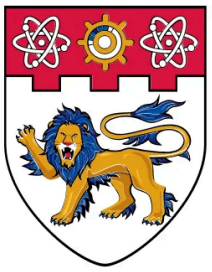


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SCC0006 - Software Engineering

Group Project Report (Software requirement specifications)

Name	Matriculation number
Chua Chong En	U2320759L
Wong Si Kei Wynette	U2210236B
Sanjana Shanmugasundaram	U2321261L
Murugan Rithika	U2323063E
Kumaresan Gayathri Devi	U2323162L
Chang Jing Xian, Max	U2221183G

Software Requirements Specification

for

FlatFinder

Version 1.0 approved

Prepared by Group 43

Nanyang Technological University

17 November 2024

Table of contents

1. Introduction

1.1. Purpose

In recent years, Singapore has witnessed an increasing trend of more eligible singles opting to purchase flats, particularly resale flats. This shift is driven by the faster move-in times and greater flexibility resale flats offer compared to Build-To-Order (BTO) flats. However, despite the growing demand, purchasing a flat comes with its set of challenges:

1. **Complex Eligibility Criteria**: Navigating the rules and requirements for purchasing a flat can be daunting for first-time buyers, especially with various schemes, grants, and conditions.
2. **Tedious Research Process**: Finding the right property involves extensive research into locations, amenities, pricing, and financial implications, making it a time-consuming task.
3. **Difficulty in Determining Appropriate Financing Options**: Understanding and selecting the most suitable financing option, whether through HDB loans or bank loans, adds another layer of complexity. Buyers need to consider their salary, CPF balance, and loan tenure carefully to ensure financial sustainability.

These challenges highlight the need for innovative solutions, such as **FlatFinder**, to simplify the housing search and purchasing process, providing tailored recommendations and financial insights to empower buyers.

1.2. Document Conventions

This Software Requirements Specification (SRS) document follows specific standards and conventions to ensure clarity, consistency, and ease of understanding for all stakeholders. Below are the key conventions used in this document:

1.2.1 Formatting and Typographical Conventions

- **Headings**:
  - **Level 1 headings** (e.g., Section 1, Section 2) are bold and numbered for primary organisation.
  - **Subheadings** (e.g., Section 1.1, 1.1.1) are bold and indented for better hierarchy.
- **Text Styles**:
  - **Bold**: Used for section titles, keywords, or terms of emphasis.
  - *Italic*: Used for examples, notes, and supplementary explanations.

    - **Code Blocks/Commands**: Represented in monospace font to distinguish technical instructions.
- **Lists**:
    - **Bullet points**: Used for general items or unordered lists.
    - **Numbered lists**: Used for steps or sequences of tasks.
- **Prioritisation**:
    - High-priority requirements are marked with **[High Priority]** in parentheses, while medium and low-priority items are marked with **[Medium Priority]** or **[Low Priority]** where relevant.

1.2.2 Inheritance of Priorities

- Priority levels specified at a higher-level requirement are **assumed to be inherited** by all detailed sub-requirements unless explicitly stated otherwise.

1.2.3 Requirement Identifiers

- Each requirement is uniquely identified using a hierarchical numbering system (e.g., **1.1.3.1**, **1.2.3.4**) to enable traceability throughout the document.

1.2.4 Acronyms and Abbreviations

- Common acronyms and abbreviations used in the document include:
    - **SRS**: Software Requirements Specification.
    - **HDB**: Housing and Development Board.
    - **CPF**: Central Provident Fund.
    - **URA**: Urban Redevelopment Authority.
    - **UID**: Unique Identifier.
      A glossary of terms is provided in **Appendix A**.

1.2.5 Notes and Annotations

- *Notes* or additional explanations are indicated in italicised text for clarification.
- Warnings or important information are highlighted in **bold**.

1.2.6 Version Control

- Document updates are tracked in the **Revision History** section to reflect changes, additions, or deletions of requirements.

- This consistent structure ensures that all stakeholders, including developers, testers, and business analysts, can easily navigate, understand, and implement the requirements detailed in this document.

1.3. Intended Audience and Reading Suggestions

1.3.1 Intended Audience

This Software Requirements Specification (SRS) document is designed for the following types of readers, each with their specific roles and responsibilities:

- **Developers**:
  To understand the functional and non-functional requirements of the system and implement them accordingly.
- **Project Managers**:
  To oversee the project's progress and ensure that deliverables align with the specified requirements.
- **Marketing and Product Teams**:
  To comprehend the features and functionalities of the system for promoting the product to the target audience.
- **Testers**:
  To identify the expected behaviours and edge cases for verifying that the system meets the requirements.
- **Documentation Writers**:
  To produce user manuals, training materials, and other supporting documents based on the system features and specifications.
- **End Users**:
  To gain insight into the functionalities of the system and understand how it caters to their needs.

1.4. Product Scope

FlatFinder is a web-based application designed to assist individuals in finding resale housing options tailored to their financial capacity and preferences. By leveraging user inputs such as salary, CPF balance, housing preferences, and housing type, the app provides personalized property recommendations, financial insights, and interactive tools to simplify the process of purchasing a flat.

Relevant Benefits

1. **Simplified Housing Search**:
   Reduces the complexity of manually searching through vast property listings by offering tailored recommendations based on user inputs.
2. **Financial Insights**:
   Provides users with an affordability range, loan eligibility, and repayment estimates to ensure they can make financially sound decisions.
3. **User-Friendly Tools**:
   Offers interactive features like map-based property views, nearby amenities, and street views to enhance decision-making.
4. **Time Efficiency**:
   Enables users to save time through real-time recommendations and dynamic updates as preferences change.

Objectives

- Provide **accurate and personalised property recommendations** by analysing user inputs and government data (e.g., HDB and URA resale listings).
- Deliver **financial calculations**, such as loan amounts and repayment periods, based on individual salary and CPF balances.
- Offer **interactive visualisations**, including map views, colour-coded affordability indicators, and nearby amenities.
- Allow users to **bookmark and compare properties** to streamline the decision-making process.

Goals

- Empower users to make **informed housing decisions** by simplifying the process of finding and evaluating resale flats.
- Align with Singapore's housing policies and address challenges such as complex eligibility criteria and difficulty in determining appropriate financing.
- Support broader corporate goals of innovation in property search technology and enhancing user satisfaction through smart solutions.
-

Relation to Business Strategies

FlatFinder aligns with the growing demand for **digitised and personalised services** in the real estate industry, providing a competitive edge by:

- Targeting the underserved market of eligible singles and first-time homebuyers seeking resale flats.

- Enhancing user engagement through innovative features and a seamless user experience.
- Contributing to corporate strategies of digital transformation and improving customer satisfaction.

By addressing the challenges of affordability, eligibility, and financial complexity, FlatFinder becomes a valuable tool in the housing search process, ensuring users can find their ideal homes efficiently and confidently.

1.5. References

This section lists the references used in the development of this Software Requirements Specification (SRS). These documents, APIs, and data sources provide relevant information and context for the system's design and functionality.

1.5.1 Housing Data

- **Title**: Singapore Resale Housing Data
- **Source**: Data.gov.sg
- **Description**: Provides publicly available datasets on resale flats, including transaction details, property types, locations, and prices.
- **Version**: Latest version as of November 2024
- **Usage**: Used for retrieving accurate and up-to-date resale housing data for recommendation algorithms.

1.5.2 Geospatial Data and Mapping

- **Title**: Google Maps Javascript, Geocoding and Places API
- **Source**: Google Cloud
- **Description**: Provides geolocation services, including map views, street views, and nearby amenities data.
- **Version**: Latest stable release as of November 2024
- **Usage**: Used for interactive map visualisations, nearby amenities integration, and providing street views of properties.

1.5.3 User Data and Authentication

- **Title**: Firebase Authentication and Firestore Documentation
- **Source**: Firebase
- **Description**: Official documentation for implementing secure user authentication.
- **Version**: Latest stable release as of November 2024
- **Usage**: Used to handle user registration, login, logout, and secure profile management.

1.5.4 AI Personal Assistant

- **Title**: Google Gemini for Google Cloud & Generative Language API
- **Source**: Google Cloud
- **Description**: Provides insightful advice on potential housing options while taking into account the user's financial and housing preferences.
- **Version**: Latest stable release as of November 2024
- **Usage**: Used for the interactive AI chatbot on the Edit Preferences Screen.

