Software Requirements Specification for Nestling

Version 1.0 approved

Prepared by:

Huai Zhi U2221000J

Nicole Kaira Almonte Imatong U2222544H

Panduga Pranavi U2222479D

Ravipudi Abhijeet Chowdary U2220816C

Aaron Jayadhitya Fernandez U2221487F

Sharmilla D/O Ramesh U2221899G

SCSD SC2006 Group 4

15 March 2024

Table of Contents

1. Introduction

1.1 Purpose

We are Nestling. We have developed a web application that helps prospective private home buyers or investors find information on estimated pricing and evaluations of their desired housing properties, generated using artificial intelligence, and up-to-date data on current properties on the market, based on their specifications.

1.2 Document Convention

Main Header Font: Times

Main Header Font Size: 18

Subsection Header Font: Times

Subsection Header Font Size: 14

Sub-subsection Header Font: Times

Sub-subsection Header Font Size: 12

Content Font: Times

Content Font Size: 11

Image Description Font: Times

Image Description Font Size: 8

1.3 Intended Audience and Reading Suggestions

The Software Requirement Specification document is created for public consumption. The document provides high level analysis of the software development process and goes through each stage of the software development life cycle of our website application in great detail for developers to understand the procedural steps of this project. This document contains requirements elicitation that contains detailed information on the requirements of this website application from the users, developers and other various stakeholders. It also contains class diagrams, use case diagrams and sequence diagrams that refine the user experience into great details and splits the development into achievable and deliverable segments. For those interested in the user aspect of the application, especially investors and the marketing team, the functionalities of the application are explained in Section 4, System Features, and the details on the user interface would be available in Section 3.1, User Interface and in certain portions of Section 4.

1.4 Product Scope

Nestling allows users to input attributes of their desired property, such as the location, proximity to amenities, number of rooms and gross floor area, in order to generate an estimated price. The estimated price is predicted using our proprietary artificial intelligence (AI), PricePointAI. They can also save their desired property attributes into our application database. Users can also look up listings of properties that are currently available on the market. Users are able to compare the current listings with the desired property attributes to see how different the available property is from their dream property and the web application can also generate a review on the comparison using a chatbot that makes use of artificial intelligence to produce an output, called CompareAI. Nestling also allows users to save current property listings in the application database to be viewed later on. Users can also make side-by-side comparisons of two saved listings and make use of CompareAI to provide a comprehensive assessment of the two properties being compared. Additionally, as secondary features, the user is able to view news and articles on the property market.

1.5 References

1. Urban Redevelopment Authority (URA), Private Residential Property transactions, retrieved from https://www.ura.gov.sg/uraDataService/invokeUraDS?service=PMI_Resi_Transaction&batch=1
2. Data.gov.sg, List of Mass Rapid Transport (MRT) stations, supermarkets, primary schools, secondary schools, parks and malls, retrieved from https://beta.data.gov.sg/
3. OneMap API, location verification and latitude and longitude retrieval accessed via https://www.onemap.gov.sg/api/common/elastic/search?searchVal={searchString}&returnGeom=Y&getAddrDetails=Y&pageNum=1
4. OpenAI, query response via https://platform.openai.com/docs/api-reference/introduction?lang=python

It is worthy to recognise that the datasets from Data.gov.sg are in the format of .csv, .json, .kml or .geojson files.

2. Overall Description

2.1 Product Perspective

Nestling is a self-contained web application used on laptops and desktops to showcase to prospective private home buyers their desired property listings. The application utilises data from the URA API from data.gov.sg, OneMap API, OpenAI API, Flask Server API and Supabase API. The following were used to implement a comprehensive application for buyers to make informed decisions along with CompareAI and PricePointAI of desired properties selected by users from past listings.

2.2 Product Functions

Nestling has the following functions:

1. Credentials: users must sign up if they do not have a valid account and existing users can log in using their log-in credentials.
2. Desired Property Function: users can select their desired property attributes and save them to search for their properties later.
3. Property Recommendations: the web application will perform the percentage match between the user's search and the current properties data.
4. AI Comparison Analysis of properties: users can compare properties using the CompareAI.
5. Saving favourite properties: users can save their favourite properties for shortlisting their dream home.

2.3 User Classes and Characteristics

Nestling is suitable for private home buyers who want to know the general market of private housing in Singapore. The application has clear interfaces to enable users ages 21 and above to easily interact with the application.

Nestling requires personal information such as email, username, password and mobile number to create an account. As such, the user requires a valid email and mobile number. Additionally, the user interface is in simple terminology thus, basic English proficiency and digital literacy are required.

2.4 Operating Environment

For the software environment, our web application is built using the React JavaScript framework and supports various modern web browsers including Google Chrome, Safari, FireFox, Microsoft Edge, etc. As for the backend, we used supabase as our database which operates within a Node.js environment. Moreover, Nestling features functions such as "Generate Estimated Price" (PricePoint AI) and "Comparison Analysis"

(CompareAI) that utilise AI. Therefore, to prevent any conflict, we established 2 virtual environments, one for each AI feature, where we installed all the python library dependencies in. This setup allows the AI to operate in the background concurrently with the main React application. For the PricePoint environment, we created a virtual environment called "pricegeneration" with libraries such as pickle, numpy and scikit-learn. For the CompareAI environment, we created a virtual environment called "ai_env" where we installed all our python libraries such as Langchain, OpenAI, supabase, flask, tiktoken, etc. Moreover, to establish communication between the frontend and backend, we create a flask server to allow seamless data transfer and manipulation.

For the hardware environment, the web application can be hosted on any client device as it is designed to be responsive and functional across various devices including laptops, desktops, tablets. In addition, Nestling runs on operation systems such as macOS, iOS and Android OS, covering a wide range of technology accessibility and user preferences. For network wise, a stable and fast internet connection is needed to guarantee real-time data syncing and a smooth user experience.

For the front-end environment, we employed the use of the React environment for our application. Feeling that our application is well-suited to be viewed on a web browser due to the multitude of listings, we designed our application based on the dimensions of a typical desktop (1440 by 1024) for optimal viewing experience. However, we also made sure the elements are viewable on smaller devices for the user's convenience. We employed the use of JSX (a combination of Javascript and XML) for ease of integrating display elements with the functions of the elements, avoiding confusion.

2.5 Design and Implementation Constraints

Some limitations and considerations need to be taken into account when developing the Nestling web application. The application will extract and display the real-time on-sale properties from the URA and will be updated weekly on Tuesdays and Fridays. Specific technologies and tools, like ReactJS for the frontend development, and databases for data storage such as Supabase are utilised and may be requisitioned by project requirements. As for the AIs, the two virtual environments have to concurrently run with the React application to display the PricePointAI for desired properties and the CompareAI of two properties. Additionally, the users must be connected to the Internet to login/sign up and to view the properties.

2.6 User Documentation

Introduction: Our web application 'nestling.ai' aims to help buyers looking for private homes find an ideal property for purchase.

Getting Started: Once on the landing page, a user can click 'sign up', as shown below, to create an account. After the account has been created, the user can then proceed to explore the rest of the application.

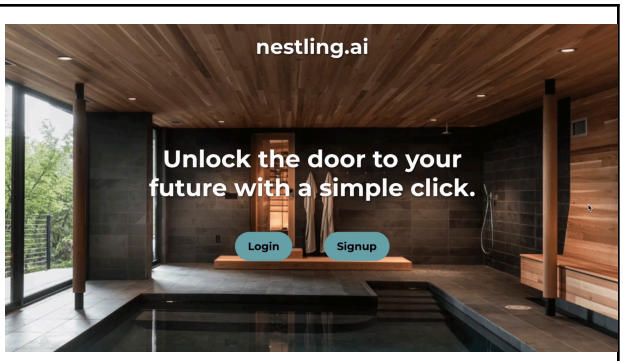


Figure 2.6.1 : Landing Page

'Desired Property' - This feature allows users to input the attributes they would like their desired property to have.

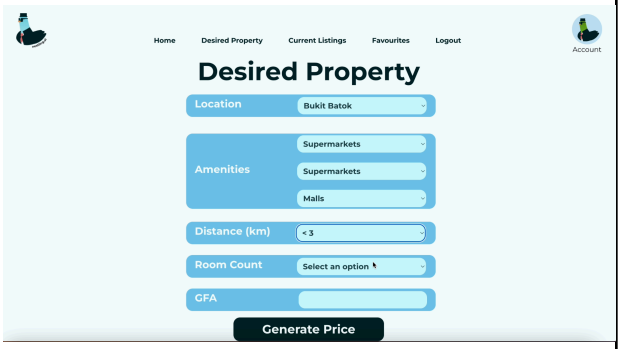


Figure 2.6.2: Desired Property Page

'Current Listings' - This feature allows users to key in certain attributes of their liking and then search for properties on the market that match these attributes. Alternatively, users can choose to click on the 'search using desired property attributes' button in order to see properties that match their desired property attributes

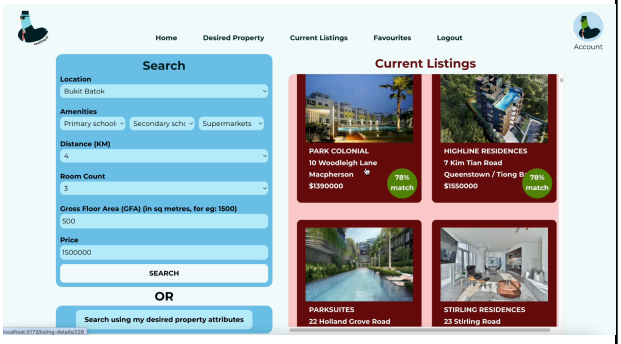
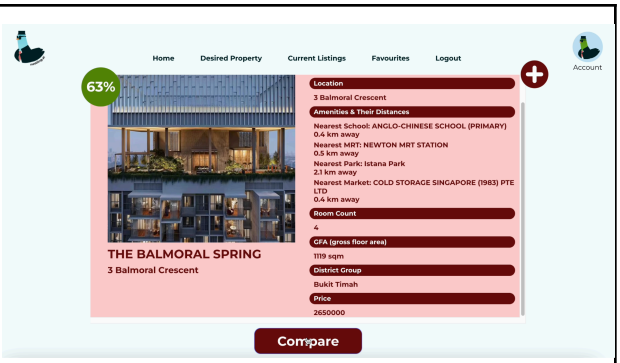
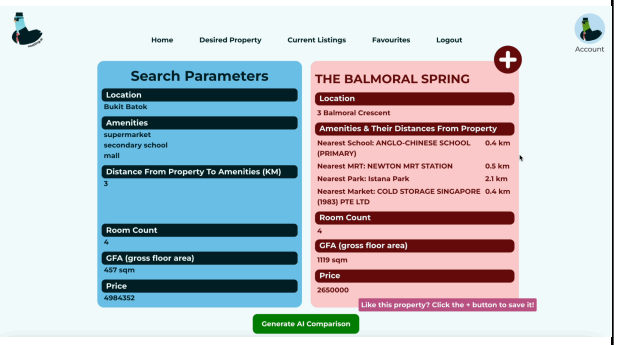
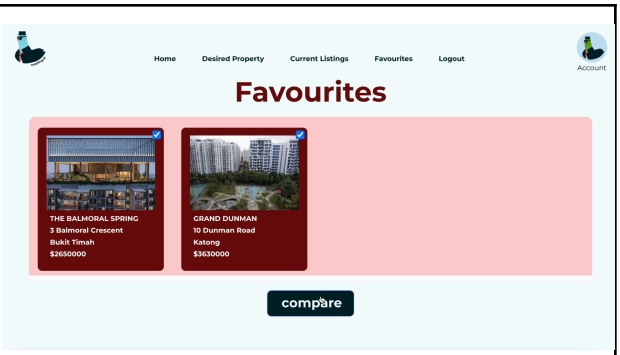
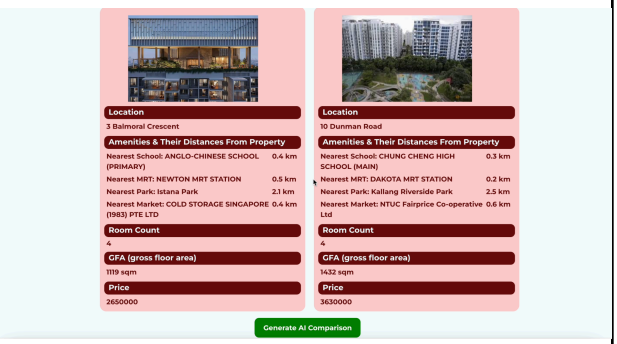
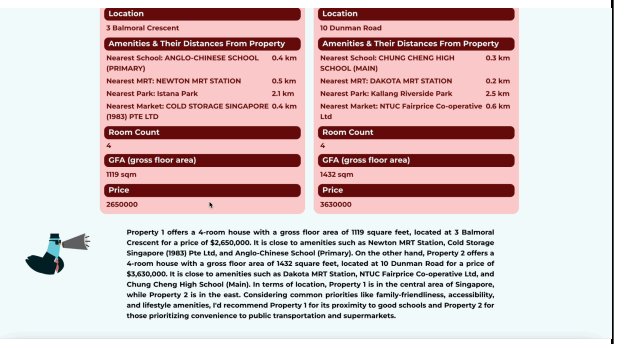


Figure 2.6.3: Current Listings page

'Listing Details' - Users can view further details of a property of their choosing by simply clicking on the property panel. They can then click the '+' button on the top right of the panel to save the property to their favourites	 Figure 2.6.4: Listing Details
'Compare(Current Listings)' - Users can view a specific property side by side with their specified search parameters or their desired property attributes(depending on which one they selected). Users can then click on the 'Generate AI comparison' button to generate a text comparison of the property they have selected and their search parameters/desired property attributes	 Figure 2.6.5: Current Listings Compare Page  Figure 2.6.6: Current Listings AI output

'Favourites' - Users can click on the 'Favourites' tab to view all their saved properties. On this page, they can choose 2 or more properties to compare by clicking the 'Compare' button	 Figure 2.6.7: Favourites page
'Compare (Favourites)' - Users can click on the 'compare' button to view a side by side comparison of the two selected properties. Users can then click on the 'Generate AI comparison' button in order to generate a text comparison of the two properties	 Figure 2.6.8: Favourites Compare function  Figure 2.6.9: Favourites Compare AI output

2.7 Assumptions and Dependencies

Assumptions:

- Our project assumes the continued availability and reliability of the URA API, Supabase API, OpenAI API, OneMap API and Flask API throughout the development and deployment phases. Any

unexpected downtime or changes to these APIs could impact the functionality and performance of our application.

- We assume that Supabase will continue to provide robust backend database management services. Any changes to the Supabase API or its underlying infrastructure could require adjustments to our application's database operations and functionality.

Dependencies:

- Our project relies heavily on the Supabase API for backend database management. Any changes or disruptions to the Supabase API could directly impact our application's ability to store and retrieve data, affecting its overall functionality.

- Additionally, our project depends on Python data science libraries like scikit-learn for creating machine learning models. The availability and compatibility of these libraries are essential for developing and deploying our data science functionalities.

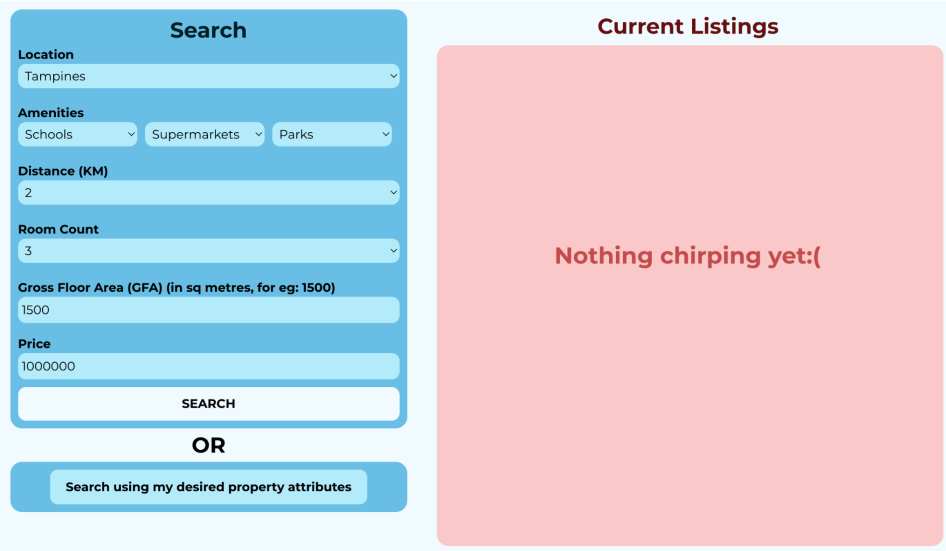
3. External Interface Requirements

3.1 User Interfaces

The design of the application as a whole uses vibrant, yet muted colours to attract user's attention but not strain their eyes. We used a multitude of design principles and theories when designing our application (BairesDev Editorial, n.d.), but these two theories stand-out:

3.1.1 Gestalt Psychology Theory:

Humans perceive visual elements in groups, rather than individual components. Grouping ideas together works in terms of similar colour palettes or patterns. Leveraging on this principle, we used three major colour palettes for our application - a green-based colour palette, a pink-based one and a cyan-based one, each associated with a different function. The cyan based colour palette was used for the Desired Property related features, while the pink based colour palette was used for the Current Listing related features and the Favourites features. This was to clearly segregate between the different features. Most of the components are in modular formats, making it easier for users to view as well.



