the favourites and enabled notifications for the favourite district(s).
Preconditions:	The user must be logged in.
Postconditions:	The notifications are displayed to the user.
Priority:	Medium
Frequency of Use:	Depends on the changes in district information for the favourite districts
Flow of Events:	1. The user navigates to the Favourites page. 2. User selects the notification button(bell icon) to enable notifications for the district. 3. System monitors updates in any information for the district 4. If there is an update in district information for the favourite districts of the user, system shows a pop-up near the name of the district in the Favourites page saying “1 new notification!” 5. Upon clicking on the district, user is redirected to Compare page, where the system shows the details of the notification.
Alternative Flows:	AF-S3: If the user disables notifications, 1. System records the preference and stops sending updates.
Exceptions:	-
Includes:	Push notification integration

Special Requirements:	Notifications should be concise and actionable..
Assumptions	Users want to stay informed about their Favourite districts.
Notes and Issues:	NA

4.8.3 Functional Requirements

1. Receiving notifications

1.1. The system must allow users to enable notifications for individual favourite districts via a bell icon.

1.1.1. The system must not enable notifications for user if the user is not registered or logged in.

1.2. The system must monitor updates to any favourite district for which notifications are enabled.

1.3. Upon detecting an update, the system must display a pop-up near the district's name: "1 new notification!"

1.4. Upon clicking on the district, the system must redirect to the Compare page where the notification details are displayed.

1.4. The system must allow users to disable notifications for specific districts.

1.5. If notifications are disabled, the system must stop sending updates for that district.

4.9 User Profile Page

4.9.1 Description and Priority

A process where a user can access their profile information regarding their username, email and preferences.

Overall Priority	Description
Medium	A user will be able to check their account information, change their preferences, change their password, sign out of their account or even delete it.

4.9.2 Stimulus/Response Sequences

Use Case ID:	009		
Use Case Name:	View Profile		
Created By:	Dion	Last Updated By:	Dion
Date Created:	19/03/2025	Date Last Updated:	29/03/2025

Actor:	User
Description:	The user views the profile, which includes personal information, account settings and preferences.
Preconditions:	The user must be logged in.
Postconditions:	The profile information is displayed to the user.
Priority:	Medium
Frequency of Use:	-
Flow of Events:	1. The user navigates to the profile page. 2. The system retrieves data from the user database. 3. The system displays the user's profile information and preferences.
Alternative Flows:	NA
Exceptions:	EX 1: Failure to load profile data

	If the profile data fails to load, the system informs the user and prompts to try again.
Includes:	NA
Special Requirements:	Secure Handling of personal data.
Assumptions	The user has a profile set up in the system.
Notes and Issues:	NA

4.9.3 Functional Requirements

1. User Profile page
    1.1. The system must display the user's personal information.
        1.1.1. The system must display the user's username.
        1.1.2. The system must display the user's email.
    1.2. The system must display the user's preferences.
        1.2.1. The system must display a price range for the user to set a preferred price.
        1.2.2. The system must display a range for crime rate for the user to set crime rate.
        1.2.3. The system must display a range for the number of schools for the user to set the preferred number of schools.
        1.2.4. The system must display a range for the number of malls for the user to set the preferred number of malls.
        1.2.5. The system must display a range for the number of stations for the user to set the preferred number of stations.
        1.2.6 The system must display the number of favourite districts the user has saved.
        1.2.7. The system must display the number of notifications the user has received for the favourite districts the user has saved.
        1.3. The user must be able to change their password from the user profile page.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Response Time :

- The system should respond to user queries within 2 seconds for 95% of requests based on the industry standard for web application responsiveness
- The web app must respond to user action within 2 seconds
- The map must be able to respond to tap, zoom in and zoom out actions within a second.
- District search results should load within 3 seconds, with more time to accommodate the complex database queries going through multiple filters and sorting options during processing.
- Chart loading time should be completed within 3 seconds, especially for district details viewing, as charts need to be computed and data needs to be pulled from the APIs to display the various information.
- Signing in should be able to be done within 3 seconds after the sign-in button is clicked.