2. Overall Description

2.1. Product Perspective

FlatFinder is a new, self-contained product designed specifically for the Singapore housing market. It aims to address the challenges faced by individuals and families in navigating the complexities of the resale flat search process.

This product is not part of an existing product family or a replacement system but is developed as a standalone solution to fill a gap in the market. It leverages cutting-edge web technologies to deliver an intuitive user experience while integrating with external systems, such as HDB flat databases, to provide accurate and relevant recommendations.

1. Nature of the Product

This product, "FlatFinder," is a web-based website designed to facilitate the search and management of resale flats for users in Singapore. It provides an interface for users to manage their preferences (e.g., salary, CPF balance, HDB type), search for flats based on personalised criteria, and receive recommendations that align with their budget and housing needs.

2. **System Components and Interfaces**

- **FlatFinder Core System:**
    - User preference management module
    - Flat search and recommendation engine
    - Budget and affordability calculator
- **External Interfaces:**
    - HDB database for resale flat listings and policies
    - External APIs for map, location and AI LLM services

2.2 Product Functions

2.2.1 FlatFinder allows users to search for flats that are related to their liking through considerations of their preferences and financial abilities.

2.2.2 Secure authentication process for login and registration

2.2.3 Personalised Result based on User Preferences and Information

2.2.4 An interactive map allows users to pick listings that are within/ slightly above or outside the user's preferred price range.

2.2.5 Allows point of view perspective of the surrounding of chosen listing.

2.2.6 Allows listings to be compared to other listings.

2.2.7 Allows listings to be saved for later viewing and deleted if not needed.

2.2.8 Chatbot to provide additional help related to choosing listings

2.3 User Classes and Characteristics

2.3.1. General Users (Home Seekers):

2.3.1.1 Frequency of Use:

2.3.1.1.1 Occasional to frequent, depending on how actively they are searching for resale flats.

2.3.1.2 Functions Used:

2.3.1.2.1 Search for flats.

2.3.1.2.2 Save and compare listings.

2.3.1.2.3 View budget recommendations.

2.3.1.3 Technical Expertise:

2.3.1.3.1 Basic familiarity with web applications.

2.3.2. Advanced Users (Property Agents):

2.3.2.1 Frequency of Use: Frequent

2.3.2.2 Functions Used:

2.3.2.2.1 Search and filter flats for multiple clients.
2.3.2.2.2 Save listings for later recommendations.

2.3.2.3 Technical Expertise: Comfortable using advanced search and comparison features.

2.3.3. Administrators:

2.3.3.1 Frequency of Use: Infrequent.

2.3.3.2 Functions Used:

2.3.3.2.1 Manage external API integrations and user credentials.
2.3.3.2.2 Oversee data integrity and perform system audits.

2.3.3.3 Technical Expertise: High-level understanding of the system.

2.4 Operating Environment

2.4.1 Hardware Requirements:

2.4.1.1 Windows or MacOS computer
2.4.1.2 Minimum: 4 GB RAM, 1.6 GHz processor

2.4.2 Operating Systems:

2.4.2.1 Windows 10 or later
2.4,2.2 macOS Catalina or later

2.4.3 Browsers:

2.4.3.1 Microsoft Edge (latest version)
2.4.3.2 Google Chrome (latest version)
2.4.3.3 Safari (latest version)

2.4.4 Software Dependencies:

2.4.4.1 Terminal to launch initial setup scripts
2.4.4.2 Required libraries installed during setup (e.g., Node.js, React, FastAPI for backend)

2.4.5 Network Requirements

2.4.5.1 Reliable internet connection for external backend access.

2.5 Design and Implementation Constraints

2.5.1 Regulatory Constraints

2.5.1.1 Compliance with Singapore's housing data and PDPA regulations.

2.5.2 Technology Constraints:

2.5.2.1 The system must integrate with the HDB resale flat database and external map services (e.g., Google Maps).
2.5.2.2 Chatbot utilising Google Gemini API
2.5.2.3 Backend development using FastAPI.
2.5.2.4 Frontend using React.js.

2.5.3 Security Constraints:

2.5.3.1 Secure user authentication using Firebase.
2.5.3.2 Encrypted communication for sensitive data (e.g., passwords).

2.5.4 Database Constraints:

2.5.4.1 Relational database to store user preferences and search history (e.g., FireStore).

2.6 User Documentation

2.6.1 A video guide on how to use and navigate Flatfinder will be provided. It consists of demonstrations on:

2.6.1.1 Account Creation Procedure

2.6.1.1.1 Includes specifications on the type of email and password to be used.

2.6.1.2 Password Reset Procedure

2.6.1.2.1 Include instructions on how to find the password reset pin and type of new password to be used.

2.6.1.3 Login Procedure

2.6.1.3.1 Include specifications on type of email and password to be used.

2.6.1.4 Application walkthrough

2.6.1.4.1 Include instructions on how to navigate FlatFinder and its features

2.6.1.5 Logout procedure

2.6.1.5.1 Include instructions on how to logout.

2.7 Assumptions and Dependencies

2.7.1 Assumptions:

2.7.1.1 Users have access to a stable internet connection.
2.7.1.2 HDB database and external APIs will provide consistent, reliable data and response.
2.7.1.3 User devices meet the minimum hardware and software requirements.

2.7.2. Dependencies:

2.7.2.1 External APIs:

2.7.2.1.1 Google Maps API for location and map services.
2.7.2.1.2 HDB API for flat listings and policies.
2.7,2.1.3 Google Gemini API for chatbot response.

2.7.2.2 Third-party Libraries:

2.7.2.2.1 React for frontend, FastAPI for backend.
2.7.2.2.2 Libraries for authentication and database (Firebase, Firestore).

2.7.3 Operating System: Compatibility with Windows and macOS.

3. External Interface Requirements

3.1 User Interfaces

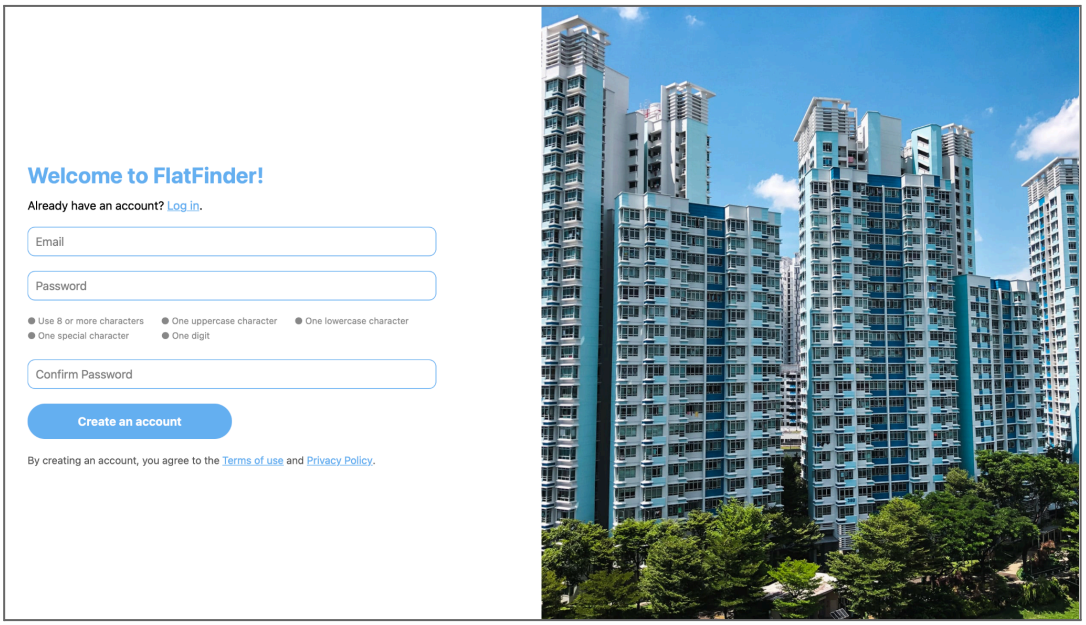
React Native's ActivityIndicator and TouchableOpacity components are used throughout the UI to ensure that it is intuitive for the user to understand when a backend process is occurring through a visible loading indicator after clicking a button. Clickable links for navigation are also highlighted and underlined.