Figures 3.1.1 & 3.1.2: Example of the different colour palettes

3.1.2 Impact of Attention on User Experience:

By having push notifications and little pop-ups scattered throughout our application, we draw attention to certain features in the application, working on the principle of selective attention. Despite being a useful tool, these pop-ups are scarcely used in application to prevent viewing fatigue.

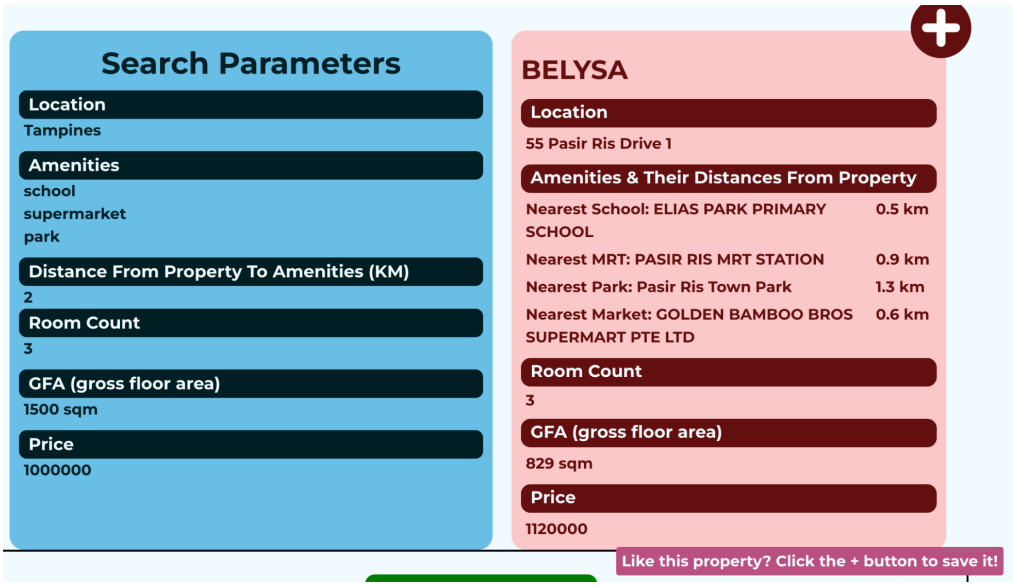


Figure 3.1.3: Pop-up used to draw attention to the ‘save’ button at the top of the screen

In terms of standard features, we used a modular design, depicted in the figures above, to contain the information, making it easier for users to read the information with the label and information pairs. Furthermore, we developed common components for shared use amongst the different features in the document, for convenience. In the figures below, our Navigation Bar and Listing Panel components are used

across multiple applications. We further customised the Listing Panel components to display slightly different versions of the components dependent on the file.

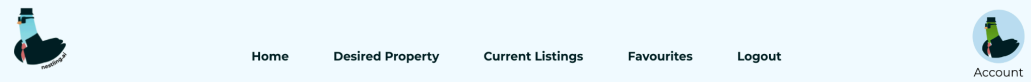


Figure 3.1.4: Navigation Bar Function

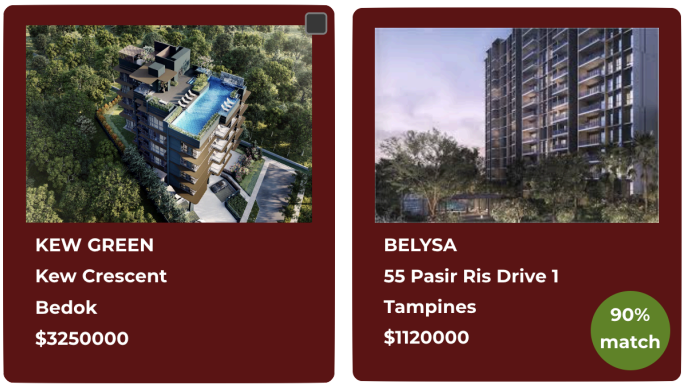


Figure 3.1.4: Two different implementations of the Listing Panel components

The right side panel depicts the implementation of the Listing Panel in the Favourites screen, without the percentage match bubble and with a checkbox instead, while the left side panel depicts the implementations of the Listing Panel in the Current Listings screen, with the percentage match bubble.

3.2 Hardware Interfaces

3.2.1 Client-side Requirements

Nestling supports all desktop platforms. The desktop must have a web browser that can support HTML, CSS and JavaScript. Additionally, the device must have a stable internet connection in order to transmit and receive data from the server.

3.2.2 Server-side Requirements

The Nestling back-end server must be hosted and run on a server-computer. The back-end server will perform Create, Read, Update, Delete (CRUD) operations on the back-end database. The database must be hosted and run on a server-computer in the cloud.

3.3 Software Interfaces

3.3.1 Software Components and Versions

Nestling is a web application that utilises a back-end server and database to handle data storage and authentication services required in the system features. The front-end is implemented using React.js using

the Vite development build tool. The back-end server and database is built and implemented primarily using Supabase for authentication and storage of information such as user info and property listings. Flask was also used to host server-side python scripting in order to run the AI features for both price generation and virtual real estate agent property comparison.

The application router paradigm uses Client-Side Rendering (CSR). Pages are rendered on the client side rather than the server side which brings several advantages, such as smoother interactivity, faster page loads, and more dynamic content. In React, this is achieved by updating the Virtual Document Object Model (DOM), resulting in what is known as a Single Page Application (SPA). We implemented CSR using a React library known as React Router.

3.3.2 Software Architecture

Nestling is based on the Model-View-Controller (MVC) architectural pattern that can organise the software components and ensure a separation of responsibilities. The MVC pattern helps to structure the codebase, making it more modular, scalable, and maintainable.

The model layer represents the data and business logic of the application. For Nestling, the model would include components responsible for managing data related to properties, user profiles, authentication and other relevant information. This layer can interact with APIs to fetch and update data and encapsulates the logic for data manipulation, validation and storage.

The view layer is responsible for presenting the data to the users and receiving their input. The layer includes components related to the user interface, such as pages, components, and templates for displaying information about properties, photos and user profiles. It should be designed to be responsive and user-friendly, ensuring a positive experience for users interacting with the application.

The controller layer acts as an intermediary between the model layer and the view layer. Nestling uses the controller layer to handle user input, processes requests, and updates the model layer accordingly. It receives input from the user interface (such as attributes in the filter form for current listings), invokes the appropriate methods in the model layer to perform the necessary actions (such as percentage match logic), and updates the view layer to reflect the changes (renders list of properties generated in descending order of percentage match value).

3.4 Communications Interfaces

The Nestling web application communication architecture must follow a client-server model. Nestling will communicate with the OneMap API, URA API, Supabase API, Flask API and OpenAI API using HTTPS protocol to ensure data security during transmission. Requests and responses will be in JSON format, following RESTful conventions. Before any request is made, all the APIs need to be initialised with an API key first to access the service securely.

URA API is used to fetch the property listings dataset. Supabase API performs operations on the fetching of user profiles, authentication and desired property attributes and saved properties. OneMap API performs operations to fetch map and geocoding information such as latitude and longitude. OpenAI API is used to perform comparison analysis on the properties retrieved from Supabase API to provide the user virtual real estate agent services. Flask API is used to invoke server-side python scripting to pass the property data to OpenAI API for comparison.

4. System Features

4.1 Credentials

4.1.1 Description and Priority

The Credentials feature includes the authentication functionality which allows users to log in, sign up, and reset their password. This feature is of high priority as it is fundamental to the user experience and security of the application.

4.1.2 Stimulus/Response Sequences

Sign-up: User clicks the 'sign up' button, enters their details and fills up the required fields. The system then validates the information, creates a new user account and logs the user in (Figure 4.1.2.1 & Figure 4.1.2.2).

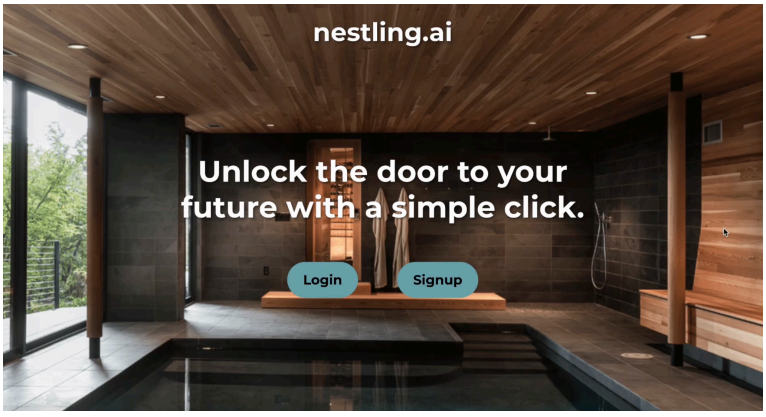


Figure 4.1.2.1 Landing Page

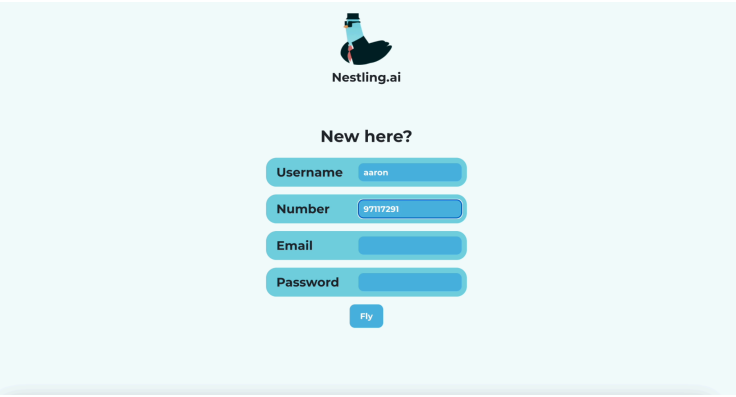


Figure 4.1.2.2 Sign Up page

Log In: User clicks on the 'login' button and enters their email and password. User is then brought to the homepage(Figure 4.2.1.3 to Figure 4.2.1.5)

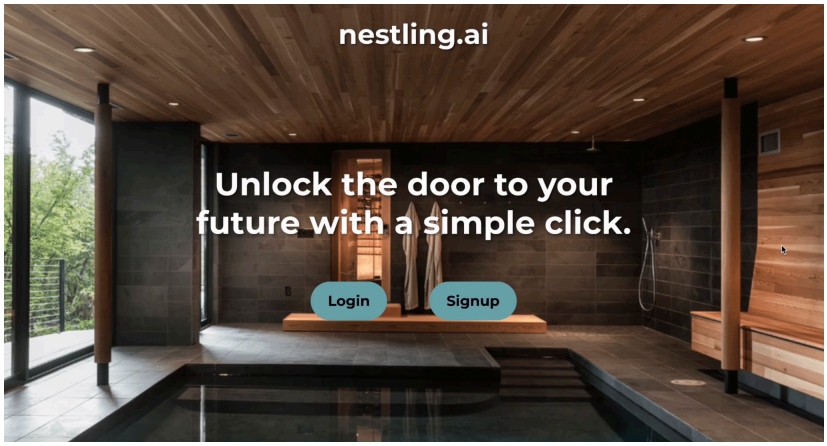


Figure 4.2.1.3 Landing Page

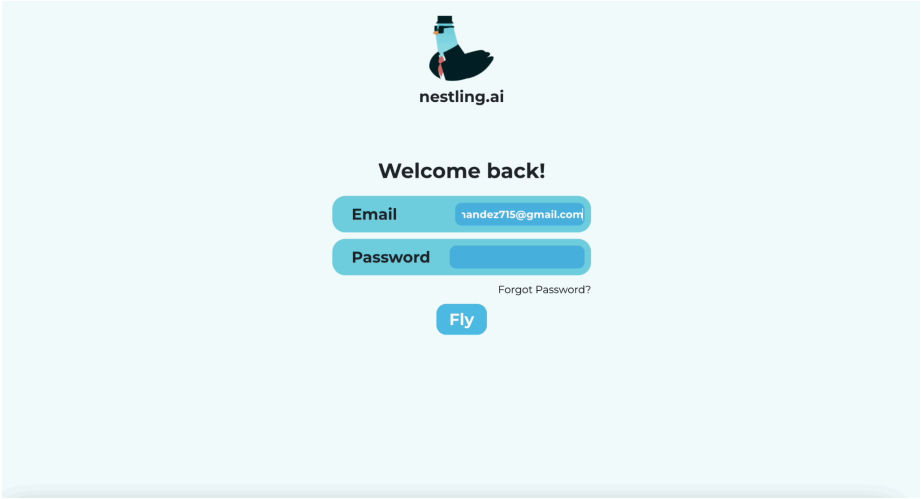


Figure: 4.2.1.4 Login Page

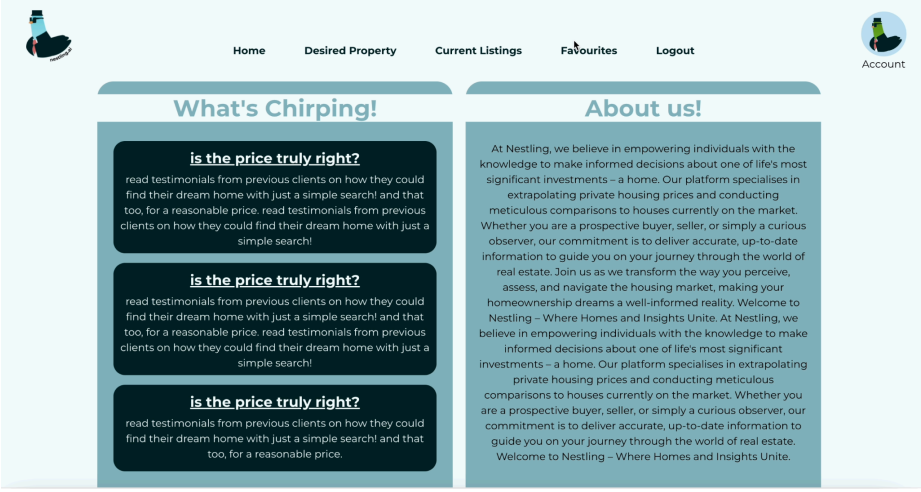


Figure 4.2.1.5 Homepage

Forgot Password: User clicks on the link ‘forgot password’ and enters their email for a reset link to be sent. User then navigates to their email and clicks on the reset link . User then fills in the relevant fields and inputs a new password. User is then brought to the homepage(Figure 4.2.1.6 to Figure 4.2.1.9)