Concurrency :

- The system should support at least 100 concurrent users, representing a reasonable number of people who might use the web app simultaneously.
- Database operations should maintain performance with up to 1000 concurrent transactions to account for the multiple operation requests per user session.
- Multiple favourited district details information needs to be cached so that these districts are ready to be compared within 2 seconds.

Scalability:

- The database should be able to handle up to 50,000 user accounts for future-proofing as more people start using the app in the future and as more APIs on housing details are available for us to implement in the app.
- API endpoints should be able to handle at least 100 requests per second to accommodate the projected number of users using the web app at any given time.

5.2 Safety Requirements

Data Protection:

- All user data (email and password, custom preferences) must be encrypted at rest (when not being used)
- Regular backups must be performed daily for data such as user accounts to prevent data loss.
- The system should implement automatic data recovery procedures in the event of system crashes to minimise downtime and maintain system availability

User Safety:

- Implement rate limiting to prevent system overload and failures. Rate limiting should be a number that is hard to hit under normal use circumstances. For example, limit login attempts to 5 per minute.
- Validate all user inputs via server-side validation and client-side validation to prevent any possible injection attacks.
- Implemented proper error handling to prevent system crashes.

5.3 Security Requirements

Authentication and Authorisation:

- Implement secure password hashing to protect user passwords even if the database is compromised.
- Require strong password policies (minimum 8 characters)

Data Security:

- All API communications should use HTTPS to encrypt all data in transit, ensuring data privacy
- General Data Protection Regulation(GDPR) compliance for user data handling with user consent management, right to access personal data, and to be forgotten and data portability.
- The user must be clearly informed of the data being collected (eg, their preferences and email) and must be able to delete their data if they want to.

API Security:

- Rate limiting for API endpoints to prevent abuse and ensure fair usage, as mentioned before
- Input validation for all API parameters, such as data type checking to aid in response handling.

5.4 Software Quality Attributes

Reliability and Accuracy:

- System uptime of 99.9% to ensure nearly all-time server availability
- Automated error logging and monitoring, with Real-time error detection techniques such as input compatibility detection and monitoring system errors.
- Graceful degradation during high load, with resource allocation optimization strategies.
- Latest government API used to get the most accurate information as up to date.

Maintainability:

- Maintaining proper code documentation for all major components.
- Modular architecture for easy updates and feature addition in the future. Design principles such as single responsibility, loose coupling, High cohesion and interface segregation are used.
- Comprehensive test coverage (minimum 80% of all code components are tested). Various tests include unit tests, system tests, and performance tests to test core functionality and error conditions with performance targets met.

Usability:

- Intuitive user interface with consistent layout, straightforward navigation, logical flow and familiar patterns. The same floating glassmorphism design language has been used throughout the user interface in the system for familiarity and visual appeal.
- Responsive design for all device types with smooth transitions and efficient rendering.
- Clear error messages and user feedback with user-friendly language and actionable solutions for warning alerts and error notifications. For example, users are told to log in before trying to add a district to their favourites.
- The user should be able to understand how to navigate the entire app within 2 minutes of loading up the app for the first time as a testament to the user-friendliness of the web app.

Interoperability:

- Good API design with filtering and sorting support for APIs and proper request and response formats
- Standard data formats (JSON) for data exchange with clear naming for proper validation during data handling.

- Cross-browser compatibility: Chrome and Safari were tested using manual verification.

5.5 Business Rules

User management:

- Rightfully, users should be above the age of 21 as this app is more suited to their age group.

API Usage:

- APIs used are all government-based to ensure accurate and reliable information.

Language Support :

- Mainly English but can extend support to various other national languages.

6. Other Requirements

Optimisation Areas:

- User queries: for better input handling
- Price range filtering: to accommodate all proper price ranges from the maximum to the minimum

Monitoring and Logging:

- System performance monitoring to monitor system errors, which includes response time, resource allocation and error rates.
- Application and error logs for performance improvements later on.