3.1.1 User Account Creation

3.1.1.1 Registration:

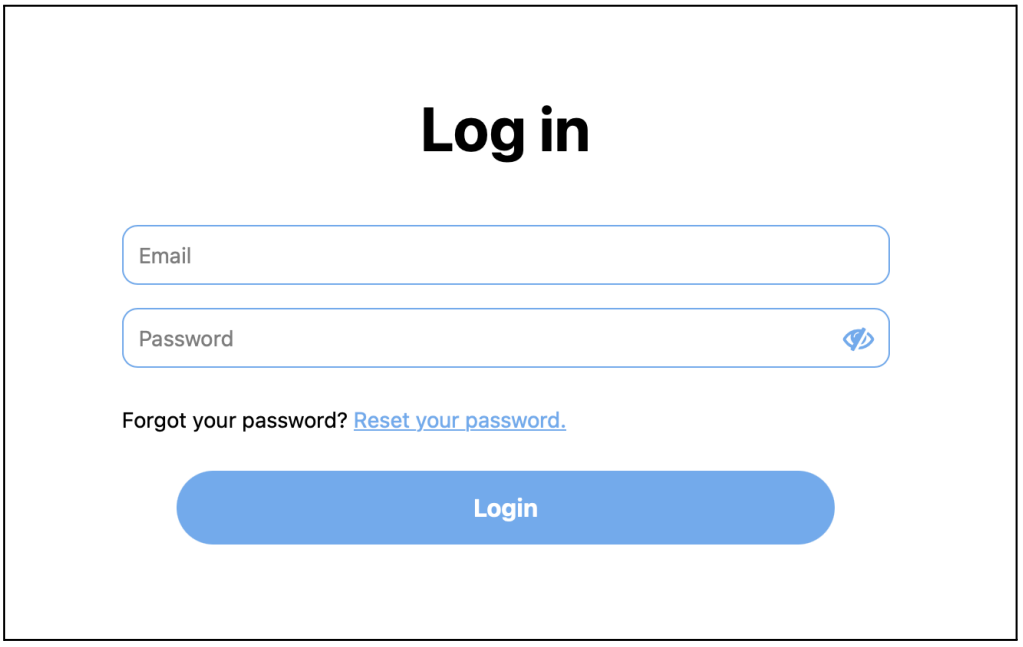
This page provides fields (Email, Password, Confirm Password) to create an account. Error messages will be displayed when certain criterias such as invalid email, existing account, a weak password or mismatching passwords

are entered.



3.1.1.2 Login:

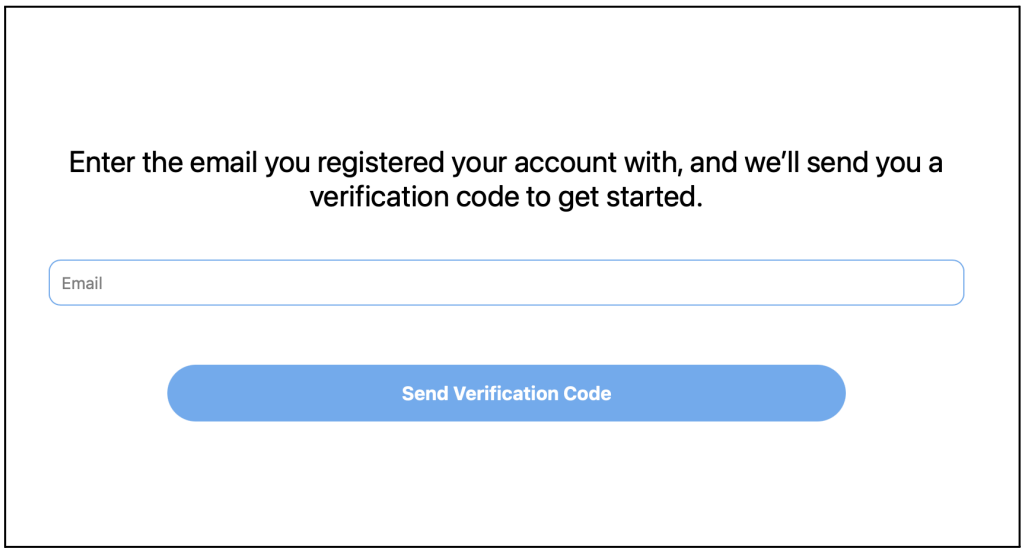
This page provides fields (Email, Password) to enter user credentials to log in to FlatFinder. Error messages will be displayed when certain criterias such as invalid email or incorrect password are entered.



3.1.2 User Account Management

3.1.2.1 Reset Password:

This page allows the user to reset their password after entering their email. A secure 6-digit verification code is sent to their email using the email.js library. The user enters the verification code on the verification screen and if the code is valid, the user is allowed to set a new password.

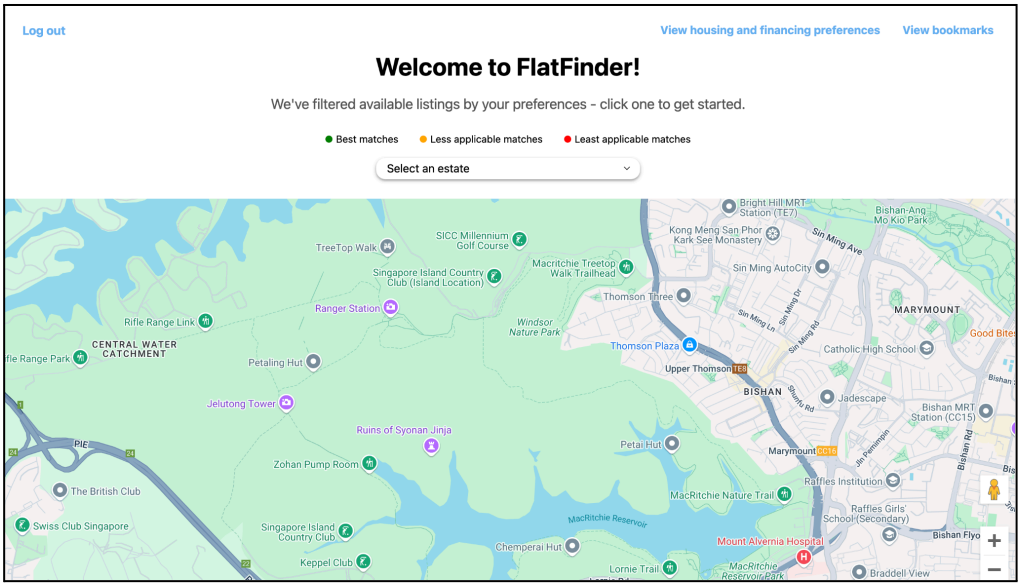


3.1.3 User App Navigation

3.1.3.1 Home Screen:

This page allows the user to perform various actions such as selecting estates and listings on the map. Users can filter apartment listings by selecting an estate from a dropdown menu (Picker component). The selected estate (selectedEstate value) retrieves its data by sending an API request to the backend (e.g. fetchApartments(selectedEstate)). The backend returns a list of apartments with details like block, street name and resale price which are then displayed on the map.

It is also the main page of the website, hosting links to “View housing and financial preferences” and “View bookmarks”.