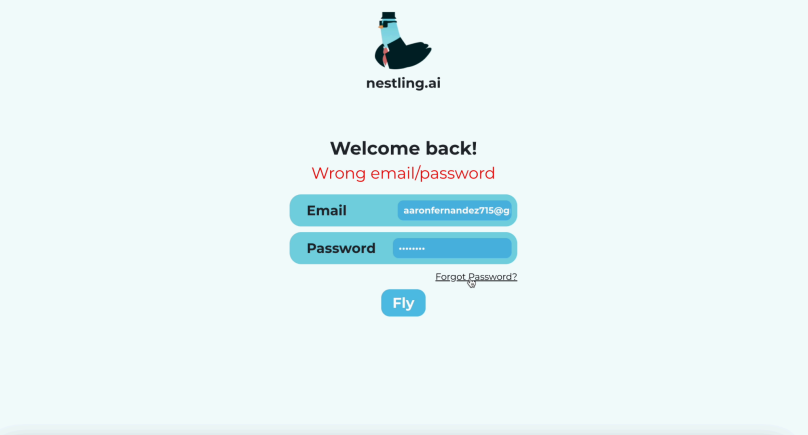


Figure 4.2.1.6 Wrong password prompt

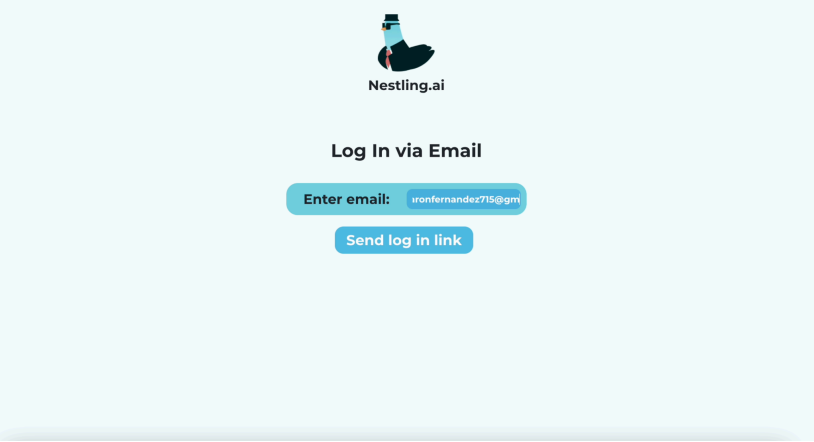


Figure 4.2.1.7 Log in via email page

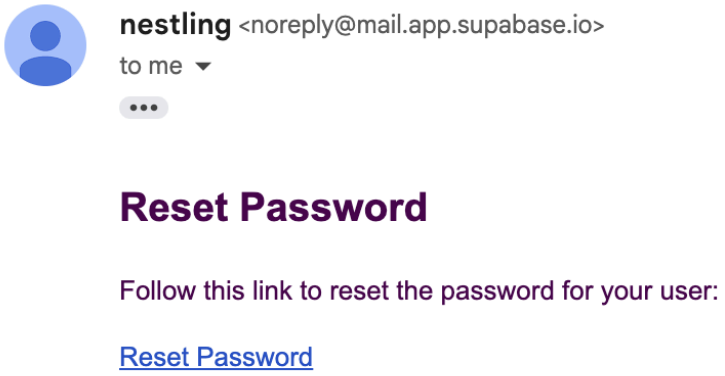


Figure 4.2.1.8 Reset password link via email

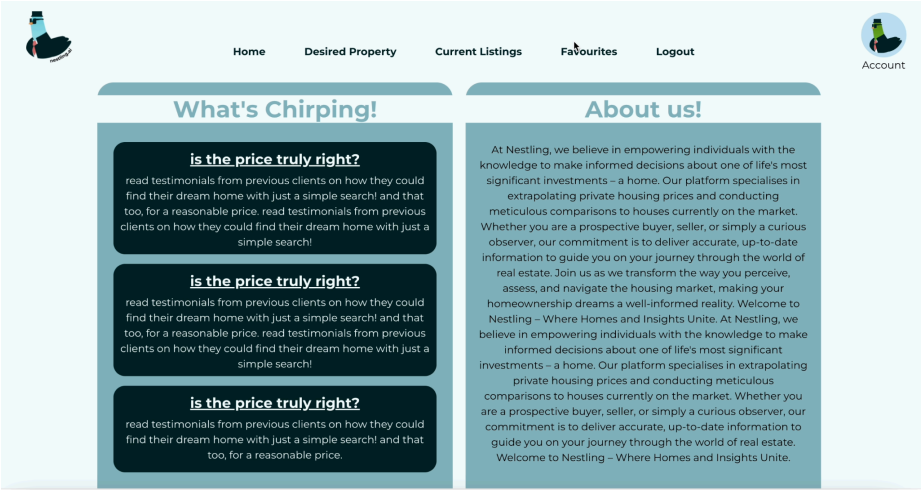


Figure 4.2.1.9: Homepage

Change Username/Number: User clicks on the ‘Account’ link on the top right of the homepage and they are then directed to the account details page. The user can then edit username and number and click on the ‘change username/number’. The user’s new information is then saved in the user account details database (Figure 4.2.1.9 & Figure 4.2.1.10)

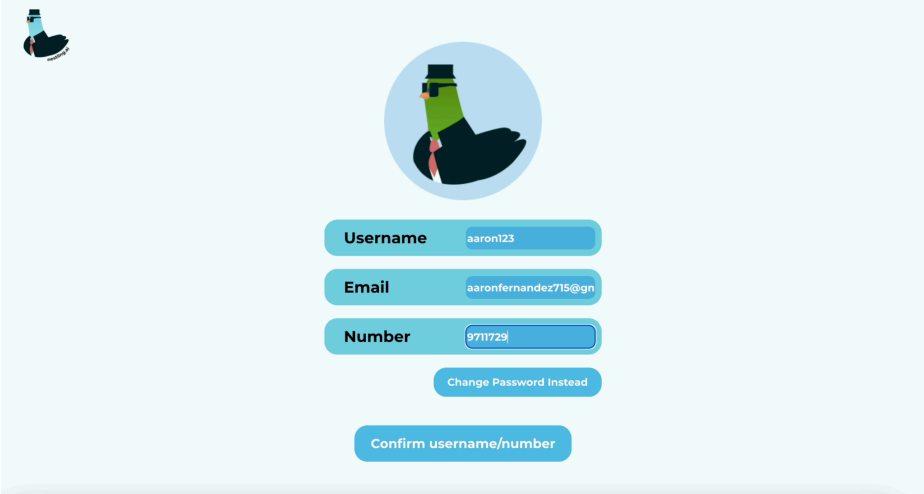


Figure 4.2.1.10: Account details page

4.1.3 Functional Requirements

REQ-1: The system shall provide a login page where users can enter their username and password.

- If the username or password is incorrect, the system shall display an error message.

- If the user enters invalid inputs (e.g., empty fields), the system shall prompt the user to correct them.

REQ-2: The system shall provide a sign-up page where new users can create an account by entering a username, email, and password.

- The system shall validate that the email is in a valid format.
- The system shall enforce password complexity rules (e.g., minimum length, required characters).

REQ-3: The system shall provide a password reset mechanism where users can request a password reset link via email.

- If the email address is not found in the system, the system shall display an error message.
- The system shall send a password reset email with a unique link to the user's email address.

REQ-4: The system shall provide an account details page where users can view and update their account information.

- Users shall be able to change their username and number on this page.
- The system shall save the new information in the user account details database

4.2 PricePoint AI (Desired Property Function)

4.2.1 Description and Priority

PricePoint AI has an AI prediction tool, that we have developed from scratch, used to predict prices of a user's desired properties based on a set of defined attributes. The defined attributes for PricePoint AI is the distance to nearest MRT stations, distance from nearest supermarkets, distance from nearest primary schools, distance from nearest secondary school, distance from nearest parks, distance from nearest malls and gross floor area of the desired property. We have allowed the user to select the distance from 3 amenities ,out of the 6 possible amenities (MRT stations, supermarkets, primary schools, secondary schools, parks and malls), which has a unique trained and tested model in PricePoint AI. As a result, we have generated 20 models and every of these models includes the gross floor area of the property as a variable. As a result every model generated will require 4 inputs in order to generate an output of predicted price. We have obtained all the data to test the models from various sources as mentioned in Section 1.5, References.

Before determining which models to select the models from, we decided to do some testing between 6 models which are multi-linear regression, Least Absolute Shrinkage and Selection Operator (Lasso) regularisation, Ridge regularisation, decision tree regression, neural networks and eXtreme Gradient Boosting (XGBoost) regularisation. Our primary factor in determining which model is the time taken for consistent and reliable testing. The factors that take up a lower priority in determining which models to use are coefficient of determination or R^2 value, mean squared error and mean absolute error. Neural network

took at least 3 hours to do 1 training of the model with 8 layers of 100 perceptrons each which resulted in an R^2 value of 0.42. It would take days to do an mean testing between the number of layer and number of perceptron within each layer, in order to even come up with a satisfactory R^2 value. XGBoost has at least 10 important parameters that can be varied to find the most optimum R^2. It would also take days of testing to find a optimum values for each parameter till we find a satisfactory R^2 value. Multi- linear regression is too elementary a model to use as we cannot vary overfitting or other aspects of the model to achieve a better R2 value. As a result, we decided to proceed with Lasso regularisation, Ridge regularisation and Decision Tree regression as we would be able to deliver an AI product with a high level of accuracy within the stipulated time frame of initial application deployment.

Lasso regularisation is a technique that adds a penalty to the ordinary least squares objective function in order to reduce the coefficient estimates towards zero and perform variable selection by forcing some coefficients to be exactly zero. The penalty incurred by Lasso regularisation is as follows:

$$\lambda \sum_{x=1}^{n} \left| c_x \right|$$

λ is the regularisation strength parameter or the alpha value and c is the weight of each attribute

The alpha value can be varied from 0 to infinity. Therefore we vary the alpha value until we find the highest R^2 value.

Ridge regularisation is very similar to Lasso regularisation however the only difference is that the weight of each attribute is squared. The penalty incurred by Ridge regularisation is as follows:

$$\lambda \sum_{x=1}^{n} \left| c_x^2 \right|$$

λ is the regularisation strength parameter or the alpha value and c is the weight of each attribute

The alpha value can be varied from 0 to infinity. Therefore we vary the alpha value until we find the highest R^2 value.

Decision Tree Regression is a technique that works by partitioning the feature space into a set of rectangular regions and predicting the average target value of the training samples within each region. For the purposes of our AI we have decided to vary the maximum depth the tree can grow to find the maximum depth that produces the highest R^2 value.

In order to generate a model for each of the 20 possible combinations of amenities, we run the required test data and train data through each of the models to determine which model produces the best outcome.

```
Highest R2 value for Lasso Regularisation:  0.680992025800983
Corresponding Alpha value for Lasso Regularisation:  763754.0834341669
MSE for highest R2:  6828917120008.927
MAE for highest R2:  1815607.0917365812

Highest R2 value for Ridge Regularisation:  0.6674201634004007
Corresponding Alpha value for Ridge Regularisation:  -23.0
MSE for highest R2:  6512071984510.858
MAE for highest R2:  1955141.6924583665

Highest average R2 value for Decision Tree: 0.9262013997646298
Corresponding Max Depth: 7
MSE for Highest Average R2: 1347453619666.3904
MAE for Highest Average R2: 702110.0111454279

Best model is Decision Tree Regression
```

Fig 4.2.1.1 Output of algorithm to determine best model for each combination

As seen from the image above, for the combination of distance to nearest MRT station, distance to nearest supermarket, distance to nearest secondary school and gross floor area of desired property, the best model is Decision Tree Regression.

After selecting and generating the model for each of the 20 possible combinations we download the models into pickle (.pkl) files which then can be used to generate an output in a Python script running in a secondary environment as mentioned in Section 2.4, Operating Environment. The output then can be displayed to the user in the application.

This feature is a high priority feature because it fulfils one of the main requirements of the application which is to generate a predicted price for the user.

4.2.2 Stimulus/Response Sequences

Once a user clicks on the desired property tab, they would be led to the page as seen in Fig 4.2.2.1.

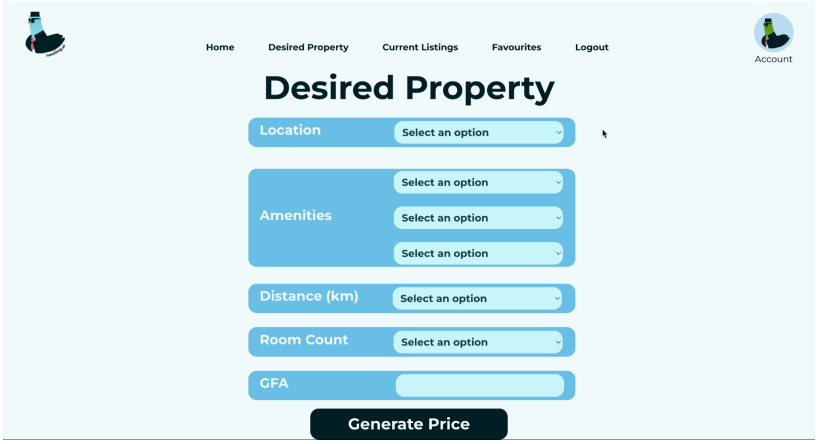


Fig 4.2.2.1 Desired Property Page without populated attributes

After the user populates the fields the page would look as seen in Fig 4.2.2.2.

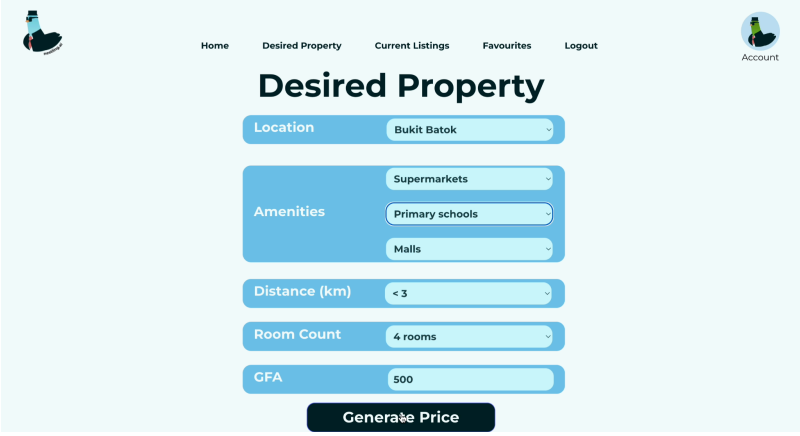


Fig 4.2.2.2 Desired Property Page with populated attributes

The user will then click on the Generate Price button to produce an estimated price as seen in Fig 4.2.2.3.

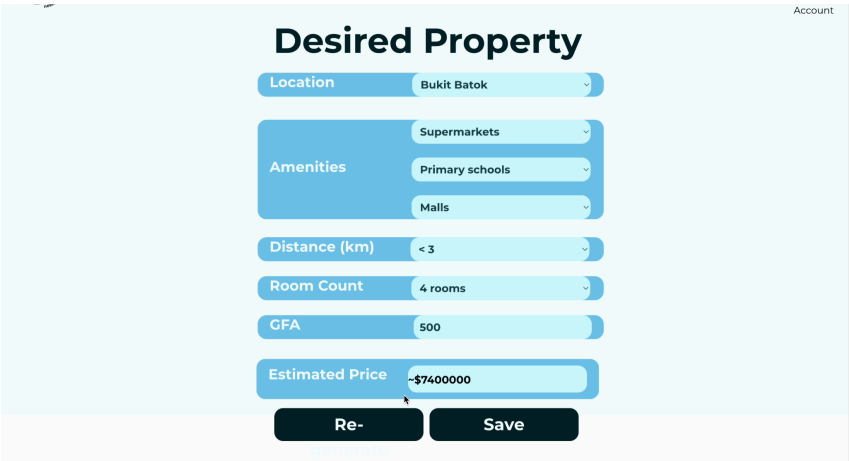


Fig 4.2.2.3 Desired Property Page with populated attributes and estimated price

The user can also regenerate new values after entering new attributes, especially Amenities, Distance and GFA, as seen in Fig 4.2.2.4. It also can be seen that the estimated price generated has changed after clicking on Re-generate Price.

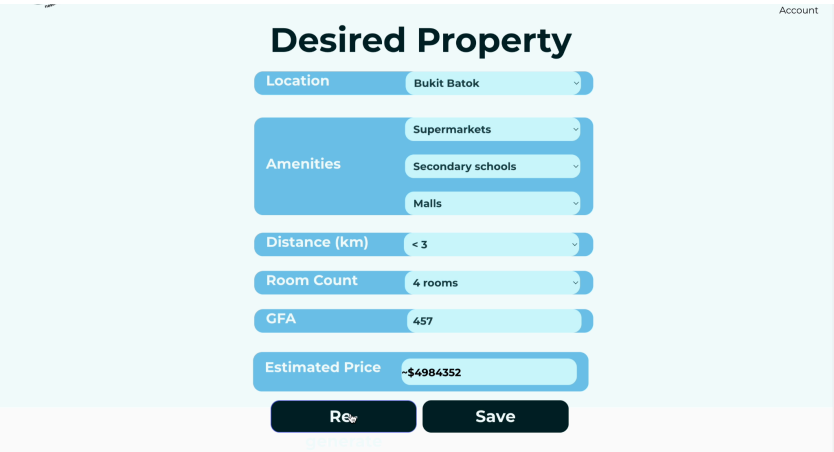


Fig 4.2.2.4 Desired Property Page with changed attributes and re-generated price

The user can also save the desired property attributes as seen in Fig 4.2.2.5.

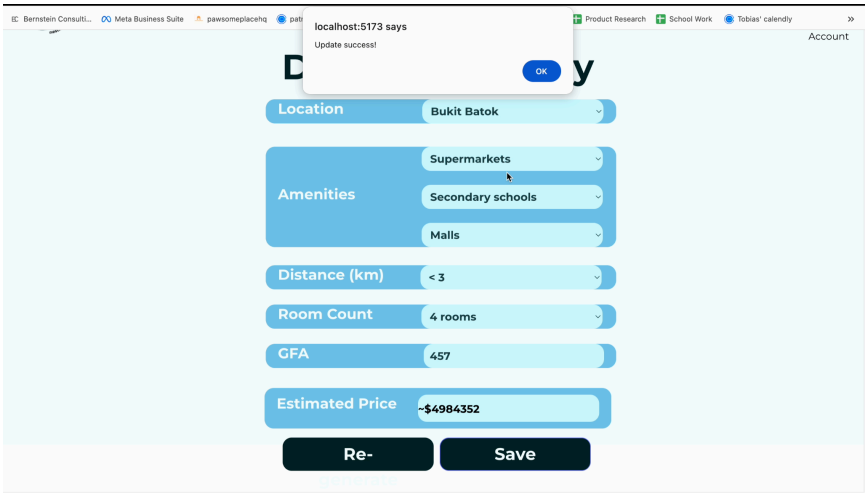


Fig 4.2.2.5 Desired Property Page with saved attributes

4.2.3 Functional Requirements

REQ-1: The users shall be able to select the desired property attributes.

- The User shall select the desired property location from the drop-down menu.
- The User shall select the desired number of rooms from the drop-down menu.
- The User shall select their desired amenities nearby, including the nearest MRT stations, shopping malls, primary schools, and secondary schools and their proximity range from the desired amenities.