Appendix A: Glossary

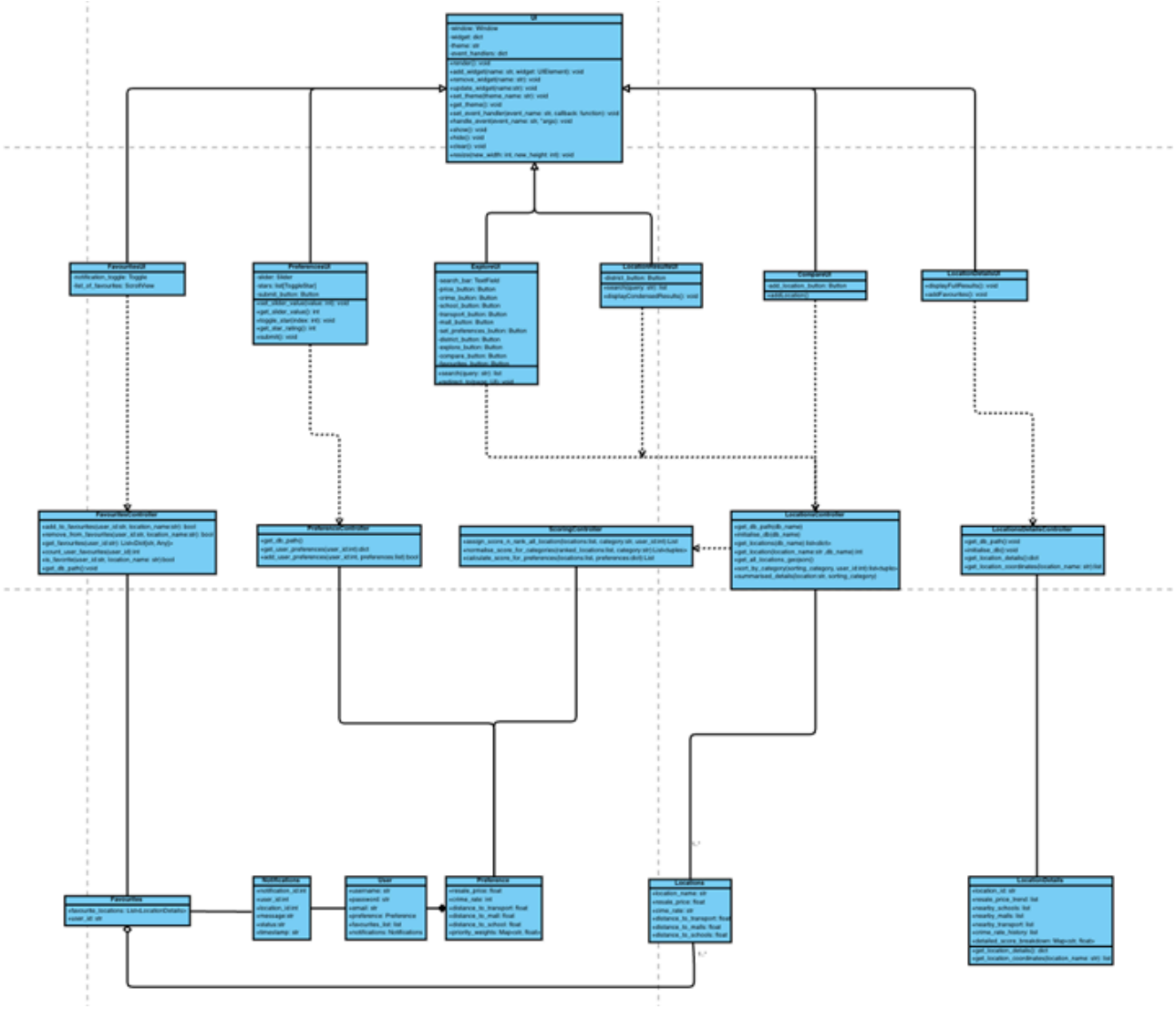
Data Dictionary

Term	Definition
District	One of Singapore's 55 urban planning areas, each with unique housing characteristics
Favourite Districts	Specific districts that users saved for quick access in the future.
Favourite Page	A page where users can view and manage their saved Favourite Districts.
Compare Page	Interface displaying side-by-side analysis of two selected districts
Categories	Key factors that users consider when choosing a new residence. These factors are used to filter locations. The 5 categories include: - Resale price (as defined below) - Neighbourhood crime rate – Estimated based on past crime statistics. - Nearby MRT stations – The number of MRT stations in the district - Nearby schools – The number of primary, secondary and Junior College schools in the district. - Nearby malls – The number of shopping malls in the district.
Resale Price Trends	12-month historical price fluctuations by flat type (visualized monthly)
Crime Rate	A measurement of the safety of a location, derived from an estimation based on the number of past police reports and crime data in the area
District Details	Key details about a district, including: - Resale price - Crime rate - Nearby Schools - Nearby Malls - Nearby MRT stations - Details on past crime incidents
Watchlist	Synonymous with “Favourite Location” or “Favourite Districts” – users saved locations for tracking any updates in the location details.

User Preferences	Customisable settings that allow users to refine their search based on personal criteria in the 5 categories.
Map View	A visual representation of the locations on an interactive map.
List View	A structured text-based format displays location search results with location information, ranked according to the score.
Score	An average of the score for each category. Score for each category is derived from data analysis and comparative metrics. Score is from 0-5.