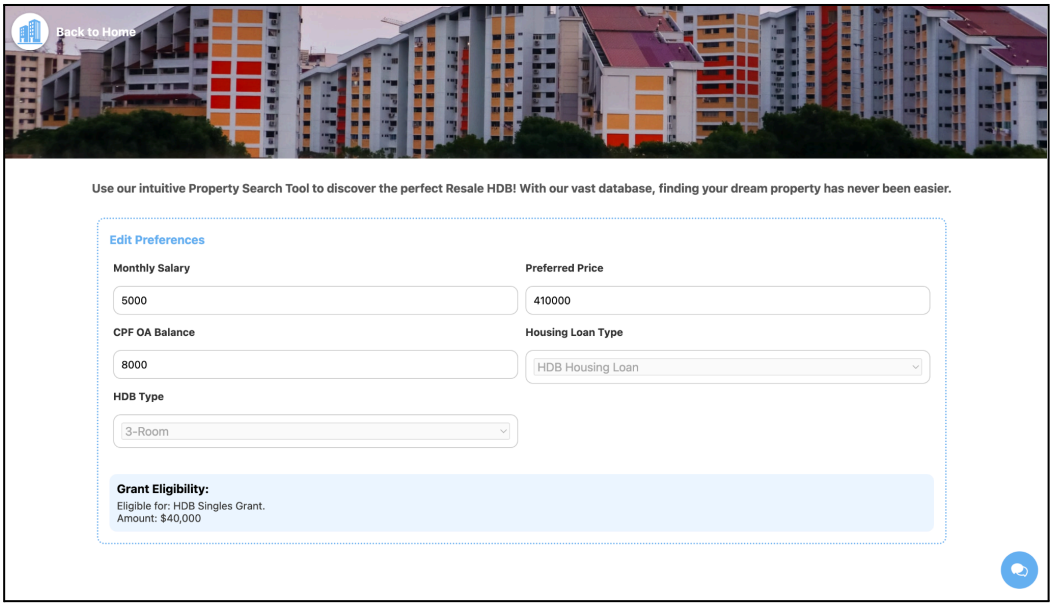


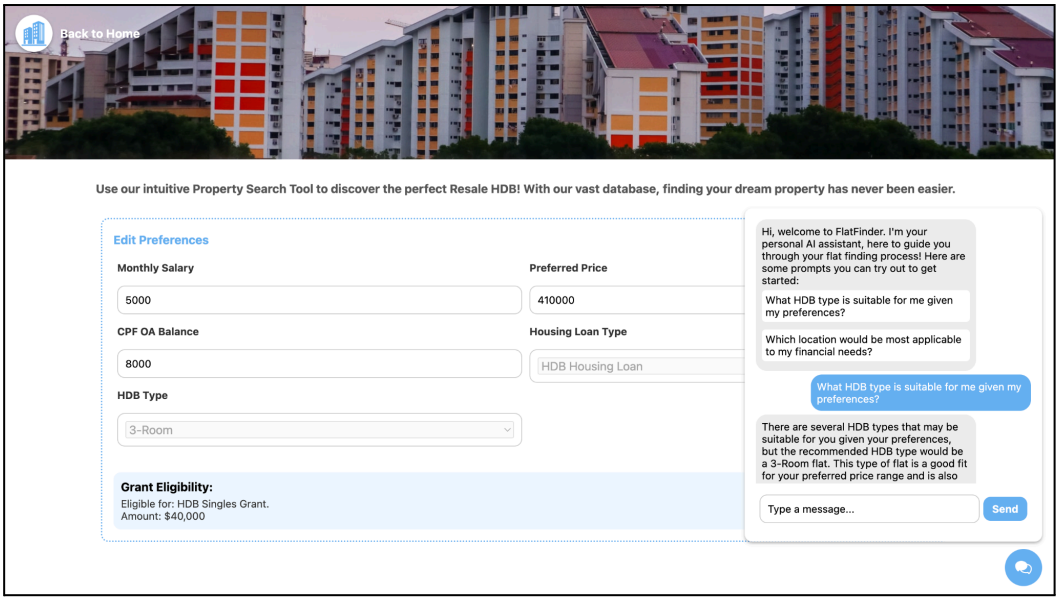
3.1.3.2 User Preferences Screen:

This page enables users to input and manage their housing preferences, which are utilised by the system to filter and calculate property listings. The system dynamically adjusts its recommendations based on these inputs, providing listings that fall within, slightly above, or outside the user's preferred price range.

Users can enter numerical values for their monthly salary, CPF OA balance, and preferred price using TextInput components and select their preferred HDB type and housing loan type from dropdown menus implemented using Picker.

The user can also utilise the Google Gemini embedded chatbot for further assistance pertaining issues regarding housing. The chat window is implemented using a ScrollView component, ensuring all messages are accessible and the chat is scrollable, while the scrollViewRef component ensures the chat automatically scrolls to the bottom whenever a new message is received or sent, providing a seamless conversational experience.

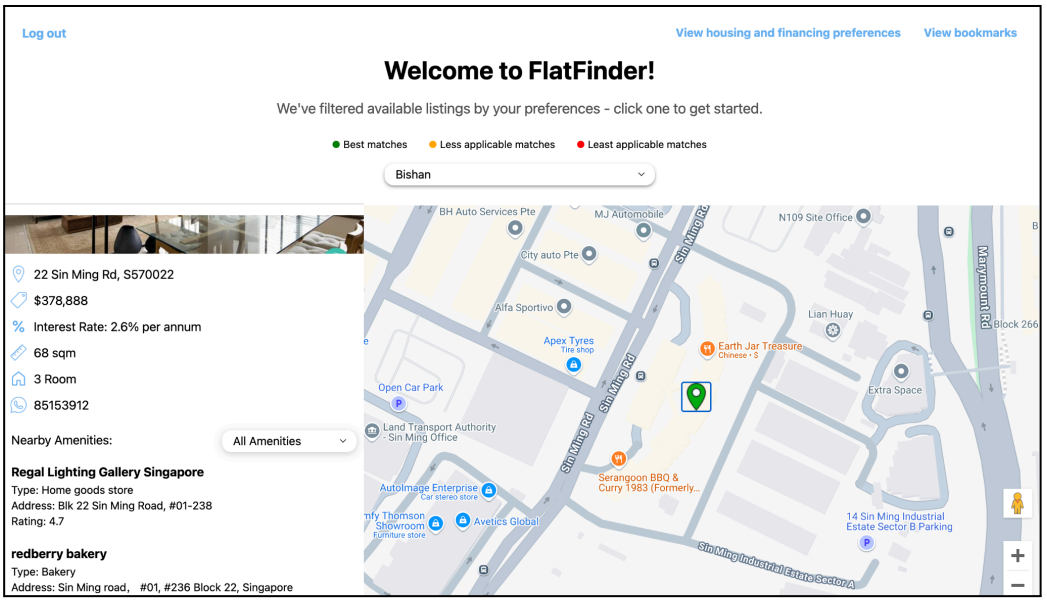




3.1.3.3 Expanded Listing:

This allows the user to see details regarding the listing that they have selected. It also allows the user to bookmark/unbookmark listings by clicking on the heart shape icon on the top right of the pop-up.

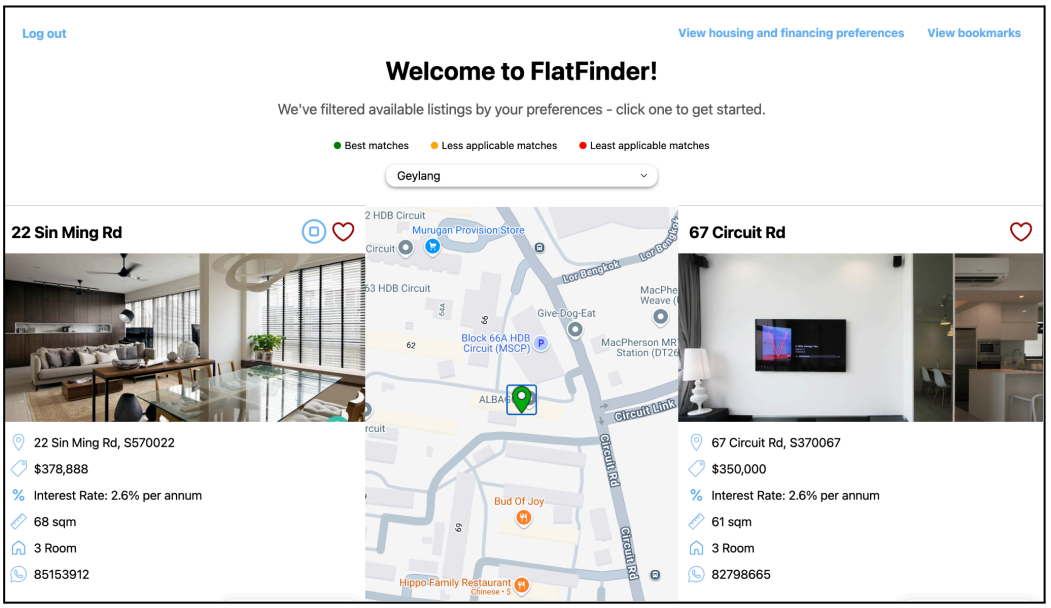
The pop-up panel leverages a ScrollView component, enabling users to scroll through all the available details, including pricing and amenities. Additionally, a Picker component is provided within the panel to filter and view nearby amenities by category. Users can select specific types of amenities, such as schools, points of interest, or restaurants, from the dropdown menu, making it easier to evaluate the listing's surroundings based on their preferences.