REQ-2: The user display must consist of the desired property's estimated buying price: the estimated price will be generated using an AI model.

4.3 Property Recommendations

4.3.1 Description and Priority

In the current listings component, upon filling out the filter form and clicking the search button, the user's filter inputs will be routed into a percentage match logic control class in order to generate a percentage match number by comparing with each individual property in the database.

A list of properties would then be generated, sorted from the highest to the lowest percentage match, and then rendered on the screen for the user to pick from.

This feature is of high priority as it is critical in helping the user find out the most suitable property based on their circumstances and preferences, and subsequently how effective the user perceives the system to be after using it.

4.3.2 Stimulus/Response Sequences

User will first fill out attributes in the filter form on the left:

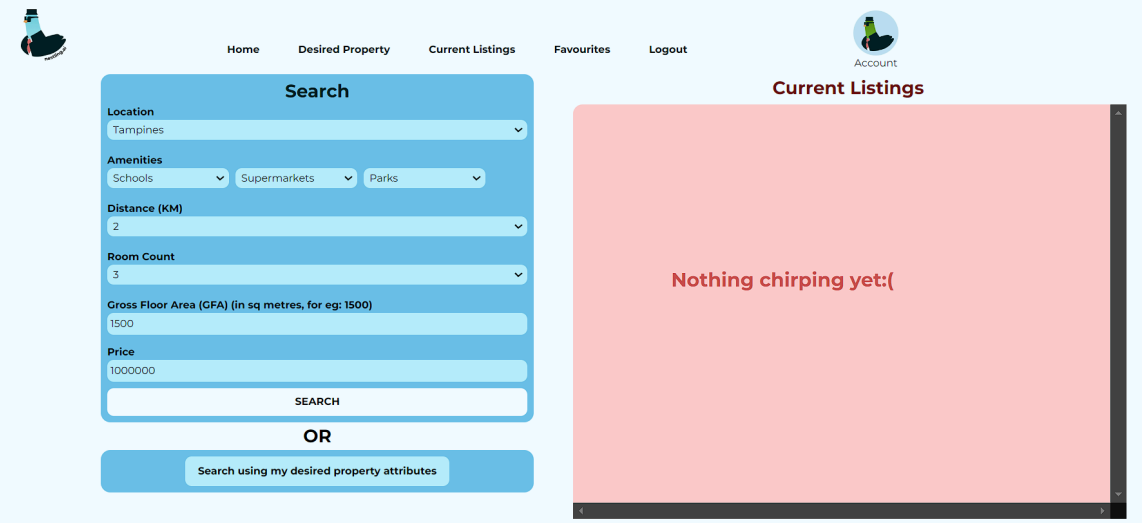


Figure 4.3.1 : Current Listing page before submitting filter form

Once the search button is clicked, a list of properties sorted based on descending order of percentage match will be rendered on the right:

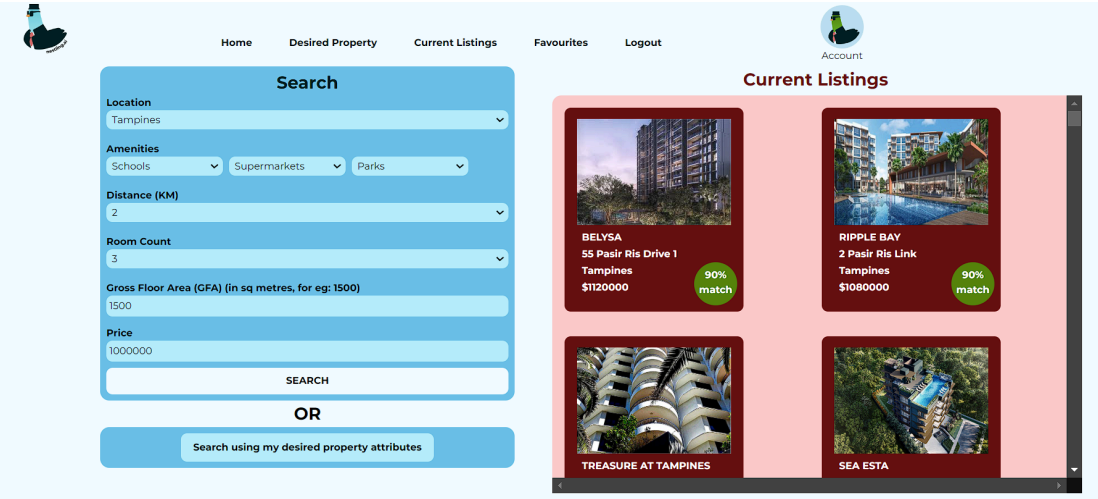


Figure 4.3.2 : Current Listing page after submitting filter form

Once a user clicks into an individual property listing, the percentage match indicator shown within a green circle on the top left would continue to display the percentage match compared to the attributes filled out by the user in the filter form on the previous page.

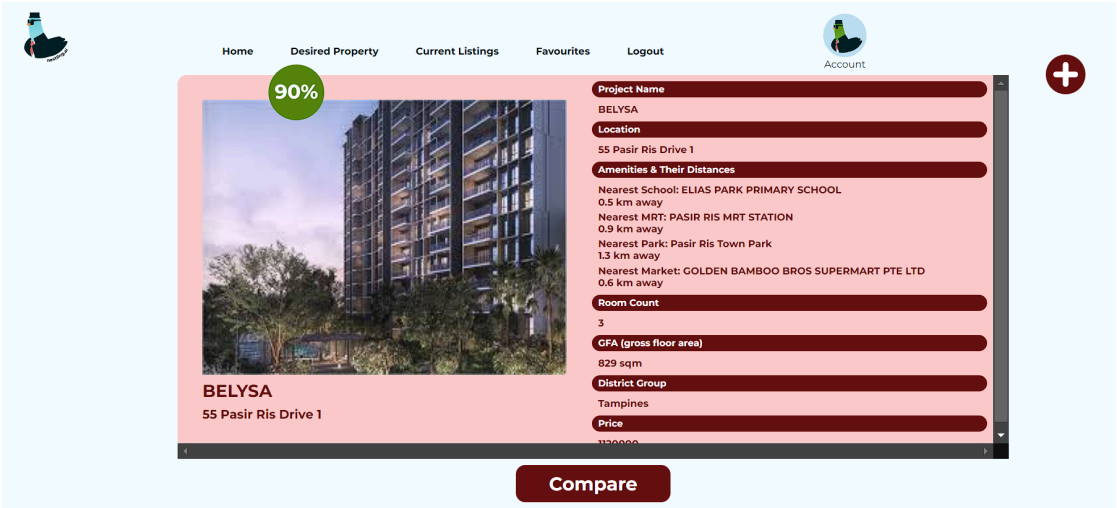


Figure 4.3.3 : Individual Property Listing

4.3.3 Functional Requirements

REQ-1: The system shall generate a numerical percentage match for every single property, once the filter form is completely filled out by the user and the search button is clicked on.

- Even if there is no match, the number shall be displayed as 0%
- Value should be between 0 to 100

REQ-2: The system shall generate a numerical percentage match for every single property, once the search using desired property attributes button is clicked on and the user has completely filled out and saved their desired property attributes.

REQ-3: The system shall retain the numerical percentage match and display it when the user clicks on any individual property listing.

4.4 CompareAI (AI Comparison Analysis of Properties)

4.4.1 Description and Priority

The CompareAI tool creates a structured AI output paragraph for two pages of our web application. One is comparing their searched properties to a specified property attributes page, while another is comparing two saved properties. The AI is designed to respond similarly to a housing agent, stating the main attribute differences as well as the best option, taking into account common criteria such as family friendliness, accessibility, and lifestyle facilities.

For the AI, we utilised the LangChain toolkit in conjunction with OpenAI to perform comparison analysis. Furthermore, our backend architecture is built on Flask, a Python-based web framework that handles API requests for content uploads to our servers and data retrieval for display on the frontend user interface.

This feature is regarded as a high priority as it is relevant to many housing buyers. Employing a housing agent can be costly to some extent, but with Nestling AI's comparison analysis feature, buyers can receive instant and specific suggestions for any property they have chosen from the pool of current listings at any time.

4.4.2 Stimulus/Response Sequences

After obtaining property recommendations, the user will select a property and then click the compare button at the bottom of the page. This button would generate an AI output for comparison analysis.

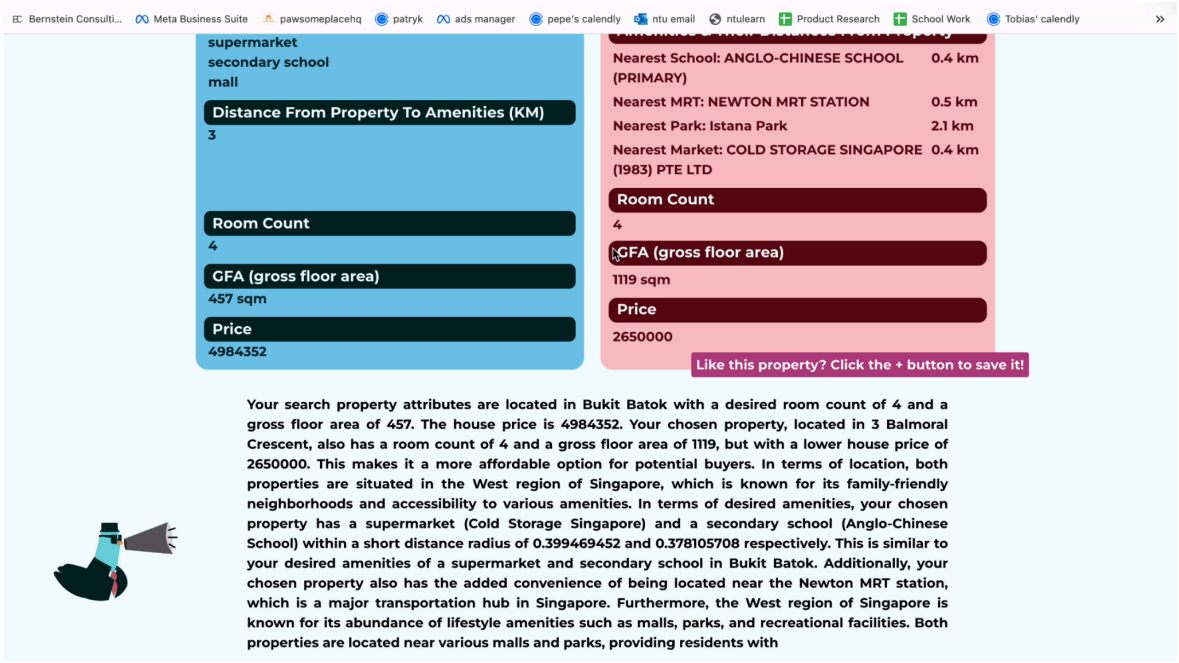


Figure 4.4.1: AI output on compare page for search attributes vs chosen listing attributes

Similar to the favourites page, customers can compare up to two properties from their saved favourites. The user can then click on the generate button at the bottom to generate the AI comparison analysis.

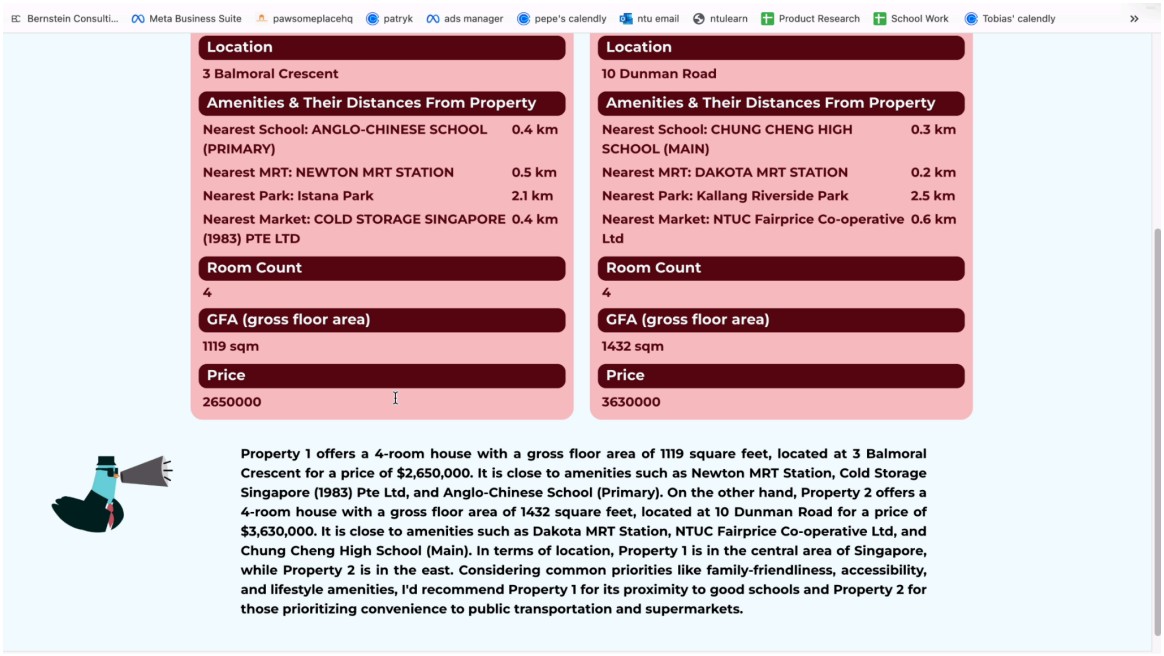


Figure 4.4.2: AI output on compare page for 2 saved favourites property listing

4.4.3 Functional Requirements

REQ-1: The system should output an AI comparison analysis paragraph comparing searched attributes and 1 property, once the user has filled the search page and chosen one property from the pool of recommended properties.

- The System will only generate the paragraph when the user clicks on the compare button.

REQ-2: The system should output an AI comparison analysis paragraph comparing only 2 properties, once the user has selected 2 properties from their saved favourites page.

- The System will only generate the paragraph when the user clicks on the generate button.

4.5 Saving Favourite Properties

4.5.1 Description and Priority

The Saving Favourite Properties feature allows users to create a repository of properties that match their needs for later perusal. Upon gathering a repository of properties, users are able to compare two properties to see the difference between them and evaluate which one is objectively better.

This is a high priority feature and is what sets us apart from current property-finder websites out there. Many new homefinders have access to knowing about a plethora of properties but lack the real estate knowledge to evaluate them and make an informed decision when buying their dream house.

4.5.2 Stimulus/Response Sequences

When the user chances upon a property listing that they like while browsing through properties in Current Listing, they are able to save the property by clicking the ‘plus’ button on the top left hand corner of the property.

Figure 4.5.1: Screen Capture of the ‘plus’ button

The properties saved would show up in the Favourites section, where the users can still view the property details.

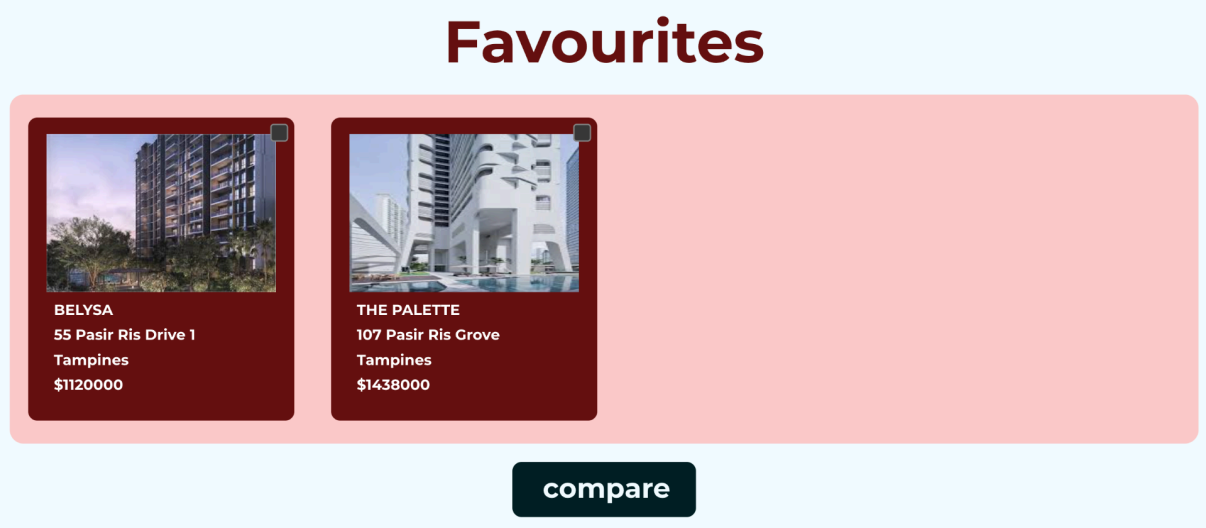


Figure 4.5.2: Screen Capture of the Favourites page

After selecting two properties, the user is able to view a side-by-side comparison of the properties, with their attributes listed out, and if interested, is able to hear from our AI Real Estate Agent.