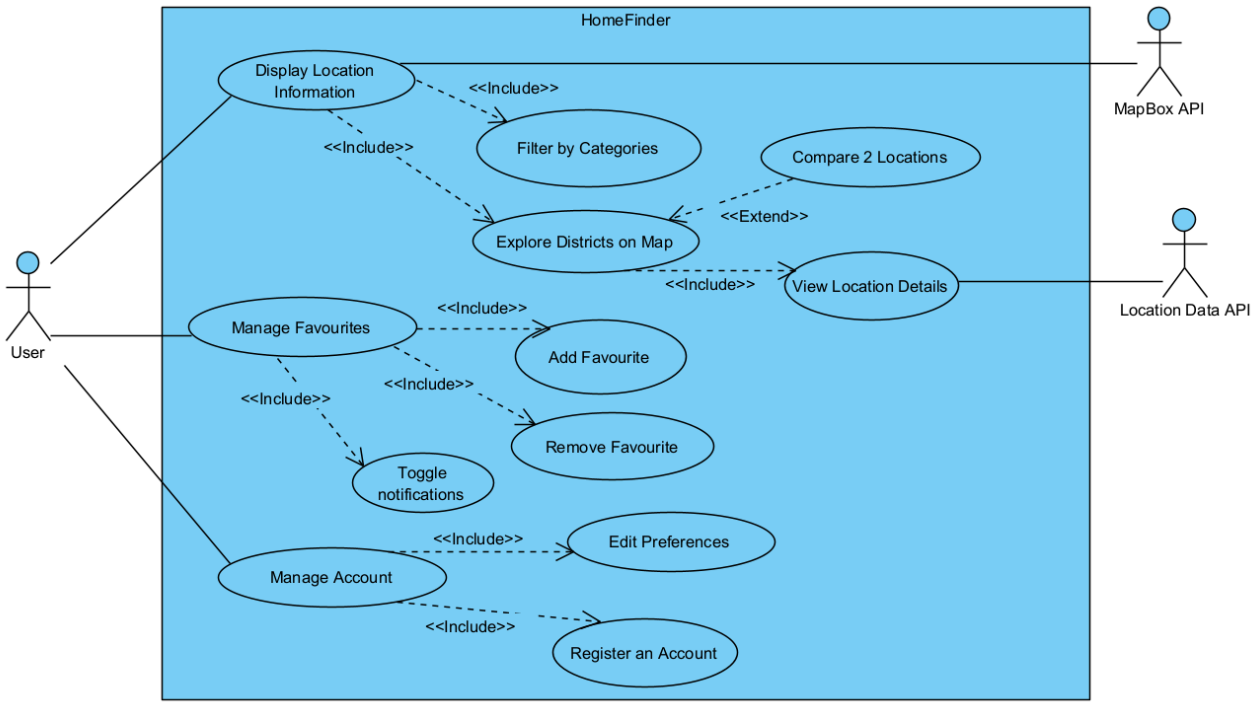
Appendix B: Analysis Models

Class Diagram

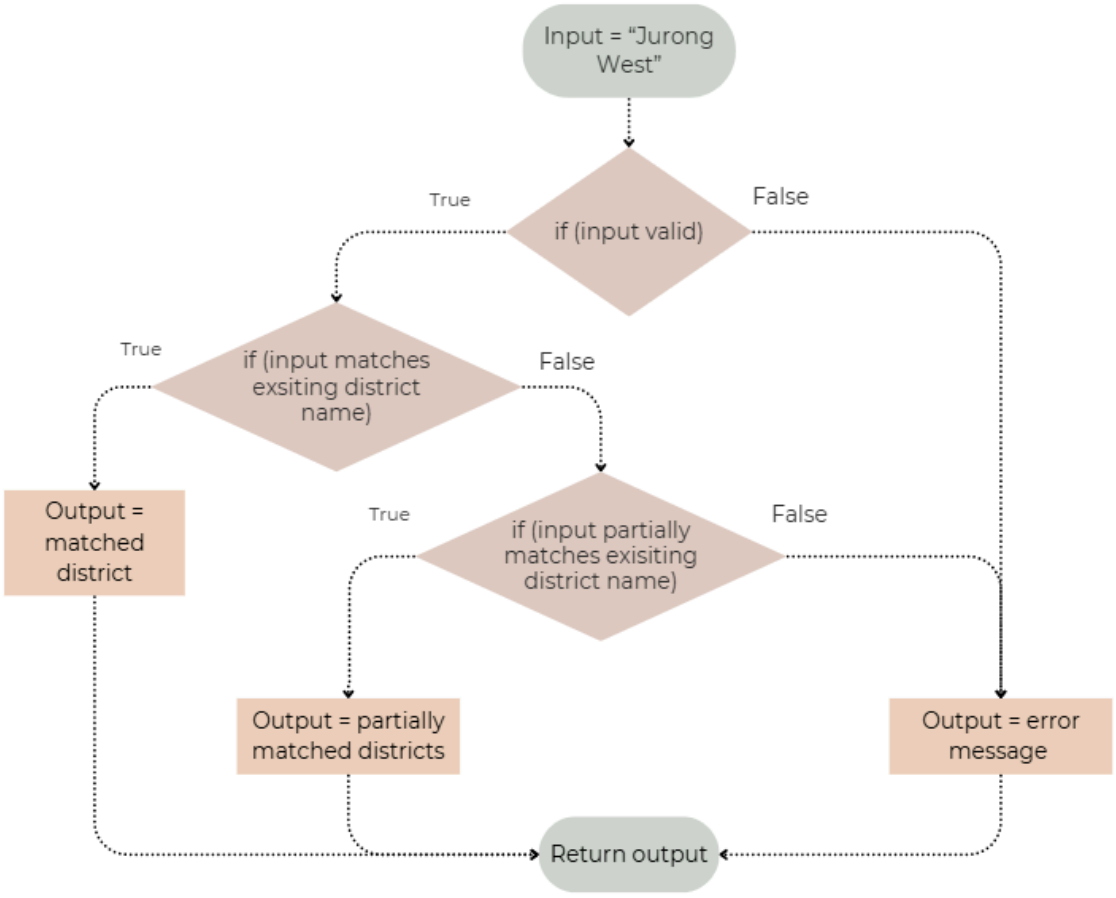


Use Case Diagram