3.1.3.4 Compare Listing:

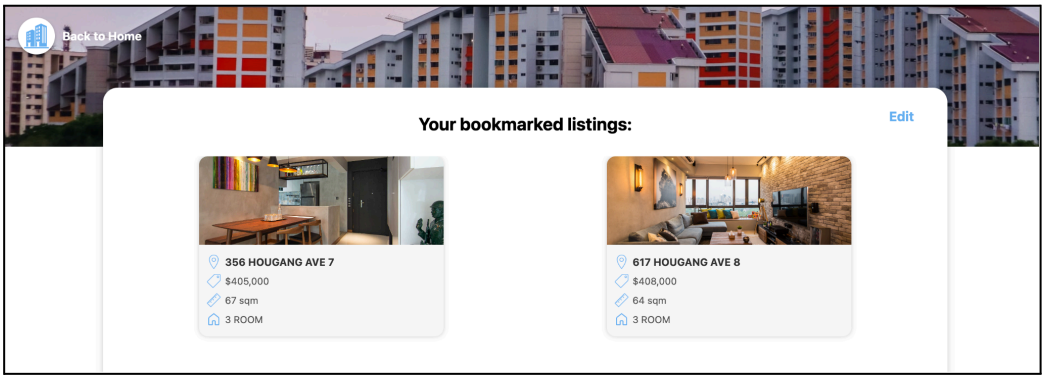
By clicking the Compare button within the expanded listing, users can initiate a side-by-side comparison of two listings. The application uses separate components and functions for the regular and compare listing panels, ensuring that each panel displays distinct information without duplication.

Each panel includes a ScrollView for seamless navigation through detailed listing data. Additionally, both panels provide a Picker dropdown menu to filter and compare nearby amenities based on the selected category, enabling users to evaluate the surrounding features of each property efficiently.



3.1.3.5 Bookmarked Listings Screen:

This page allows the user to see listings that they have bookmarked for future references. It also allows the user to delete the listing by clicking on the “delete” button when they hover over the selected listing.



3.2 Hardware Interfaces

3.2.1 User’s Device:

3.2.1.1 Desktop or laptop for accessing the web application.

3.2.2 Input Devices:

3.2.2.1 Keyboard for data input.
3.2.2.2 Mouse or trackpad for navigation and map interaction.

3.2.3 Output Devices:

3.2.3.1 Display screen for viewing flat listings and recommendations.

3.3 Software Interfaces

FlatFinder integrates with various software components, tools, and databases to deliver its functionality. Below are the detailed descriptions of the connections, data exchanges, and services provided or required by the system:

3.3.1 Database Interfaces

3.3.1.1 Database Used: MySQL (version 8.0) or PostgreSQL (version 14) for storing user profiles, preferences, and flat listings.

3.3.1.2 Purpose:

3.3.1.2.1 Store and manage user data, including preferences, search history, and recommendations.
3.3.1.2.2 Maintain a dynamic list of resale flats and associated details from external data sources.

3.3.1.3 Data Flow:

3.3.1.3.1 Incoming: User preferences, CPF details, search parameters.

3.3.1.3.2 Outgoing: Recommended flat listings, affordability analysis results.

3.3.1.4 Nature of Communication: SQL queries over a secure connection (e.g., HTTPS or SSH).

3.3.2 External API Interfaces

3.3.2.1 HDB Database API (Data.gov.sg):

3.3.2.1.1 Purpose: Access flat availability, pricing, and location data.

3.3.2.1.2 Data Flow:

3.3.2.1.2.1 Incoming Data: Flat listings with details such as address, price, size, and eligibility conditions.

3.3.2.1.2.2 Outgoing Data: Search queries with filters (e.g., flat type, price range, location).

3.3.3 Map Services API (e.g. Google Maps, One Maps Singapore, Google Places, Google Geocoding):

3.3.3.1 Purpose: Display flat locations and calculate commute times.

3.3.3.2 Incoming Data: Latitude, longitude, and commute time results.

3.3.3.3 Outgoing Data: Flat addresses or user-inputted locations.

3.3.4 Google Gemini API (Google Gemini for Google Cloud API and Google Generative Language API)

3.3.4.1 Purpose:

3.3.4.1.1 Enable conversational AI capabilities within FlatFinder to assist users with queries related to flat recommendations, affordability, and housing policies.

3.3.4.1.2 Provide personalised advice by interpreting user preferences and chat history.

3.3.4.2 Incoming Data:

3.3.4.2.1 User Input: Query text related to flat search or housing concerns.

3.3.4.2.2 User Preferences: Salary, CPF balance, preferred flat type, price range, etc.

3.3.4.2.3 Chat History: Past interactions for maintaining context.

3.3.4.2.4 Outgoing Data

3.3.4.2.4.1 AI-generated response providing recommendations or answering user queries.

3.3.5 Operating System Interfaces

3.3.5.1 Operating Systems Supported:

3.3.5.1.1 Windows 10/11

3.3.5.1.2 macOS Ventura and later

3.3.5.2 Purpose: Ensure compatibility with devices running these operating systems for web application functionality.

3.3.6 Frontend Frameworks

3.3.6.1 Tools Used:

3.3.6.1.1 React.js (for the web application).

3.3.6.2 Purpose: Enable dynamic user interfaces and seamless interaction with backend services.

3.3.6.3 Communication: JSON responses from APIs to populate user dashboards and search results.

3.3.7 Shared Data Across Components

3.3.7.1 User Profile Data: Shared between the frontend, backend, and database to personalise the user experience.

3.3.7.2 Flat Listings: Accessed by both the recommendation engine and the frontend interface for display purposes.

3.3.7.3 Search Parameters: Transferred from the frontend to backend APIs for processing and returned with results.

3.4 Communications Interfaces

This section describes the communication functions required for FlatFinder, including protocols, standards, and security considerations.

3.4.1 Email Services

3.4.1.1 Purpose:

3.4.1.1.1 Facilitate password recovery by sending recovery codes.

3.4.1.2 Protocol:

3.4.1.2.1 SMTP (Simple Mail Transfer Protocol).

3.4.1.3 Message Formatting:

3.4.1.3.1 Emails formatted in HTML for a professional appearance.

3.4.1.4 Security Standards:

3.4.1.4.1 TLS encryption for secure transmission of emails.

3.4.1.4.1 SPF, DKIM, and DMARC policies enforced for email authentication to prevent spoofing.

3.4.1.5 Service Integration:

3.4.1.5.1 Integration with email services such as AWS SES, SendGrid, or Gmail API.

3.4.1.6 Synchronisation Mechanisms:

3.4.1.6.1 Real-time email dispatch for critical actions like password resets.

3.4.2 Web Browser Communication

3.4.2.1 Purpose:

3.4.2.1.1 Enable the user interface for the web application, including flat search, user profile management, and real-time chat.

3.4.2.2 Protocol:

3.4.2.2.3 HTTPS for secure web communication.

3.4.2.3 Standards:

3.4.2.3.1 Adheres to modern web standards (HTML5, CSS3, and JavaScript).

3.4.2.4 Security Considerations:

3.4.2.4.1 HTTPS ensures data encryption during transmission.

3.4.3 Chat Services Integration

3.4.3.1 Purpose:

3.4.3.1.1 Enable conversational interfaces through Google Gemini for real-time user assistance.

3.4.3.2 Protocol:

3.4.3.2.1 WebSocket for maintaining persistent communication during chat sessions.

3.4.3.2.2 REST API for initiating and finalising chat interactions.

3.4.3.3 Standards:

3.4.3.3.1 Message formatting adheres to the Google Gemini API specifications.

3.4.3.4 Security Considerations:

3.4.3.4.1 API key-based authentication.

3.4.3.4.2 Data transmission encrypted using HTTPS.

3.4.4 Data Synchronisation and Transfer Rates

3.4.4.1 Data Transfer Rates:

3.4.4.1.1 Sufficient bandwidth is required for seamless operation, particularly during:

3.4.4.1.1.1 Real-time chat interactions.

3.4.4.1.1.2 Map loading for flat locations.

3.4.4.1.1.3 Image or media attachments in flat listings.

3.4.4.2 Synchronisation Mechanisms:

3.4.4.2.1 Webhooks for real-time updates from third-party APIs (e.g., HDB or CPF).

3.4.5 Communication Security and Encryption

3.4.5.1 Encryption Standards:

3.4.5.1.1 HTTPS/TLS for data in transit.

3.4.5.2 Authentication Mechanisms:

3.4.5.2.1 OAuth 2.0 for secure API access.

3.4.5.2.2 Two-factor authentication (2FA) for user accounts.

3.4.5.3 Error Handling:

3.4.5.3.1 Graceful fallback and error messages for network failures.

3.4.5.3.2 Logging mechanisms for monitoring communication errors.

By implementing these communication interfaces, FlatFinder ensures secure, efficient, and seamless interaction across all components and stakeholders.