Figure 4.5.3: Comparison of Properties

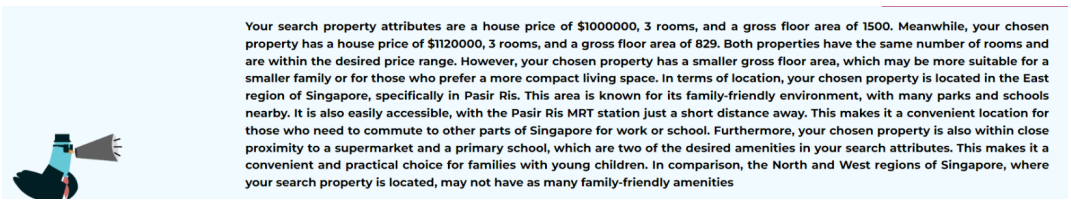


Figure 4.5.4: AI Output Generated

4.5.3 Functional Requirements

REQ-1: When the save button is pressed, the web application shall save the property to the favourites file.

REQ-2: The favourites file will consist of all the saved current properties, and when the User visits the favourites files, the user can select his best-suited property.

- Where differences and similarities between current properties are concerned, the system shall present a clear comparison, aiding users in making informed decisions.
- The User data display must consist of the information generated by an Artificial Intelligence (AI) model.
- The AI model will output a comparison between the two properties.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Performance Requirement	Details	Rational
Landing page	The nestling application shall display the landing page within 2 to 3 seconds of receiving the request under standard conditions. (average internet speed of 25 Mbps and server load handling up to 1000 simultaneous users).	This requirement provides users with rapid access to information, thereby keeping their interest and competitive edge in the fast-paced real estate industry.
Current Listing Properties page + percentage match	The Current Listing page should complete loads for listing details pages should not exceed 3 seconds under the same conditions.	By defining such standards, user experience promotes longer engagement times and raises the possibility of repeat visits.
Artificial Intelligence model loading	For AI based features "Generate Estimated Price" and "Comparison Analysis" should provide results within 5 seconds for 95% of the requests.	There is no doubt that AI loading time takes some time; nonetheless, it should be kept as short as possible, as quick data processing is critical for customer pleasure, particularly when making investment decisions based on AI insights.
Artificial Intelligence accuracy	For AI accuracy, the AI models should be able to anticipate house values with at least 90% accuracy by verifying against past market data in Singapore.	Implementing a high degree of precision promotes user confidence in the estimations given by the program by assuring its dependability and trustworthiness.
Real-Time Data	For Real-Time Data integration, the application should integrate government datasets to update current listing and prices at least once every 2 times a week.	Possessing latest data is essential for accurate forecasting and applicability to current market conditions.

System Availability	Uptime: The application should have an uptime of 99.9% monthly. Backup and maintenance: The system must have a maximum downtime of 0.1% per month for scheduled maintenance.	In a setting as important to business continuity as real estate, high availability is crucial to preserving user trust.

5.2 Safety Requirements

1. For data loss prevention, implement a complete data backup solution, including daily backups of all nestling users' data and transaction logs.
2. For cybersecurity measures, the system needs to deploy advanced cybersecurity frameworks including firewalls, intrusion detection systems (IDS), and intrusion prevention systems (IPS). Moreover, regular updates and patches to all security software and systems to mitigate new security vulnerabilities.

5.3 Security Requirements

1. The system's database security must meet Personal Data Protection Act (PDPA) requirements. To ensure that users credential authentication information will be securely protected.
2. The system should only collect only the data necessary for fulfilling its intended function and nothing more. Nestling collects gmail, password and phone number only.
3. The system must implement OAuth 2.0 for user authentication with multi-factor authentication (MFA) options available for all users.
4. The system must have automated session timeouts enforced after 15 minutes of inactivity, and session token should be used for managing sessions securely.

5.4 Software Quality Attributes

5.4.1 Object-Oriented Design Principles

1. Abstraction

This refers to the process of hiding the complexity of the code, while only showing those implementations which are important. Designed two major components which were utilised throughout our application - the Navigation Bar and Listing Panel components. Making these two components allowed us not to care about the functionalities of them, and rather just ensure that they were displayed

on the screen. We further customised the Listing Panel components to display slightly different versions of the components dependent on the file.

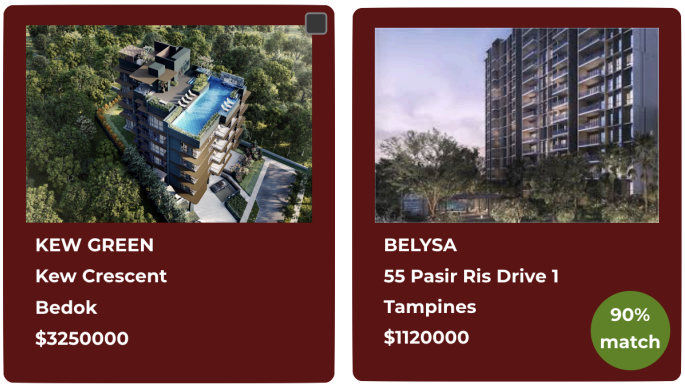


Figure 5.4.1.1: Two different implementations of the Listing Panel components

The right side panel depicts the implementation of the Listing Panel in the Favourites screen, without the percentage match bubble and with a checkbox instead, while the left side panel depicts the implementations of the Listing Panel in the Current Listings screen, with the percentage match bubble.

2. *Single Responsibility Principle (SRP)*

Each package should have a class dedicated to a specific task, avoiding unrelated responsibilities. All the files within the Nestling applications have unique tasks, such as the Desired Property and the Favourites function.

3. *Open-Closed Principle (OCP)*

This involves creating unmodifiable but extensible classes, where new classes can be added to enhance a class's functionality. This can be seen in the Account Page, where the base functionality of storing the user’s details remains, and is enhanced by the addition of more user fields. Another example is the Current Listings page, where the search results are based on the dropdown options given. Therefore, the search results can be further stratified with more dropdown options which can be easily added.

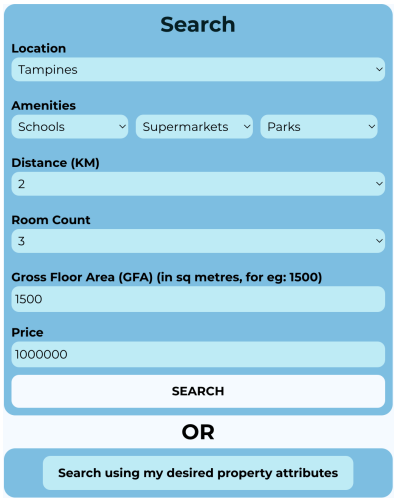


Figure 5.4.1.2: Search function in Current Listing

5.4.2 Flexibility

With big text and bright colours, this application can be viewed in any screen size and is scalable according to the screen size it is viewed on. Due to the modular layout, it is easily maximizable and minimizable, and new features can be added on as long as proper routing is done to the Navigation Bar, as shown below. The addition of new functions are tied by routing through the Navigation Bar and therefore, new functions would not affect older functions.

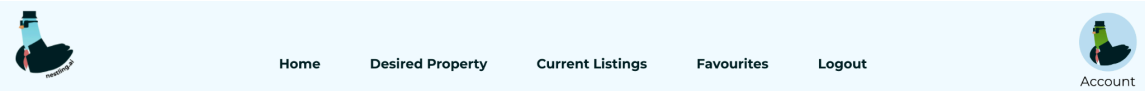


Figure 5.4.3.1: Navigation Bar Function

5.4.3 Agile Framework

According to the lecture content, the agile development framework entails the interleavement of the program specification, design and implementation. The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation. Using the agile method of incremental development, we had frequent revisions of our application, focussing on first pushing out the front-end capabilities of the code, then linking it to our database, then focussing on any additional back-end capabilities of the code, such as the AI.

5.4.4 API and Design-First approach (Visual Paradigm, n.d.)

An API-first approach prioritises the development of APIs at the forefront of any project. This strategy helps prevent neglecting essential functions and wasting time and resources on features that may not be needed. This method is similar to customer-centric, lean product design, in which designers first create concepts and then gather feedback from users before laying out detailed technical specifications. Using Figma, we were able to determine the layout of our application before we started assigning front-end and back-end functionalities to it. We utilised the design feature to plan out our application interface as shown

below, as well as the prototype feature to mimic the flow of our application. This helped us iron out our user's needs and application features in a visual format that was easy to track and see tangible changes.

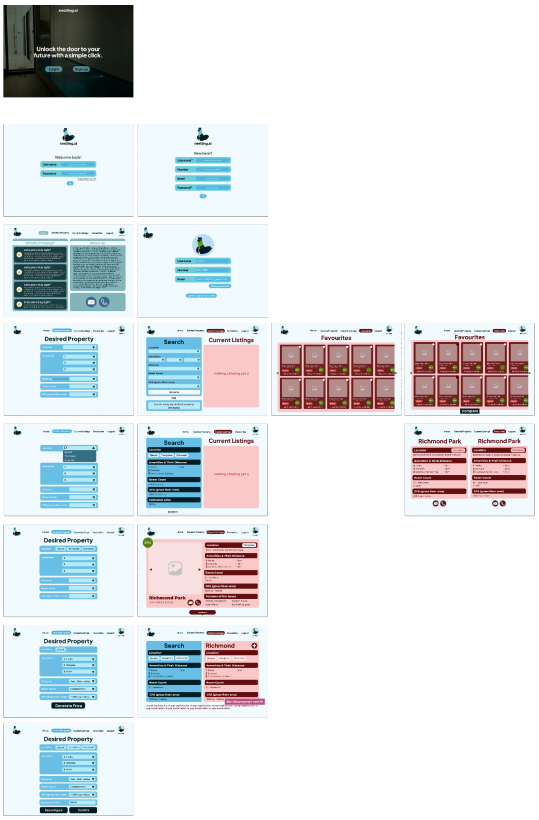


Figure 5.4.4.1: Figma Layout of our application

5.5 Business Rules

5.5.1 Account Creation (Sign Up)

Users with a valid email and phone number can create an account for the application. We need to implement a system that verifies that a valid account has been created.

5.5.2 User Authentication (Login)

Users must input the correct username and password to use the application. We need to have a backend database that can verify and authenticate a correct username and password has been inputted in order to allow the user to proceed with using the application.

5.5.3 Update User Information

Users must be able to update account information such as username and phone number. We need to ensure only authenticated users can update their information.

5.5.4 Forgot Password

Users must be able to reset their password by inputting the correct email which would allow them to reset their password. There must be a backend feature that sends a reset link to their email account, the users to reset their account. A valid email must have been provided by the user prior to the usage of this feature.

5.5.5 Save Desired Property Attributes

Users must be able to save desired property attributes after populating the fields in the desired property tab. A backend database is required to save this information and only after the user has undergone the authentication process prior in 5.5.2.

5.5.6 Save Current Listings

Users must be able to have a current property listing so that they are able to view it in 5.5.7. A backend database is required to save this information and only after the user has undergone the authentication process prior in 5.5.2.

5.5.7 View Saved Listings

Users must be able to view saved listings. The user must have accomplished 5.5.6 before being able to access this feature.

6. Other Requirements

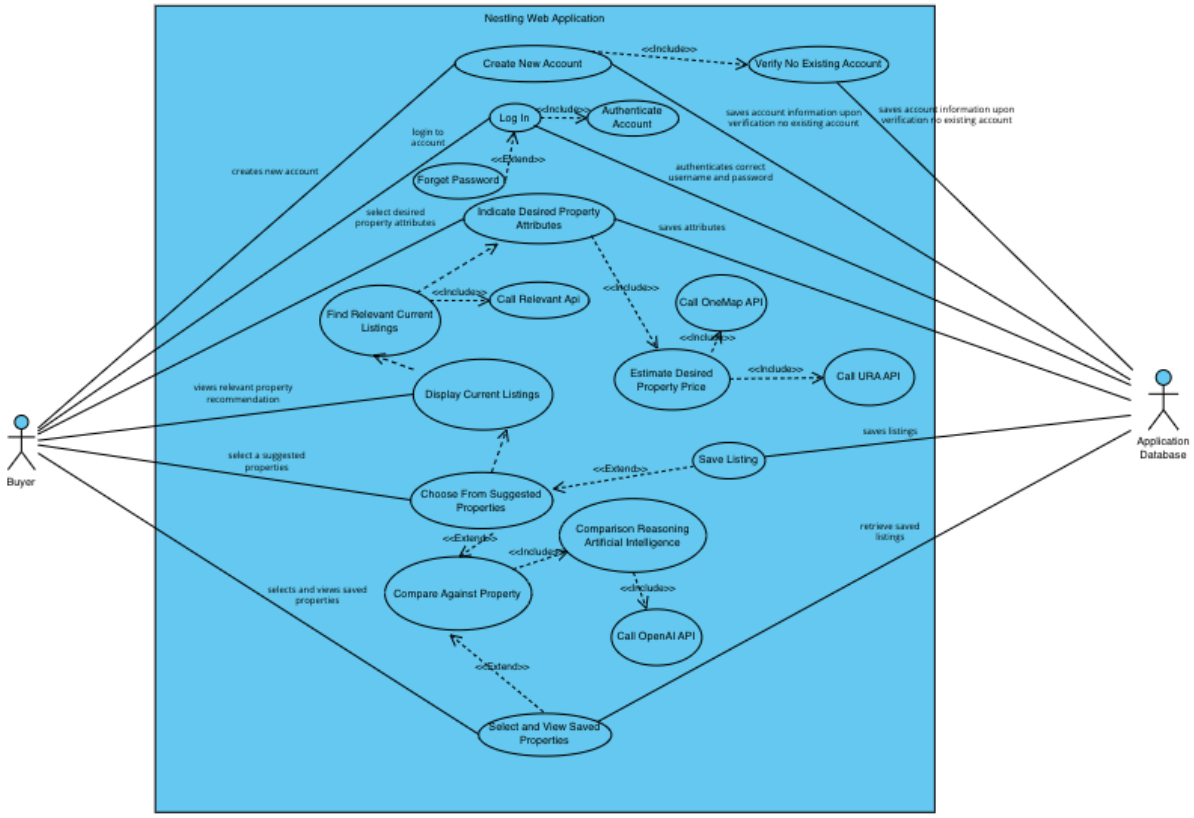
Appendix A: Glossary

Term	Definition
Property	Represents a private accommodation in Singapore. A property is either a condominium or a landed property.
Location	A place in Singapore in which a property is located at. It includes information on its town, block, level, as well as postal code.
Amenities	Facilities to service basic human needs, such as hunger, entertainment, etc. Examples of amenities include school, shopping malls, coffee shops.
Price	Total and final amount of money needed to pay for the house. Includes miscellaneous and administrative fees.
Closeness	Refers to straight line heuristic distance from a particular point of interest, such as amenities. Distance is to be measured in kilometres by default.
Transport	Means of public transportation for humans. Transport should either be a bus stop, or MRT station.