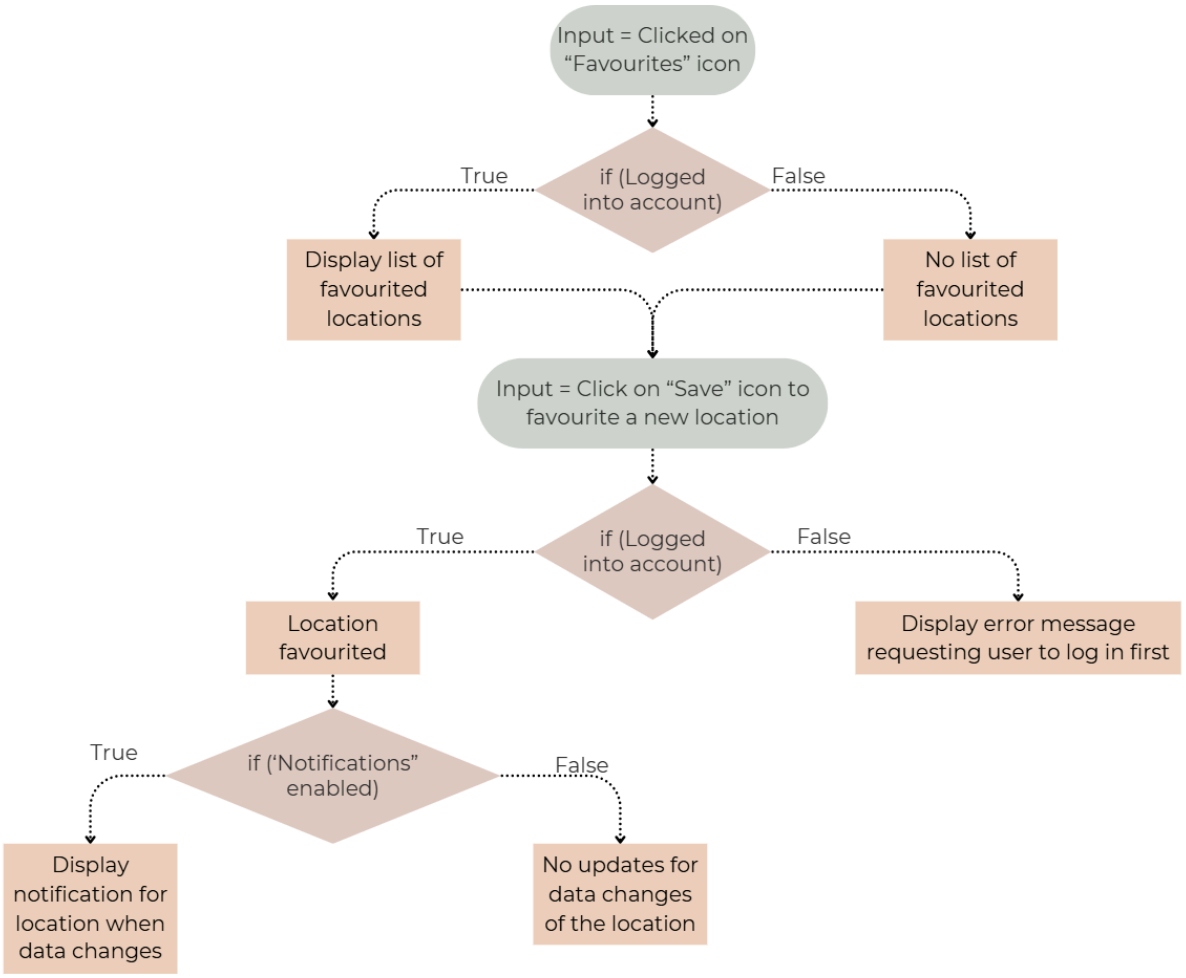
/


HomeFinder
Display Location Information
Filter by Categories
Compare 2 Locations
Explore Districts on Map
View Location Details
Manage Favourites
Add Favourite
Remove Favourite
Toggle notifications
Edit Preferences
Manage Account
Register an Account
<<Include>>
<<Extend>>
MapBox API
Location Data API
User