4. System Features

This section organises the functional requirements of **FlatFinder** based on major system features. Each feature is detailed with its purpose, user interactions, and functional requirements.

4.1 Feature: Account Management

4.1.1 Description (High priority)

This feature allows users to securely create and manage their accounts, including registration, login, logout, and reset password.

4.1.2 Stimulus/Response Sequences

Stimulus: The user enters email address, and password on the registration page.

- **Response**: The system validates the details and creates a new account, assigning a unique User ID.

Stimulus: The user logs in using their email and password.

- **Response**: The system authenticates the credentials and grants access to the user's profile.

Stimulus: The user initiates the password reset process.

- **Response**: The system sends an email to the registered email address with a 6 digit pin to reset the password.

4.1.3 Functional Requirements

4.1.3.1 User Registration

4.1.3.1.1 Users shall be able to create an account by providing a valid email address password.

4.1.3.1.2 Passwords must contain minimum 8 characters

4.1.3.1.3 Passwords must contain at least 1 lowercase case

4.1.3.1.4 Passwords must contain at least 1 upper case

4.1.3.1.5 Passwords must contain at least 1 number

4.1.3.1.6 Passwords must contain at least one special character

4.1.3.2 User Login

4.1.3.2.1 Users shall be able to log in using their registered credentials (email and password).

4.1.3.3 User Logout

4.1.3.3.1 Users shall be able to securely log out from their account.

4.1.3.4 Password Management

4.1.3.4.1 Users shall be able to reset their password using a pin number that is sent to their email that makes the reset secure. Users will then have to key in the pin to reset their password

4.1.3.4.2 Passwords must contain minimum 8 characters

4.1.3.4.3 Passwords must contain at least 1 lowercase case

4.1.3.4.4 Passwords must contain at least 1 upper case

4.1.3.4.5 Passwords must contain at least 1 number

4.1.3.4.6 Passwords must contain at least one special character

4.1.3.4.7 Error message will be displayed if the key entered is not correct.

4.1.3.5 Error Handling and Feedback

4.1.3.5.1 The system shall provide error messages for invalid login attempts or invalid registration details.

4.2 Feature: Tailored Recommendations

4.2.1 Description (High priority)

This feature provides personalised housing recommendations based on user preferences, such as location, HDB type, salary, CPF OA, preferred price, which provides users loan suggestions listings that are suitable based on the stated attributes.

4.2.2 Stimulus/Response Sequences

Stimulus: The user inputs preferences such as location, HDB type, and preferred price range.

- **Response**: The system calculates an affordability range using the user's profile data and displays relevant property listings.

Stimulus: The user adjusts their financial or housing preferences.

- **Response**: The system updates the recommendations in real-time.

4.2.3 Functional Requirements

4.2.3.1 Calculate Affordability

4.2.3.1.1 The system shall calculate the grants that the user can apply for based on the salary.

4.2.3.1.2 Properties with calculated adjusted price (listing price - total grant - cpf_oa_balance) is less than or equal to the preferred price it would be in the green marker, which suggests it is affordable.

4.2.3.1.3 Properties with calculated adjusted price (listing price - total grant - cpf_oa_balance) is greater than the preferred price but less than or equal to 1.05 times the preferred price, the property will be marked with a yellow marker, indicating it is moderately affordable.

4.2.3.1.4 Properties with calculated adjusted price (listing price - total grant - cpf_oa_balance) is greater than the preferred price but less than or equal to 1.05 times the preferred price, the property will be marked with a yellow marker, indicating it is moderately affordable.

4.2.3.2 Display Recommendations

4.2.3.2.1 The system shall display property recommendations coloured coded markers on the map based on calculated affordability using user preferences.

4.2.3.3 Update User Preferences

4.2.3.3.1 Users shall be able to modify their user preferences and see real-time updates in recommendations.

4.2.3.3.2 Upon keying any invalid user preferences or incomplete preferences the system will display an error message to prompt the user to key within the valid range.

4.3 Feature: Listings Display

4.3.1 Description (High priority)

This feature provides an intuitive display of property listings on a map with interactive options such as affordability-based colour coding, street view, and amenities visualisation.

4.3.2 Stimulus/Response Sequences

Stimulus: The user views search results based on their preferences and the estate they choose to filter.

- **Response**: The system displays the results using a colour-coded map indicating property affordability of the chosen estate. Green is most affordable, yellow, moderately affordable and red being not affordable.

Stimulus: The user clicks a property marker on the map.

- **Response**: The system shows a pop-up sidebar with property details, nearby amenities on the left side of the screen

4.3.3 Functional Requirements

4.3.3.3 Property Details

4.3.3.3.1 The system shall allow users to view property details in a pop-up sidebar on the right such as property address, cost of property, size, housing type(eg, executive, 5-room, etc.), amenities and its filter according to type (eg, schools, restaurants, etc.)

4.3.3.4 Map view

4.3.3.4.1 The map shall display nearby amenities.

4.3.3.4.2 The user shall be able to view the street view just by placing their selected marker/ping (In the form of a yellow man).

4.3.3.5 Default Display

4.2.3.5.1 When no user preferences are indicated, all available properties shall be displayed as red markers.

4.3.3.6 Dynamic Updates

4.3.3.6.1 Any change in user preferences will change the colour-coded map makers on the map as it calculate a different set of housing affordability options

4.4 Feature: Property Management and Bookmarking

4.4.1 Description (Medium priority)

This feature allows users to manage and bookmark their favourite properties for future reference, enabling them to save, sort, and compare listings.

4.4.2 Stimulus/Response Sequences

Stimulus: The user clicks on bookmarks page in the home page

- **Response**: The system will direct them to the bookmark page with all the saved listings that the user made.

Stimulus: The user clicks the heart symbol to save a property to their favourites.

- **Response**: The property is saved to the user's profile, and the heart symbol is highlighted.

Stimulus: The user clicks the heart symbol on a saved property.

- **Response**: The property is removed from the user's favourites, and the heart symbol is unhighlighted

Stimulus: The user clicks compare property and picks another property marker

- **Response**: The system shows a pop-up sidebar on the right side of the screen with property details, nearby amenities.

4.4.3 Functional Requirements

4.4.3.3 Save Property

4.4.3.3.1 The system shall offer users the ability to save properties by clicking the heart symbol.

4.4.3.4 Unsaved Property

4.4.3.4.1 Users shall be able to remove properties from their saved list by clicking the heart symbol again.

4.4.3.5 Compare Properties

4.4.3.5.1 The system shall allow users to compare selected properties side-by-side by displaying multiple pop-up sidebars simultaneously.

4.4.3.6 Removed Saved property

4.4.3.6.1 The system shall allow users to remove listings in the bookmark page by editing and removing individual listings at the time.

4.5 Feature: Data Management

4.5.1 Description and Priority

This feature manages the retrieval, processing, and display of resale unit data from government databases such as HDB and URA, ensuring users access the most accurate and up-to-date listings.

4.5.2 Stimulus/Response Sequences

Stimulus: The system automatically triggers the refresh when they calculate suitable listings for the user.

- **Response**: The system retrieves the latest resale unit listings from external government databases and updates the displayed data for users.

4.5.3 Functional Requirements

4.5.3.1 Data Retrieval

4.5.3.1.1 The system shall retrieve resale unit data from government databases, such as HDB and URA, to ensure accuracy and up-to-date information. It only retrieves data up to the previous month as the current month the api has not published the info.

4.5.3.2 Data Refresh

4.5.3.2.1 The system shall update the resale unit listings only when the user finds suitable properties.

4.5.3.2 Failed to retrieve data

4.5.3.2.1 The system will display an error message (“Failed to find suitable listings”).

4.6 Feature: AI Chatbot Housing Recommendation

4.6.1 Description *(Medium Priority)*

This feature enables users to interact with an AI chatbot on the "Edit Preferences" page to receive personalised housing suggestions or ask general housing-related questions. The chatbot uses the user's preferences stored in the Firestore database to tailor its responses.