Appendix B: Analysis Models

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

1. Use Case Diagram



2. Use Case Description

Use Case ID:	1		
Use Case Name:	Create New Account		
Created By:	Huai Zhi, Aaron	**Last Updated By:**	Huai Zhi, Aaron
Date Created:	26/01/2024	**Date Last Updated:**	01/04/2024

Actor:	User, Application Database
Description:	Registration of a new user account
Preconditions:	• There must be no previously registered accounts tied to the username and email • Connection to mobile data/WiFi
Postconditions:	• Users are able to login to access nestling services
Priority:	High
Frequency of Use:	Once per user
Flow of Events:	1. User enters details into the respective fields 2. User the password into the 'Confirm Password' field 3. User clicks on 'Fly' button 4. System validates if username and email does not exist in database 5. Save username, email, phone number and password in database 6. User account is created
Alternative Flows:	NA
Exceptions:	NA
Includes:	Verify No Existing Accounts
Special Requirements:	NA

Assumptions:	• Internet connection is available
Notes and Issues:	NA

Use Case ID:	2		
Use Case Name:	Verify No Existing Accounts		
Created By:	Huai Zhi, Aaron	**Last Updated By:**	Huai Zhi, Aaron
Date Created:	26/01/2024	**Date Last Updated:**	01/04/2024

Actor:	Application Database
Description:	Verifies no existing account in application database
Preconditions:	• User must populated fields in Sign Up page • Connection to mobile data/WiFi
Postconditions:	• Users are able to login to access nestling services
Priority:	High
Frequency of Use:	Once per user
Flow of Events:	1. Application database verifies that the user has no existing account
Alternative Flows:	AF1: Username or/and Email Address already exists in the user database • If the Username or/and Email Address matches an existing account, prompt the user that the account already exists. Return to step 1.
Exceptions:	NA
Includes:	NA
Special Requirements:	• Email is valid • Password needs to be at least 6 characters
Assumptions:	• Internet connection is available • Each user should only have 1 account

Notes and Issues:	NA

Use Case ID:	3		
Use Case Name:	Log in		
Created By:	Huai Zhi, Aaron	**Last Updated By:**	Huai Zhi, Aaron
Date Created:	26/01/2024	**Date Last Updated:**	01/04/2024

Actor:	User, Application Database
Description:	User logs in to the application
Preconditions:	• There must be a registered account in the database • Connection to mobile data/WiFi
Postconditions:	• Users are able to access Nestling services
Priority:	High
Frequency of Use:	Multiple per user (High)
Flow of Events:	1. User enters username and password 2. User clicks on “Fly” button 3. The application database authenticates username and password before opening the landing page by passing control of operation to the Authenticate Account use case.
Alternative Flows:	AF1: Incorrect username and/or password entered • The login page indicates a wrong username and/or password has been inputted.
Exceptions:	NA
Includes:	Authenticate Account
Special Requirements:	• Correct username and password has been inputted • Account exists
Assumptions:	• Internet connection is available
Notes and Issues:	NA

Use Case ID:	4		
Use Case Name:	Authenticate Account		
Created By:	Huai Zhi, Aaron	**Last Updated By:**	Huai Zhi, Aaron
Date Created:	26/01/2024	**Date Last Updated:**	01/04/2024

Actor:	Application Database
Description:	Verifies username and password is correct
Preconditions:	• User must populated fields in Login page • Connection to mobile data/WiFi
Postconditions:	• Users are able to login to access nestling services
Priority:	High
Frequency of Use:	Multiple per user (High)
Flow of Events:	1. Application database verifies that the username and password is correct
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	• Internet connection is available
Notes and Issues:	NA

Use Case ID:	5		
Use Case Name:	Forget Password		
Created By:	Huai Zhi, Aaron	**Last Updated By:**	Huai Zhi, Aaron
Date Created:	26/01/2024	**Date Last Updated:**	01/04/2024

Actor:	User, Application Database
Description:	User clicks on forgot password if they have indeed forgotten their password
Preconditions:	• There must be a registered account in the database • Connection to mobile data/WiFi
Postconditions:	• Users are able to reset password and to access Nestling services
Priority:	High
Frequency of Use:	Multiple per user (Low)
Flow of Events:	1. User clicks on the “Forget Password” button. 2. The application database sends a reset password link to the user’s email address. 3. User resets password. 4. User gains access to their account and is on the home page.
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Special Requirements:	• Valid email address • Account corresponding to email exists
Assumptions:	• Internet connection is available
Notes and Issues:	NA

Use Case ID:	6		
Use Case Name:	Indicate Desired Property Attributes		
Created By:	Abhijeet, Pranavi	**Last Updated By:**	Abhijeet, Pranavi
Date Created:	26/01/2024	**Date Last Updated:**	03/04/2024

Actor:	User, Application Database
Description:	User populates the Desired Property page
Preconditions:	• User should have login • Connection to mobile data/WiFi
Postconditions:	• Users are able to see the fields on the Desired Property page has been populated
Priority:	High
Frequency of Use:	Multiple per user (Medium)
Flow of Events:	1. User populates the fields in the Desired Property page. 2. User should be able to see the fields in the Desired Property page 3. User can save the desired property attributes
Alternative Flows:	AF1: The user inputs the same amenity twice • The application prompts the user to reinput amenities
Exceptions:	NA
Includes:	Estimate Desired Property Price
Special Requirements:	NA
Assumptions:	• Internet connection is available
Notes and Issues:	NA

Use Case ID:	7		
Use Case Name:	Estimate Desired Property Attributes		
Created By:	Abhijeet, Pranavi	**Last Updated By:**	Abhijeet, Pranavi
Date Created:	26/01/2024	**Date Last Updated:**	03/04/2024

Actor:	User, Application Database
Description:	User clicks on "Generate Price' button and the estimated price is displayed on the Desired Property page
Preconditions:	• User should have populated the fields in the Desired Property page
Postconditions:	• Users should be able to see the estimated price displayed on the Desired Property page
Priority:	High
Frequency of Use:	Multiple per user (Medium)
Flow of Events:	1. User clicks on the "Generate Price" button 2. Users should be able to see the estimated price displayed on the Desired Property page.
Alternative Flows:	NA
Exceptions:	NA
Includes:	Call OneMap API, Call URA API
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	8		
Use Case Name:	Call URA API		
Created By:	Abhijeet, Pranavi	**Last Updated By:**	Abhijeet, Pranavi
Date Created:	26/01/2024	**Date Last Updated:**	03/04/2024

Actor:	Application
Description:	• Application calls URA API and obtains data
Preconditions:	NA
Postconditions:	• Application receives data from URA API
Priority:	High
Frequency of Use:	Multiple per user (Low)
Flow of Events:	Application makes API call when required and obtains data
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	9		
Use Case Name:	Call OneMap API		
Created By:	Abhijeet, Pranavi	**Last Updated By:**	Abhijeet, Pranavi
Date Created:	26/01/2024	**Date Last Updated:**	05/04/2024

Actor:	Application
Description:	• Application calls OneMap API and obtains data
Preconditions:	NA
Postconditions:	• Application receives data from OneMap API
Priority:	High
Frequency of Use:	Multiple per user (Low)
Flow of Events:	Application makes API call when required and obtains data
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	10		
Use Case Name:	Display Current Listings		
Created By:	Pranavi, Sharmilla, Huai Zhi	**Last Updated By:**	Pranavi, Sharmilla, Huai Zhi
Date Created:	26/01/2024	**Date Last Updated:**	05/04/2024

Actor:	User, Application database
Description:	User populates search fields to search for current listings
Preconditions:	• User has logged in
Postconditions:	• User is able to see current listings being displayed
Priority:	High
Frequency of Use:	Multiple per user (Medium)
Flow of Events:	1. User populates fields manually or clicks on “Search using the desired property attributes. 2. Application conducts a search for current listings that best matches the fields or desired property attributes. 3. User sees the current listings being displayed on the Current Listings page.
Alternative Flows:	AF1: The user inputs the same amenity twice • The application prompts the user to reinput amenities
Exceptions:	NA
Includes:	Find Relevant Current Listings
Special Requirements:	• Desired property attributes has been saved
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	11		
Use Case Name:	Find Relevant Current Listings		
Created By:	Sharmilla, Huai Zhi	**Last Updated By:**	Sharmilla, Huai Zhi
Date Created:	26/01/2024	**Date Last Updated:**	05/04/2024

Actor:	Application Application database
Description:	Application conducts search using current listing search fields
Preconditions:	• User has logged in
Postconditions:	• Application obtains appropriate current listings
Priority:	High
Frequency of Use:	Multiple per user (Medium)
Flow of Events:	1. Application conducts a search for current listings that best matches the fields or desired property attributes.
Alternative Flows:	NA
Exceptions:	NA
Includes:	Call Relevant API
Special Requirements:	• Current Listing Search Fields has been populated
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	12		
Use Case Name:	Call Relevant API		
Created By:	Sharmilla	**Last Updated By:**	Sharmilla
Date Created:	26/01/2024	**Date Last Updated:**	05/04/2024

Actor:	Application
Description:	Application calls Relevant API and obtains data
Preconditions:	NA
Postconditions:	• Application receives data from Relevant API
Priority:	High
Frequency of Use:	Multiple per user (Low)
Flow of Events:	Application makes API call when required and obtains data
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	13		
Use Case Name:	Choose From Suggested Properties		
Created By:	Sharmilla	**Last Updated By:**	Sharmilla
Date Created:	26/01/2024	**Date Last Updated:**	05/04/2024

Actor:	User, Application
Description:	User is able to select from suggested properties displayed on Current Listings page
Preconditions:	• The user has undergone the Display Current Listings use case
Postconditions:	• User is able to view details about the property
Priority:	High
Frequency of Use:	Multiple per user (Medium)
Flow of Events:	1. User selects property from current listings being displayed 2. User is able to see information on the selected listing
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	14		
Use Case Name:	Save Listing		
Created By:	Sharmilla, Huai Zhi	**Last Updated By:**	Sharmilla, Huai Zhi
Date Created:	26/01/2024	**Date Last Updated:**	05/04/2024

Actor:	User, Application Database
Description:	User is able to save selected property from Choose From Suggested Properties use case
Preconditions:	• The user has undergone the Choose From Suggested Properties use case
Postconditions:	• User has saved selected property into application database
Priority:	High
Frequency of Use:	Multiple per user (Medium)
Flow of Events:	1. User clicks on the "+" button to save listing to application database
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	15		
Use Case Name:	Select and View Saved Properties		
Created By:	Nicole, Aaron	**Last Updated By:**	Nicole, Aaron
Date Created:	26/01/2024	**Date Last Updated:**	04/04/2024

Actor:	User, Application Database
Description:	User is able to save selected property from properties that has been saved
Preconditions:	• The user has saved listings into application database
Postconditions:	• User is able to view saved property
Priority:	High
Frequency of Use:	Multiple per user (Medium)
Flow of Events:	1. User clicks on property that has been saved 2. The information on the saved property is displayed
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	16		
Use Case Name:	Compare property		
Created By:	Nicole	**Last Updated By:**	Nicole
Date Created:	26/01/2024	**Date Last Updated:**	04/04/2024

Actor:	User, Application Database
Description:	User is able to compare 2 properties that has been saved
Preconditions:	• The user has selected 2 listings
Postconditions:	• User is able to compare 2 listings
Priority:	High
Frequency of Use:	Multiple per user (Medium)
Flow of Events:	1. User clicks on 2 listings to compare 2. User clicks on “Compare” button 3. The listings are displayed side-by-side
Alternative Flows:	AF1: The user does not select appropriate number of properties • The application prompts the user to select appropriate number of properties
Exceptions:	NA
Includes:	Comparison Reasoning Artificial Intelligence
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	17		
Use Case Name:	Comparison Reasoning Artificial Intelligence		
Created By:	Nicole	**Last Updated By:**	Nicole
Date Created:	26/01/2024	**Date Last Updated:**	04/04/2024

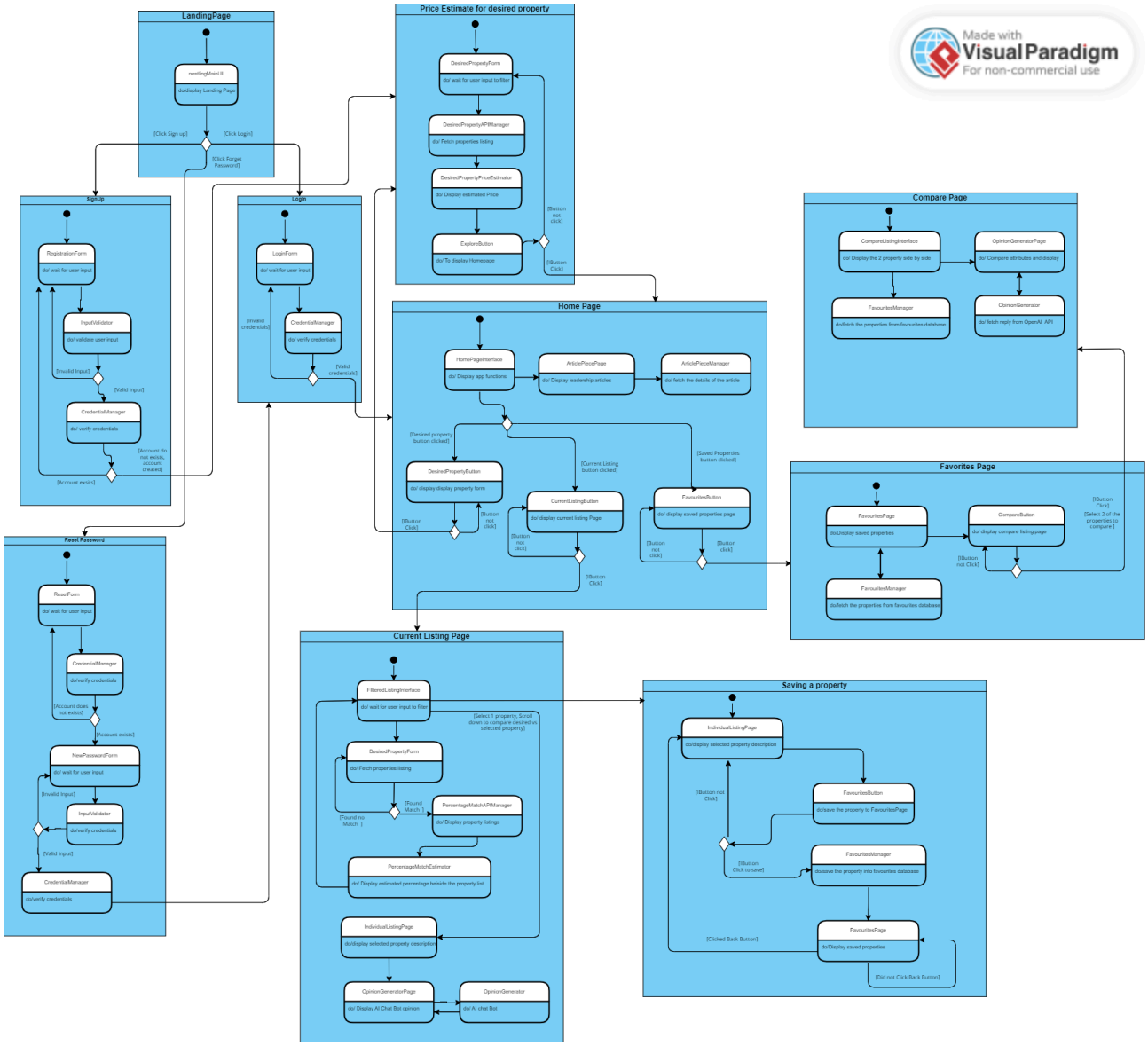
Actor:	User
Description:	User is able to generate an AI evaluation on comparison
Preconditions:	• The user has selected 2 listings in Compare Against Property use case
Postconditions:	• An AI evaluation is displayed on the Compare Listings page
Priority:	High
Frequency of Use:	Multiple per user (Medium)
Flow of Events:	1. User clicks on “Compare” button 2. Application sends query to OpenAi via Call OpenAI API use case 3. The application displays response on Compare Listings page
Alternative Flows:	NA
Exceptions:	NA
Includes:	Call OpenAI API
Special Requirements:	NA
Assumptions:	NA
Notes and Issues:	NA

Use Case ID:	18		
Use Case Name:	Call OpenAI API		
Created By:	Nicole	**Last Updated By:**	Nicole
Date Created:	26/01/2024	**Date Last Updated:**	04/04/2024

Actor:	Application
Description:	Application sends query to OpenAI API and receives a response
Preconditions:	NA
Postconditions:	• Application receives response from OpenAI
Priority:	High
Frequency of Use:	Multiple per user (Medium)
Flow of Events:	Application makes API call when required and obtains data
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Special Requirements:	• Internet Connection
Assumptions:	NA
Notes and Issues:	NA