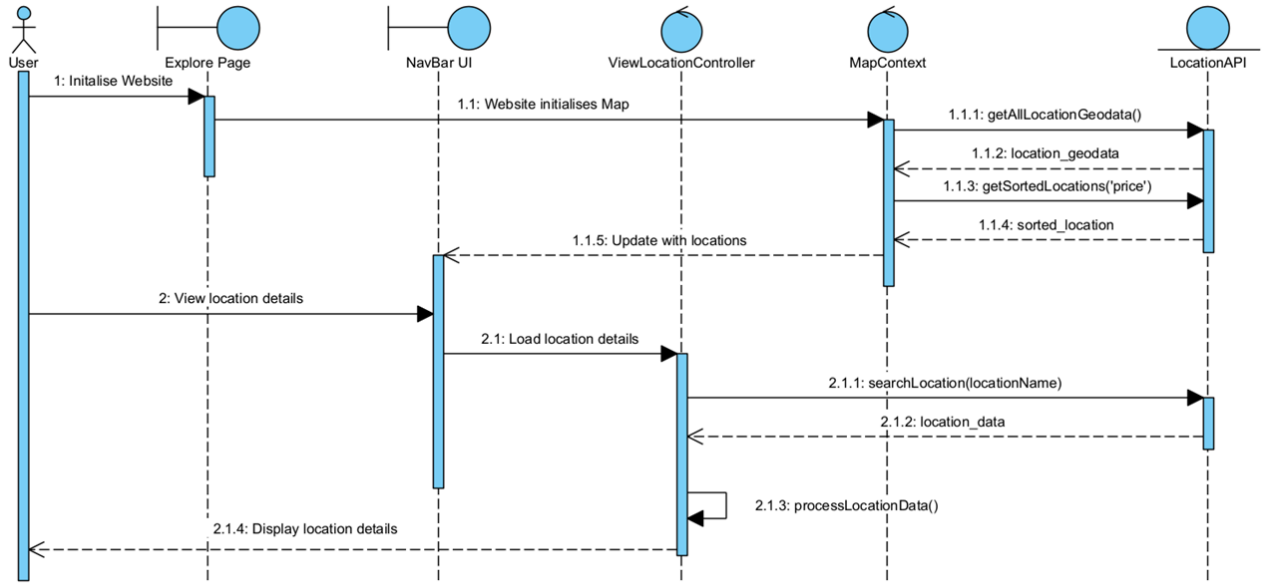
Control Flow Graph
Search Location



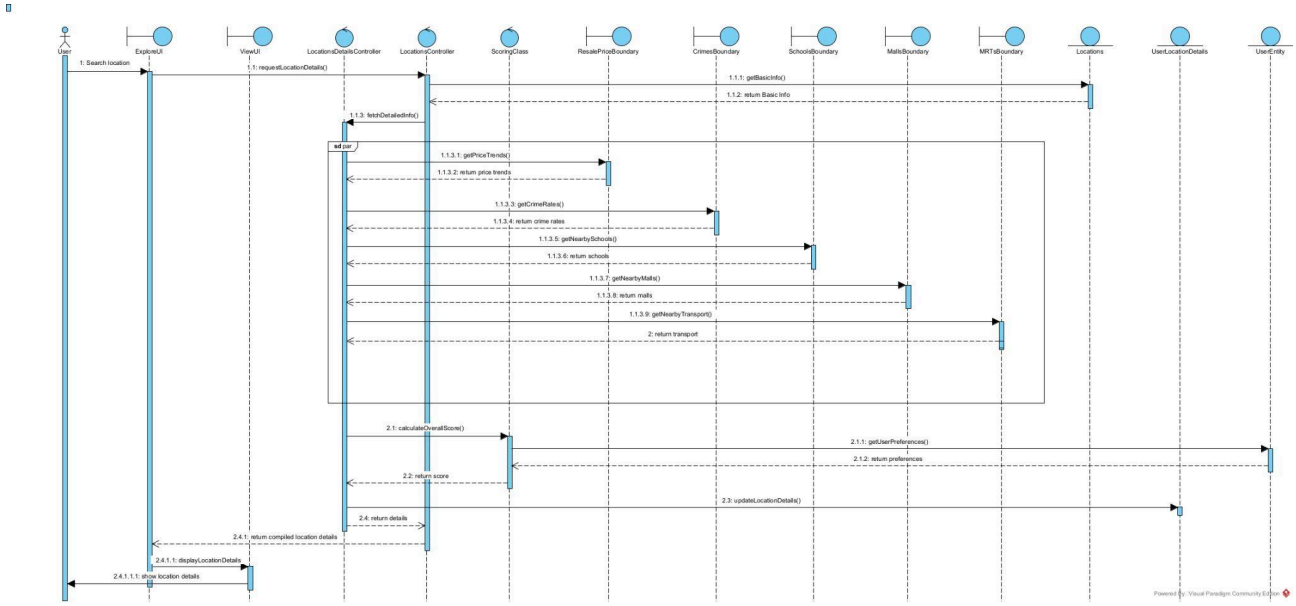
Favourite


Input = Clicked on "Favourites" icon
if (Logged into account)
True
False
Display list of favourited locations
No list of favourited locations
Input = Click on "Save" icon to favourite a new location
if (Logged into account)
True
False
Location favourited
Display error message requesting user to log in first
if ('Notifications" enabled)
True
False
Display notification for location when data changes
No updates for data changes of the location

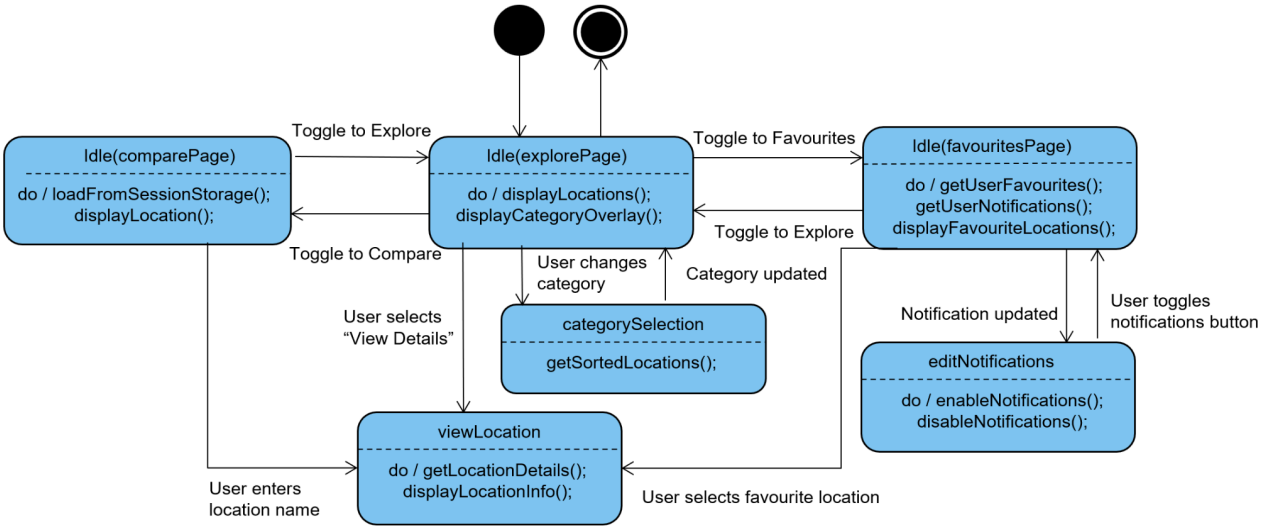
Sequence Diagram
Search Location



Viewing and Exploring Location



Dialog Map


Idle(comparePage)
do / loadFromSessionStorage();
displayLocation();
Idle(explorePage)
do / displayLocations();
displayCategoryOverlay();
Idle(favouritesPage)
do / getUserFavourites();
getUserNotifications();
displayFavouriteLocations();
categorySelection
getSortedLocations();
editNotifications
do / enableNotifications();
disableNotifications();
viewLocation
do / getLocationDetails();
displayLocationInfo();
Toggle to Explore
Toggle to Favourites
Toggle to Compare
Toggle to Explore
User changes category
Category updated
User selects "View Details"
Notification updated
User toggles notifications button
User enters location name
User selects favourite location