4.6.2 Stimulus/Response Sequences

Stimulus: The user navigates to the "Edit Preferences" page.

- **Response**: The chatbot appears as an empty chat window with intuitive prompts like "Ask me about housing suggestions!" or "Need help finding affordable homes?".

Stimulus: The user initiates a conversation by asking, "What are my housing options in Bedok?"

- **Response**: The chatbot retrieves the user's preferences (e.g., salary, CPF OA balance, preferred location) from the database and provides tailored housing suggestions.

Stimulus: The user minimises the chat window.

- **Response**: The chatbot retains the chat history, allowing the user to continue the session when the window is reopened, provided the user stays on the same page.

Stimulus: The user updates preferences while chatting.

- **Response**: If live updates are supported, the chatbot recalculates suggestions in real-time and adjusts responses accordingly.

Stimulus: The user asks a generic question like "What’s the best area for families?"

- **Response**: The chatbot responds with general insights while respecting user privacy and relevance.

Stimulus: A connectivity issue prevents the chatbot from accessing the database.

- **Response**: The chatbot notifies the user that it is currently unavailable and suggests trying again later.

4.6.3 Functional Requirements

4.6.3.1 Chat Interface

4.6.3.1.1 The chatbot shall display intuitive prompts to guide users in initiating conversations.

4.6.3.1.2 The chatbot window shall retain chat history while the user remains on the "Edit Preferences" page.

4.6.3.2 Personalised Responses

4.6.3.2.1 The chatbot shall retrieve stored user preferences to provide tailored housing suggestions.

4.6.3.2.2 The chatbot shall adapt responses in real-time if user preferences are updated during a session (if supported).

4.6.3.3 General Queries

4.6.3.3.1 The chatbot shall handle general housing-related queries and provide insights.

4.6.3.4 Session Management

4.6.3.4.1 The chatbot session shall persist while the user stays on the "Edit Preferences" page.

4.6.3.4.2 The chatbot session shall reset when the user navigates away from the "Edit Preferences" page.

4.6.3.5 Error Handling and Feedback

4.6.3.5.1 The chatbot shall notify users if it is unable to retrieve user preferences or if there is a connectivity issue.

4.6.3.5.2 The chatbot shall provide fallback responses, offering generic suggestions when specific preferences cannot be retrieved.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 Performance

5.1.1.1 Stimulus/Response Sequences

Stimulus: The system is subjected to a load of 500 concurrent users, with 1 new user starting every second.

- **Response**: The system must handle all requests within 4 seconds

5.1.1.2 The system will be tested with a total of 500 concurrent users, with a user spawning rate of 1 user per second.

5.1.1.3 The system's average response time should be under 3 seconds

5.1.2 Scalability

5.1.2.1 Stimulus/Response Sequences

Stimulus: The system is subjected to a load of 1000 concurrent users performing searches.

- **Response**: The system's response time increases by no more than 10% compared to the response time with 500 users.

5.1.2.2 Load testing shall be conducted with 1000 concurrent users

5.1.2.3 The response time for 1000 users shall be compared with that of 500 users to ensure the increase is within 10%

5.1.3 Intuitive Navigation

This feature ensures that users can access any feature within three clicks from the homepage, improving the case of navigation.

5.1.4 Ease of Use

This feature ensures that 80% of first-time users can complete a flat search within 5 minutes without external assistance, supporting a smooth onboarding experience.

5.1.4.1 Stimulus/Response Sequences

Stimulus: A first-time user begins a flat search

- **Response**: The system enables the user to complete the search within 5 minutes

5.1.4.2 Usability testing shall be conducted with 50 first-time users.

5.1.4.3 At least 80% users must complete a flat search within the 5-minute timeframe.

5.1.5 Supportability

This feature ensures that the application is fully functional on all major browsers, providing a consistent experience across platforms.

5.1.5.1 Stimulus/Response Sequence

Stimulus: The user accesses the system using different browsers.

- **Response**: The system functions properly, with all features accessible and responsive across platforms

5.1.5.2 Validation tests shall ensure that all core functionalities work on Chrome, Safari and Edge.

5.1.5.3 The layout shall adapt to the different browsers, with no horizontal scrolling required to access all functions of the system.

5.2 Safety Requirements

5.2.1 Data Integrity and Consistency

5.2.1.1 Requirement: Ensure that user data (e.g., preferences, saved listings) is not lost or corrupted due to system errors or unexpected shutdowns.
5.2.1.1 Implementation: Implement automatic data backups at regular Intervals.

5.2.2 Compliance with Regulatory Safety Standards

5.2.2.1 Requirement: Adhere to the Personal Data Protection Act (PDPA) to safeguard user information.
5.2.2.2 Implementation: Implement measures to safely handle and dispose of sensitive data when no longer needed (clearing chat history after exiting user preference page).

5.2.3 Handling of Incorrect or Corrupted Inputs

5.2.3.1 Requirement: Prevent system crashes or unexpected behavior due to invalid or corrupted input data.
5.2.3.2 Implementation: Validate all user inputs such as user preferences on client side.

5.2.4 Safe Interaction with External APIs

5.2.4.1 Requirement: Ensure that data retrieved from external systems such as Google Maps API is accurate and does not harm the system's functionality.
5.2.4.2 Implementation: Implement timeouts and retries for API calls to handle cases where the external service is unresponsive or slow.

5.3 Security Requirements

5.3.1 Description and Priority

This feature ensures the security of user accounts through strong password policies, account lockout measures, and identity verification processes.

5.3.2 Stimulus/Response Sequences

Stimulus: The user creates a new password during account registration or password reset.

- **Response**: The system validates the password strength and prompts the user if the password does not meet the requirements.

5.3.3 Stimulus/Response Sequences

Stimulus: The user confirms the newly created password during account registration or password reset by re-entering it.

- **Response**: The system compares the confirmed password with the initial password input. If they do not match, the system prompts the user to correct the

passwords, ensuring that users do not unintentionally create accounts with incorrect passwords due to typing errors.

5.3.4 Functional Requirements

5.3.4.1 Strong Password Policy

5.3.4.1.1 The system shall enforce a strong password policy requiring at least 8 characters, with a combination of uppercase letters, lowercase letters, numbers, and special characters.

5.3.4.2 Password Confirmation

5.3.4.2.1 The system shall require users to enter their password twice for confirmation during account creation and password changes.

5.3.4.3 Account Lockout

5.3.4.3.1 The system shall lock the user's account after 5 consecutive failed login attempts for a duration of 15 minutes.

5.3.4.4 Lockout Notification

5.3.4.4.1 The system shall notify the user via email when their account is locked and provide instructions for unlocking it.

5.3.4.5 Account Unlocking

5.3.4.5.1 The system shall allow users to unlock their accounts by verifying their identity through a secure link sent to their registered email address.

5.4 Software Quality Attributes

The following quality attributes are critical to ensuring the success and usability of the FlatFinder system for both customers and developers:

5.4.1 Usability

5.4.1.1 Description: The system should be intuitive and easy to use for all user classes, including first-time users.

5.4.1.2 Metrics:

5.4.1.2.1 Time taken to perform a flat search should be under 2 minutes.

5.4.1.2.2 At least 90% of users should complete their first search without external guidance.

5.4.1.2.3 User interface elements must follow consistent design standards and accessibility guidelines.

5.4.2 Reliability

5.4.2.1 Description: The system must operate consistently without failures, especially during high-demand periods.

5.4.2.2 Metrics:

5.4.2.2.1 99.9% uptime, excluding scheduled maintenance.

5.4.2.2.2 System recovery time after a failure should not exceed 5 minutes.

5.4.2.2.3 Data accuracy for flat recommendations must exceed 95%.

5.4.3 Performance

5.4.3.1 Description: The system should deliver results promptly, ensuring smooth user interactions.

5.4.3.2 Metrics:

5.4.3.2.1 Search results should be displayed within 3 seconds for most queries.

5.4.3.2.2 Chat responses from Google Gemini should appear within 1 second of user input.

5.4.3.2.3 Server response time should be under 200ms for API calls.

5.4.4 Scalability

5.4.4.1 Description: The system should handle increasing user loads without significant degradation in performance.

5.4.4.2 Metrics:

5.4.4.2.1 The system should support up to 1000 concurrent users.

5.4.4.2.2 Performance degradation under load should not exceed 10%.

5.4.5 Security

5.4.5.1 Description: User data must be protected against unauthorised access, ensuring compliance with relevant laws and standards.