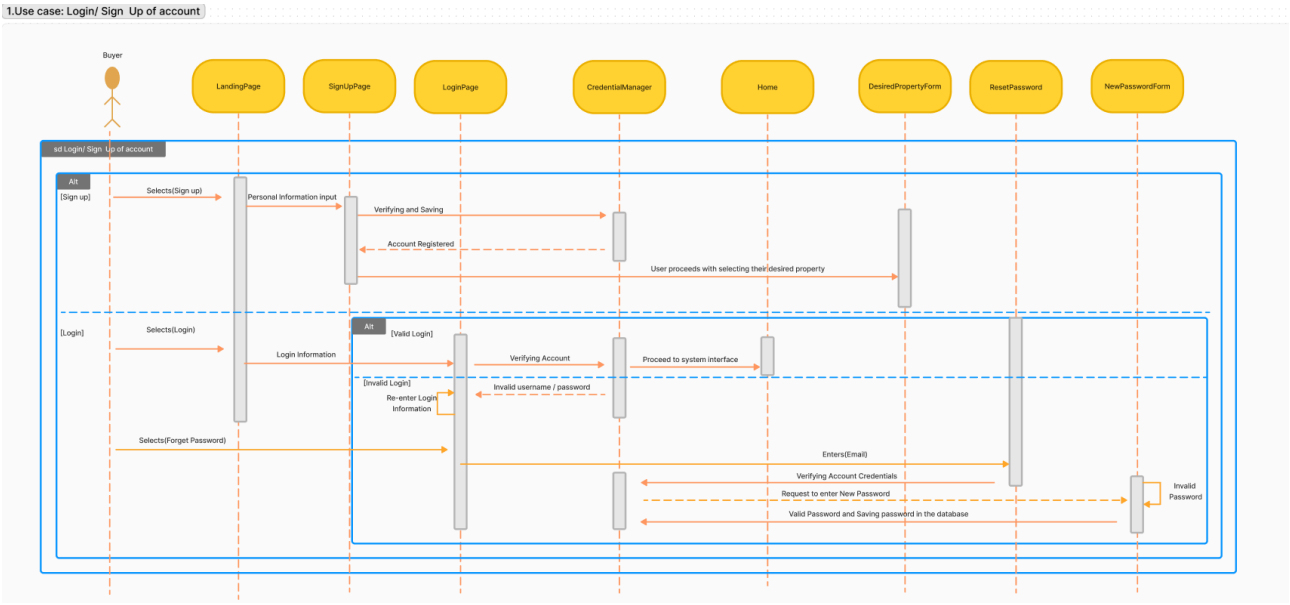
3. Class Diagram

4. Dialog Map

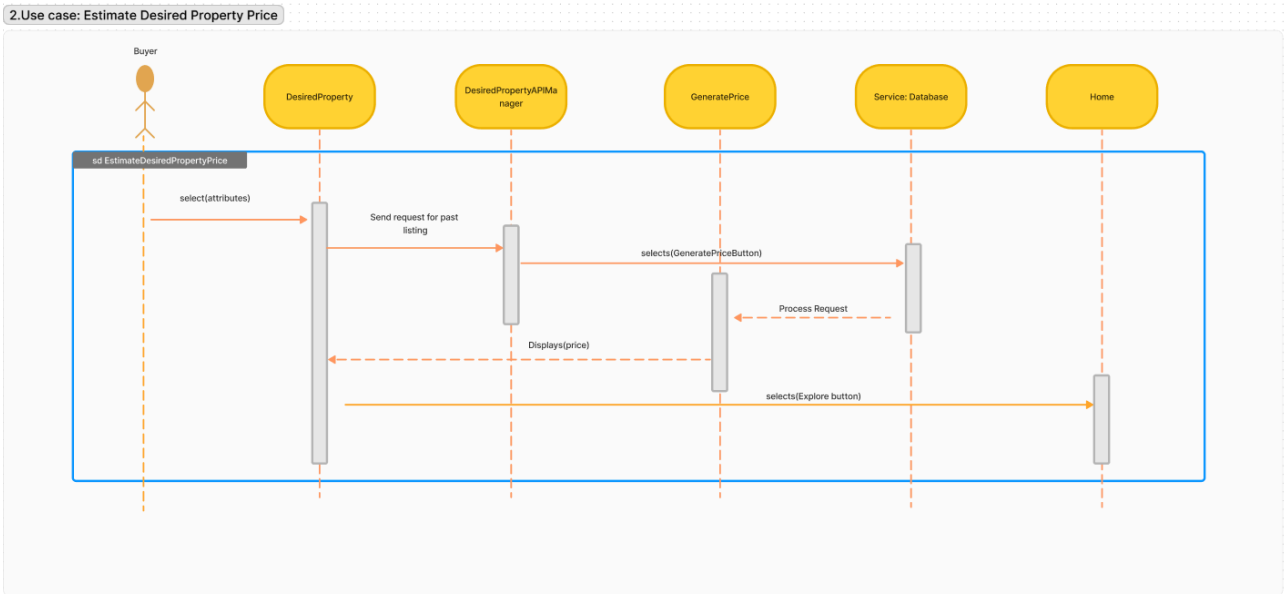

LandingPage
nestlingMainUI
do/display Landing Page
[Click Sign up]
[Click Login]
[Click Forget Password]
SignUp
RegistrationForm
do/ wait for user input
InputValidator
do/ validate user input
[Invalid Input]
[Valid Input]
CredentialManager
do/ verify credentials
[Account do not exists, account created]
[Account exists]
Login
LoginForm
do/ wait for user input
[Invalid credentials]
CredentialManager
do/ verify credentials
[Valid credentials]
Reset Password
ResetForm
do/ wait for user input
CredentialManager
do/verify credentials
[Account does not exists]
[Account exists]
NewPasswordForm
do/ wait for user input
[Invalid Input]
InputValidator
do/verify credentials
[Valid Input]
CredentialManager
do/verify credentials
Price Estimate for desired property
DesiredPropertyForm
do/ wait for user input to filter
DesiredPropertyAPIManager
do/ Fetch properties listing
DesiredPropertyPriceEstimator
do/ Display estimated Price
ExploreButton
do/ To display Homepage
[Button not click]
[Button Click]
Home Page
HomePageInterface
do/ Display app functions
ArticlePiecePage
do/ Display leadership articles
ArticlePieceManager
do/ fetch the details of the article
[Desired property button clicked]
DesiredPropertyButton
do/ display display property form
[Button Click]
[Button not click]
[Current Listing button clicked]
CurrentListingButton
do/ display current listing Page
[Button not click]
[Saved Properties button clicked]
FavouritesButton
do/ display saved properties page
[Button not click]
[Button click]
[Button Click]
Current Listing Page
FilteredListingInterface
do/ wait for user input to filter
[Select 1 property, Scroll down to compare desired vs selected property]
DesiredPropertyForm
do/ Fetch properties listing
[Found Match]
[Found no Match]
PercentageMatchAPIManager
do/ Display property listings
PercentageMatchEstimator
do/ Display estimated percentage beside the property list
IndividualListingPage
do/display selected property description
OpinionGeneratorPage
do/ Display AI Chat Bot opinion
OpinionGenerator
do/ AI chat Bot
Compare Page
CompareListingInterface
do/ Display the 2 property side by side
OpinionGeneratorPage
do/ Compare attributes and display
FavouritesManager
do/fetch the properties from favourites database
OpinionGenerator
do/ fetch reply from OpenAI API
Favorites Page
FavouritesPage
do/Display saved properties
CompareButton
do/ display compare listing page
[!Button Click]
[Select 2 of the properties to compare]
[!Button not Click]
FavouritesManager
do/fetch the properties from favourites database
Saving a property
IndividualListingPage
do/display selected property description
[!Button not Click]
FavouritesButton
do/save the property to FavouritesPage
[!Button Click to save]
FavouritesManager
do/save the property into favourites database
[Clicked Back Button]
FavouritesPage
do/Display saved properties
[Did not Click Back Button]

5. Sequence Diagram

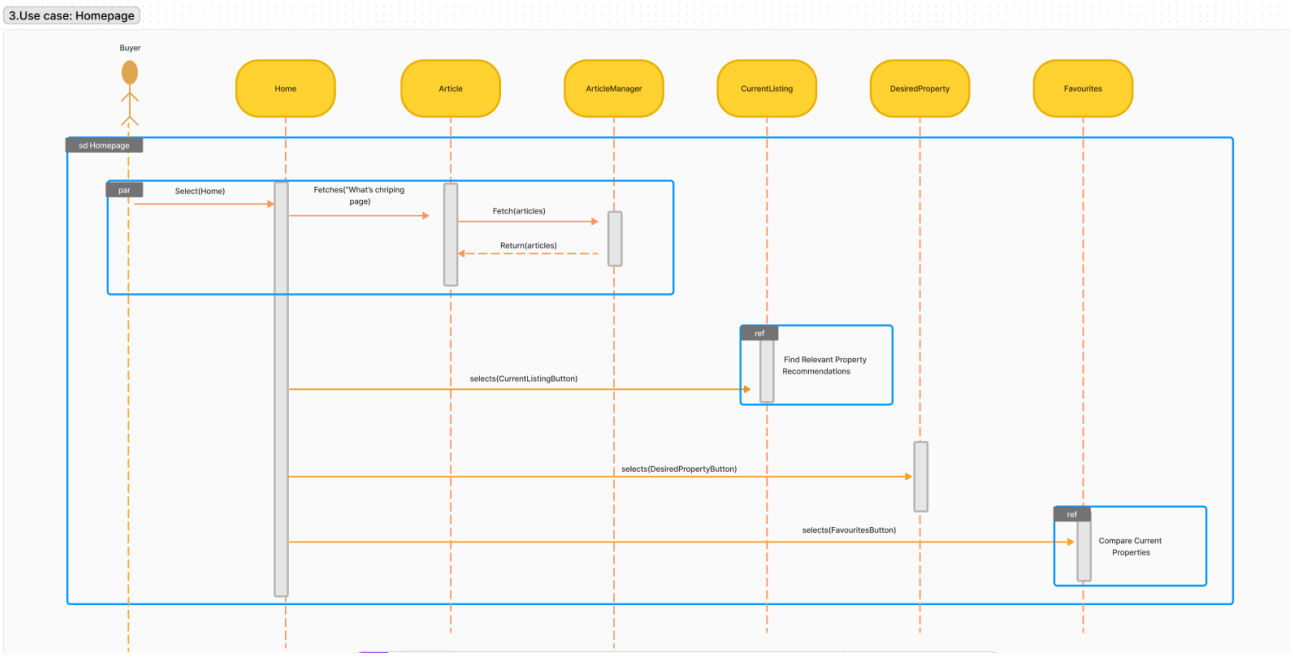
Sequence Diagram 1: Login/Sign up of Account



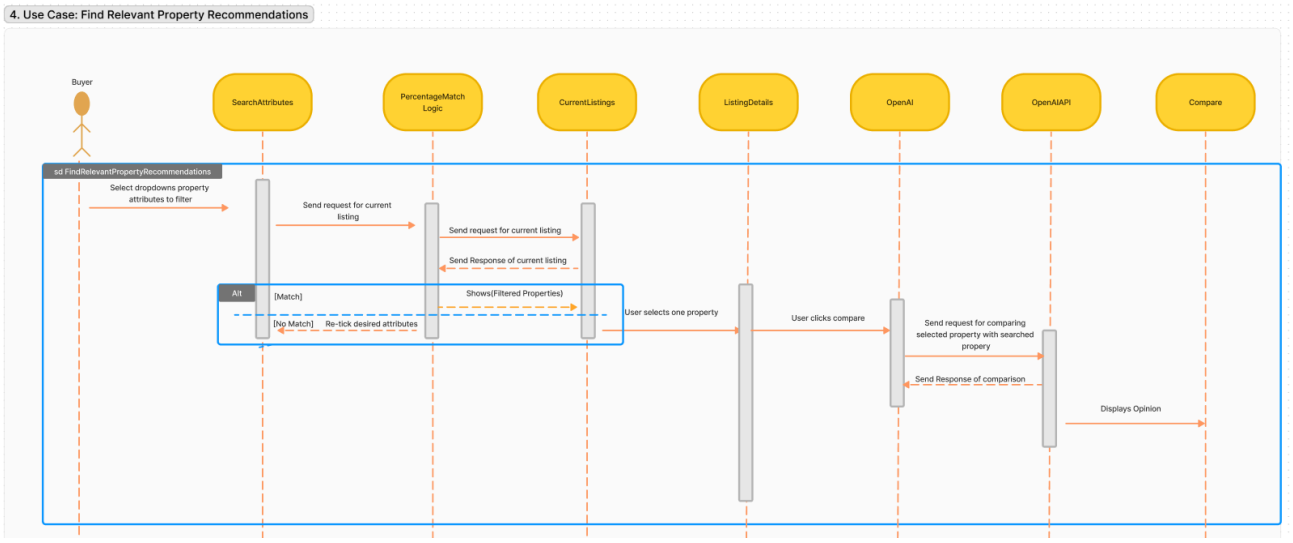
Sequence Diagram 2: Estimate Desired Property Price



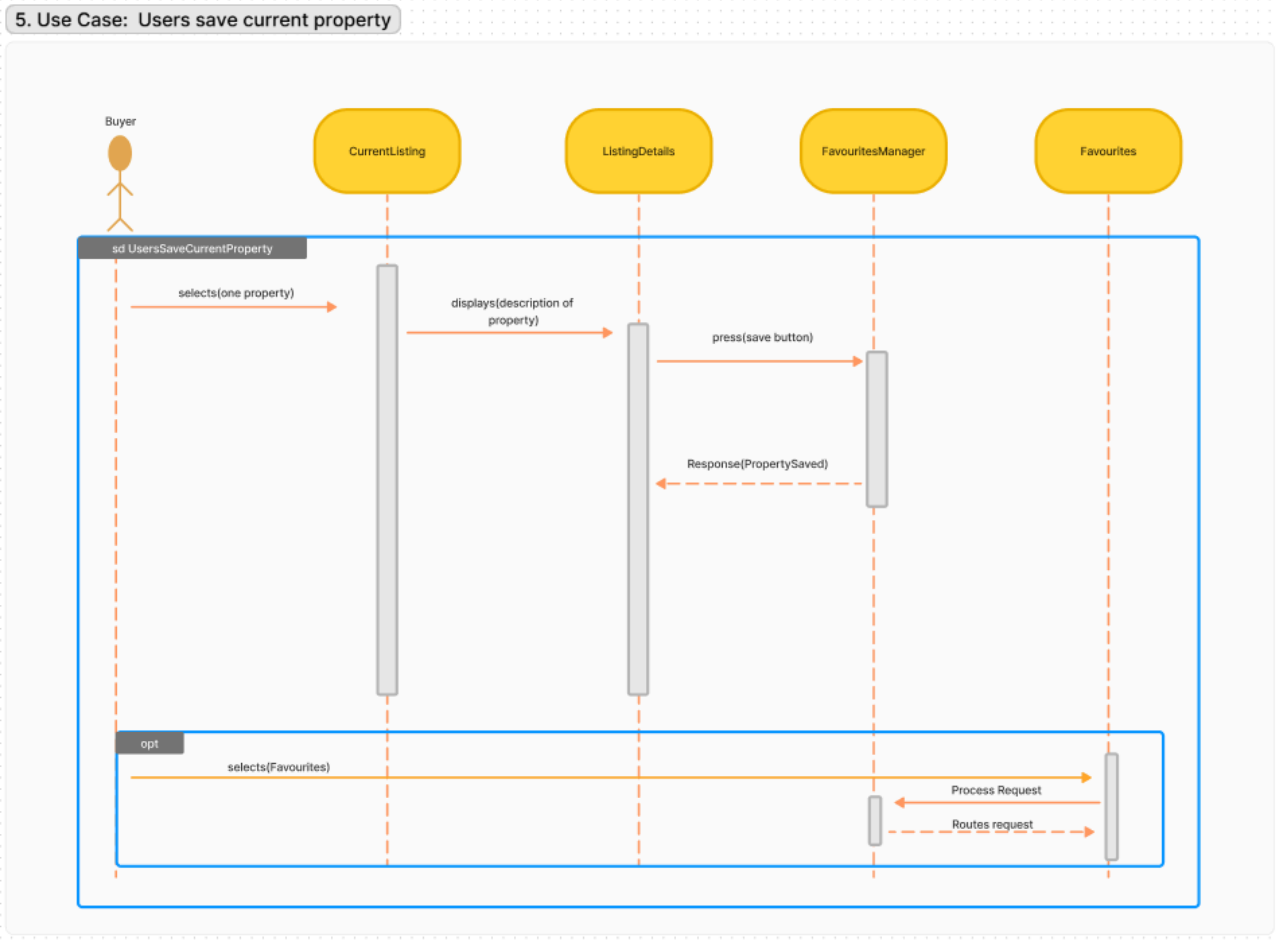
Sequence Diagram 3: Homepage



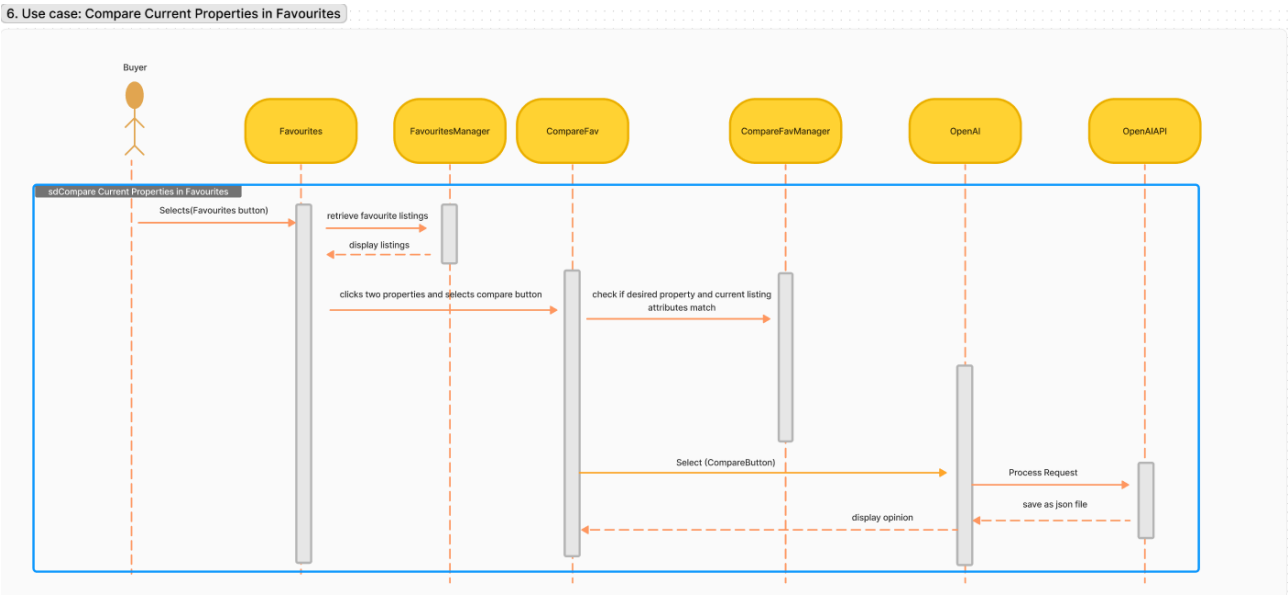
Sequence Diagram 4: Find Relevant Property Recommendations