5.4.5.2 Metrics:

5.4.5.2.1 All communications must use HTTPS with TLS 1.3.

5.4.5.2.2 The system should pass regular penetration testing and comply with the PDPA (Personal Data Protection Act, Singapore).

5.4.6 Interoperability

5.4.6.1 Description: The system must seamlessly integrate with external APIs and services, such as HDB, Google maps, and Google Gemini.

5.4.6.2 Metrics:

5.4.6.2.1 External APIs should respond to FlatFinder queries with a success rate exceeding 98%.

5.4.6.2.2 FlatFinder must handle API updates or schema changes with minimal manual intervention.

5.4.7 Maintainability

5.4.7.1 Description: The system should be easy to maintain and update by developers.

5.4.7.2 Metrics:

5.4.7.2.1 Average time to fix critical bugs should not exceed 24 hours.

5.4.7.2.2 New features should be deployable with minimal downtime (< 30 minutes).

5.4.8 Adaptability

5.4.8.1 Description: The system must be flexible enough to accommodate new features and changes in user preferences or housing policies.

5.4.8.2 Metrics:

5.4.8.2.1 Ability to incorporate new data sources (e.g., third-party APIs) within 2 weeks.

5.4.8.2.2 Modular architecture to facilitate updates and scaling.

5.4.9 Testability

5.4.9.1 Description: The system must be designed to facilitate rigorous testing of all components.

5.4.9.2 Metrics:

5.4.9.2.1 At least 90% code coverage in automated tests.

5.4.9.2.2 All critical functions should pass 100% of test cases in acceptance testing.

5.4.10 Portability

5.4.10.1 Description: The application should work across multiple platforms.

5.4.10.2 Metrics:

5.4.10.2.1 Support for major operating systems (Windows, macOS, Android, iOS).

5.4.10.2.2 Support across major browsers (Safari, chrome, edge).

By ensuring these attributes, FlatFinder will meet high standards for functionality, user experience, and long-term viability.

5.5 Business Rules

The following business rules define the operational principles of the FlatFinder system, governing user roles, permissions, and interactions:

5.5.1 User Roles and Permissions

5.5.1.1 General Users (Consumers):

5.5.1.1.1 Can search for flats based on preferences such as location, budget, and flat type.

5.5.1.1.2 Can save, update, and delete their search preferences.

5.5.1.1.3 Can view detailed recommendations tailored to their financial situation and housing needs.

5.5.1.1.4 Must log in to access personalised recommendations and saved searches.

5.5.1.1.5 Cannot access or modify administrative or developer tools.

5.5.1.2 Administrators:

5.5.1.2.1 Can manage user accounts, including resetting passwords.

5.5.1.2.2 Can update database entries for flat listings or housing policy information.

5.5.1.2.3 Have the ability to monitor and audit user activity logs for compliance or debugging purposes.

5.5.1.3 Developers (System Maintenance):

5.5.1.3.1 Can deploy updates, add new features, or fix system bugs.

5.5.1.3.2 Must not modify user data unless explicitly authorised by the user or administrator.

5.5.2 Search and Recommendation Rules

5.5.2.1 Users must provide a preferred location

5.5.2.2 Recommendations will be colour coded based on user affordability

5.4.3 Data Privacy and Security

5.4.3.1 User data, including financial details and preferences, must not be shared with third parties without explicit consent.

5.4.3.2 Passwords must be encrypted and cannot be accessed by administrators or developers.

5.4.3.3 Email addresses and contact details are only used for account recovery, or user queries.

5.4.4 User Interaction Rules

5.4.4.1 Users who fail to provide accurate preferences may receive general recommendations instead of personalised results.

5.4.4.2 Chatbot assistance (powered by Google Gemini) will prioritise answering flat-related queries. Non-relevant questions will trigger a polite redirection.

5.4.5 Housing Policy Updates

5.4.5.1 FlatFinder must automatically update HDB policies from trusted government data sources.

5.4.5.2 Changes in policies must be reflected in recommendations within 24 hours of the update.

5.4.6 Compliance Rules

5.4.6.1 All operations must adhere to Singapore's Personal Data Protection Act (PDPA).

5.4.6.2 Any changes to business rules or terms of service must be communicated to users at least 30 days before implementation.

6. Data Dictionary

User and Account Management

Field Name	Data Type	Data Format	Description
Manage Account	Varied		Functionalities related to updating and maintaining user account details, including password, salary, CPF balance, and housing preferences.
CPF Balance	Number	NNNNNN	The current amount in the user's CPF (Central Provident Fund) account, which can be used towards the purchase of a resale flat.
Housing Preferences	Varied		The user's desired HDB type, location, and price range for search purposes.
Housing Loan Type	String		The category of loan the user intends to take (e.g., bank loan, HDB loan).
Email	String		The user’s email address, used for account management tasks such as password recovery.
Password	String	NNNNNNNN	A secure string of characters used by users for authentication. It must meet the system's password policy (minimum 8 characters with one upper case, one lower case, one digit and one special character.)
Profile Information	Varied		User’s personal details, such as salary, CPF balance, and housing preferences, which can be updated from their account.
ChatRequest	Object		A request object containing user input, preferences, and chat history. This data structure is used to handle a conversation with Google Gemini.

Formatted Response	String		The response generated by the Google Gemini model, formatted for user-friendly display. Additional formatting includes replacing markdown symbols and adding bullet points.
Preferences Missing	Boolean	True/False	Flag indicating if the user preferences are missing. If true, the system prompts the user to input their preferences.

Property Listing and Management

Field Name	Data Type	Data Format	Description
Contact Number	String	NNNNNNNN	The phone number of the landlord or property manager.
Customised Property Recommendations	Varied		A list of property listings that are generated based on the property listings, user information and housing preferences, using our formula.
HDB Location	String	Street number, Street name, Postal code	The specific estate where the resale flat is located.
HDB Type	String		The size or type of the flat (e.g., 3-room,4-room).
Postal Code	String	NNNNNN	The postal code of the resale flat.
Price	Number		The selling price of the HDB resale flat.
Price Range	Number		The range of prices inputted by the user.
Property ID	String		A unique identifier assigned to each property listing.

Saved Listings	Varied		Properties that the user has bookmarked for future reference.
Street Name	String		The name of the street on which the resale flat is located.
Street Number	String	NNN	The building number of the resale flat.

Affordability

Field Name	**Data Type**	**Data Format**	**Description**
Affordability	Number	NNNNN	A computed metric to determine how much the user can afford for a resale flat.
Green Affordability Category	Boolean	True/False	Indicates flats within the user's preferred price range.
Orange Affordability Category	Boolean	True/False	Indicates flats slightly above the user's preferred price range.
Red Affordability Category	Boolean	True/False	Indicates flats outside the user's preferred price range.
Total grant	Number	NNNNN	The total financial grant amount that a user qualifies for, calculated based on their salary and the type of flat they are purchasing. It is the sum of the Singles Grant and the Enhanced Grant, if applicable.
Singles grant	Number	NNNNN	The grant amount given to single applicants based on their salary and flat type. It ranges between 0 and 40,000 SGD depending on the conditions.
Enhanced grant	Number	NNNNN	The grant amount is given to eligible applicants who have a salary less than or equal to 4500 SGD, providing an additional benefit up to 60,000 SGD.

Search and Filtering

Field Name	Data Type	Data Format	Description
Comparison	Function		Feature allowing users to view multiple properties side-by-side.
Search Criteria	Object		Filters and inputs provided by the user to find resale flats.
Search Functionality	Function		Allows users to input attributes like location for flat searches.
Search Results	Array		The list of resale flats displayed based on search criteria.

System and Performance

Field Name	Data Type	Data Format	Description
Browser Compatibility	Boolean	True/False	Ensures the application works across different browsers.
Data Refresh	Function		The process of updating property listings from external databases.
Downtime	Number	NNNN	The period when the system is unavailable.
External Databases	Array		Sources such as HDB and URA from which flat data is retrieved.
Response Time	Number	NNNN	The time taken for the system to return search results.
Scalability	Number	NN	The system's ability to handle increased load.
System Reboot	Function		The process of restarting the system.

System Uptime	Number	NNN	The percentage of time the system remains operational.
Zero-Downtime Updates	Boolean	True/False	Capability to implement system updates without downtime.