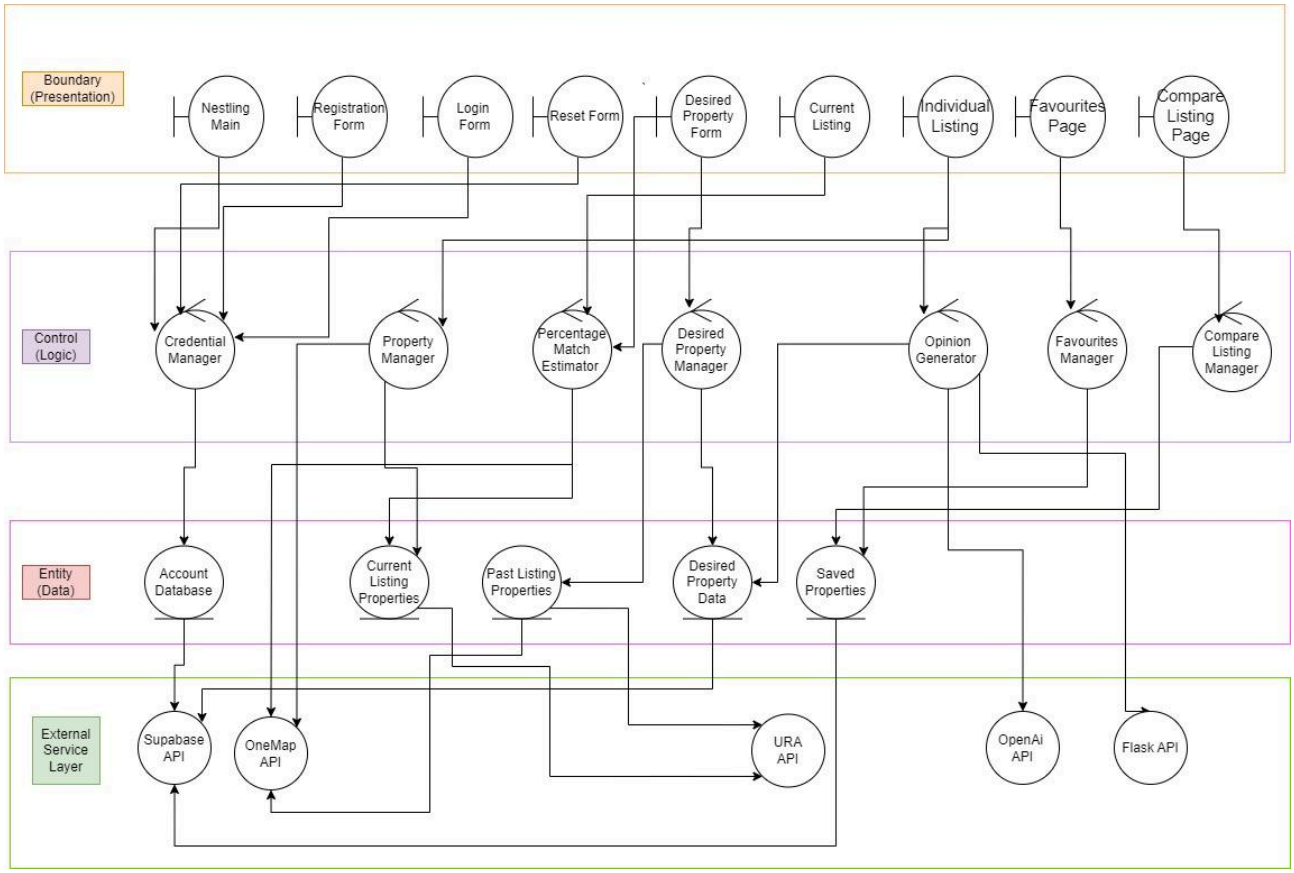
Sequence Diagram 5: Users save current property



Sequence Diagram 6: Compare Current Properties in Favourites



6. System Architecture



Appendix C: Application Skeleton

Component	Main Function
AcctDetails.jsx	Fetch user account details
App.jsx	Handles the page routing for each component
Article.jsx	Displays thought leadership article
CurrentListings.jsx	Displays list of properties being listed for sale currently
DesiredProperty.jsx	● Form for filling in attributes for ideal desired property and then: ○ 1. saving it to backend ○ 2. estimating price using AI
Dropdown.jsx	Individual dropdown component for re-use in multiple components
EmailLogin.jsx	Component to enable log in via user email if user forgot password
Homepage.jsx	● After user logs in, they will be directed to this main 'lobby' ● Contains articles tab, and about us information ● Also contains navigation bar at the top for user to redirect to other services
LandingPage.jsx	● Component for landing page, the first page users see when they first enter the web app ● Responsible for onboarding (sign in, sign up)

ListingInfo.jsx	Component for displaying all individual property details when user clicks in
ListingPanel.jsx	• Reusable component for displaying each property in current listing, mini panel style • Displays picture, property name, price, address
LoginPage.jsx	• Component for user to log in by keying in email and password • Option for user to click forgot password button, which would direct user to email login instead
Navbar.jsx	• Navigation bar component to be stickied at the top of the website for most pages, • For user ease of access to different pages
ResetPassword.jsx	• Component for user to reset password • User will be redirected here after login via email if forgotten password
SignUp.jsx	Component to onboard new users by signing up
percentageMatchLogic.jsx	Logic for determining percentage match between each current listing property, compared to user desired house attributes

Appendix D: Testing

Black Box Testing: Equivalence Class and Boundary Value Testing

Control class tested: Percentage Match Estimator

Gross Floor Area (sqm)

Equivalence class partitioning:

Invalid Partition	Valid Partition	Invalid Partition
• 0 • -1 • -2 • …	• 1 • 2 • … • 4999 • 5000	• 5001 • 5002 • 5003 • …

Boundary value testing:

Invalid Partition - Valid Partition lower boundary		Invalid Partition - Valid Partition Upper boundary	
Boundary Value just below boundary	Boundary Value just above boundary	Boundary Value just below boundary	Boundary Value just above boundary
0	1	5000	5001

Room Count

Equivalence class partitioning:

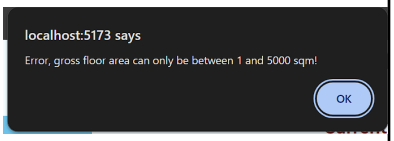
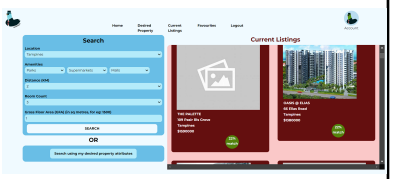
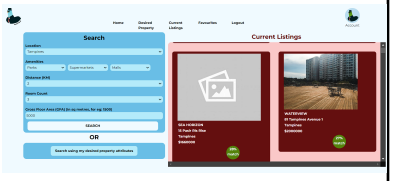
Invalid Partition	Valid Partition	Invalid Partition
• 0 • -1 • -2 • …	• 1 • 2 • 3 • 4	• 6 • 7 • …

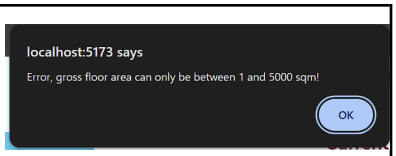
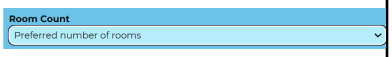
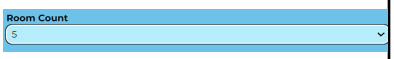
	• 5	

Boundary value testing:

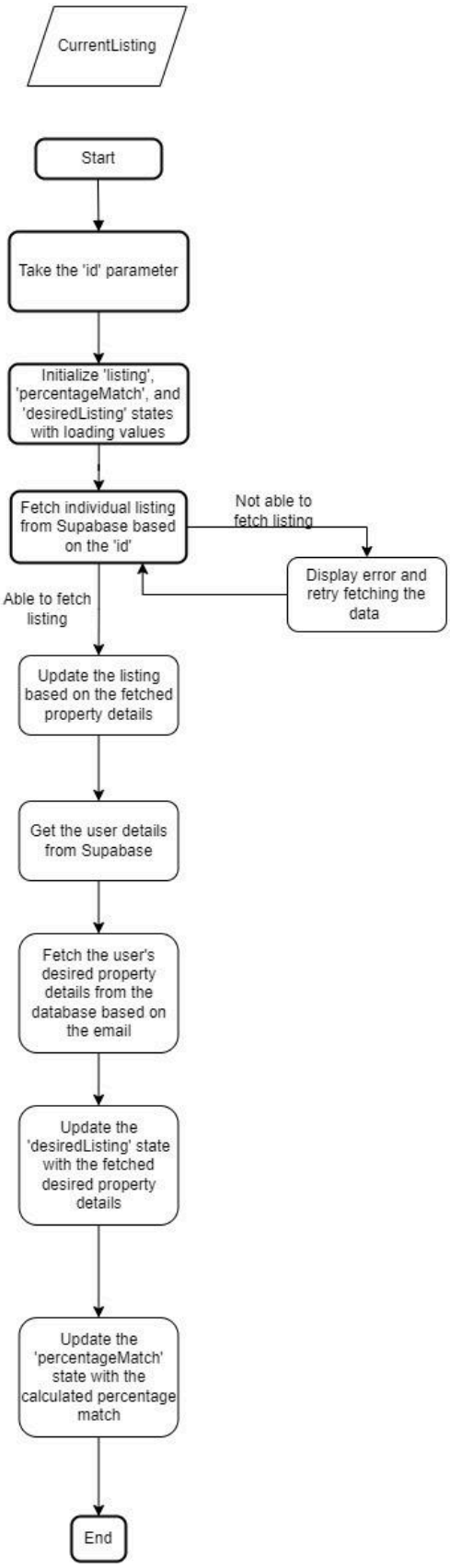
Invalid Partition - Valid Partition lower boundary		Invalid Partition - Valid Partition Upper boundary	
Boundary Value just below boundary	Boundary Value just above boundary	Boundary Value just below boundary	Boundary Value just above boundary
0	1	5	6

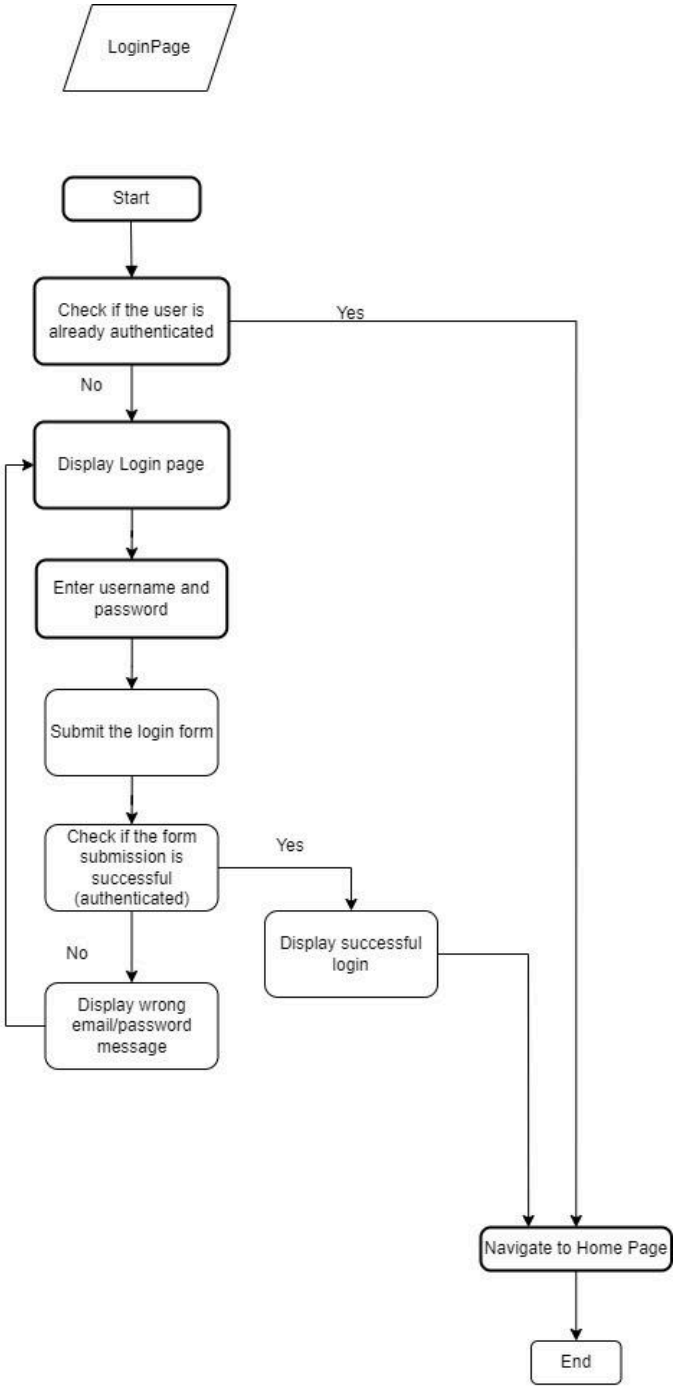
Test Cases for Gross Floor Area (sqm) and Room Count

Test Input	**Expected Output**	**Actual Output**
Input Gross Floor Area as 0	Alert: Error, gross floor area can only be between 1 and 5000 sqm!	
Input Gross Floor Area as 1	Lists of matching properties are generated successfully	
Input Gross Floor Area as 5000	Lists of matching properties are generated successfully	

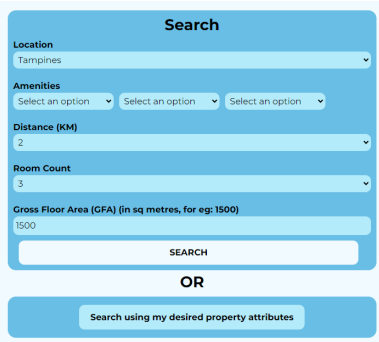
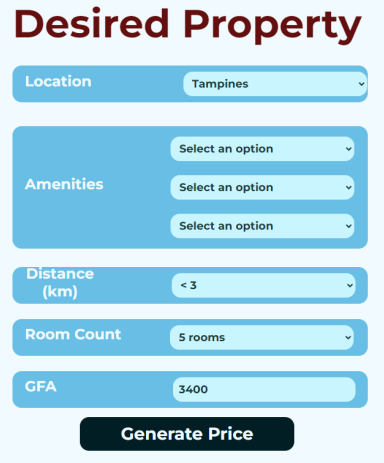
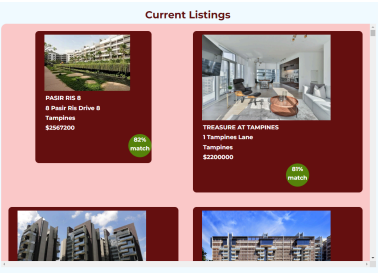
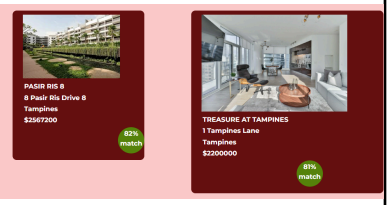
Input Gross Floor Area as 5001	Alert: Error, gross floor area can only be between 1 and 5000 sqm!	
Input Room Count as 0	Such an option does not exist in the dropdown menu. Dropdown will thus default to an empty state	
Input Room Count as 1	The dropdown option selected successfully	
Input Room Count as 5	The dropdown option selected successfully	
Input Room Count as 6	Such an option does not exist in the dropdown menu. Dropdown will thus default to an empty state	

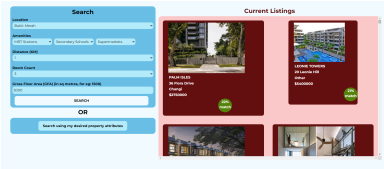
White Box Testing


CurrentListing
Start
Take the 'id' parameter
Initialize 'listing', 'percentageMatch', and 'desiredListing' states with loading values
Fetch individual listing from Supabase based on the 'id'
Not able to fetch listing
Display error and retry fetching the data
Able to fetch listing
Update the listing based on the fetched property details
Get the user details from Supabase
Fetch the user's desired property details from the database based on the email
Update the 'desiredListing' state with the fetched desired property details
Update the 'percentageMatch' state with the calculated percentage match
End

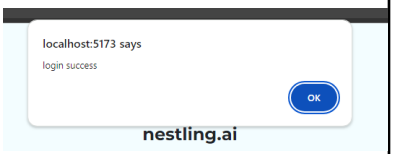
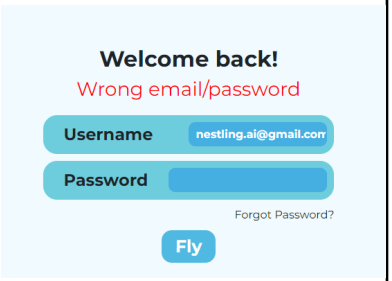
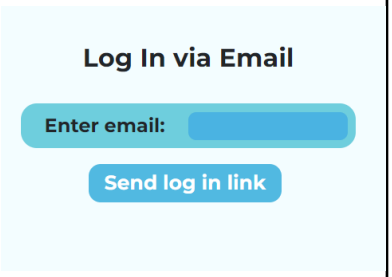
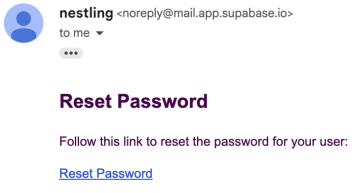

LoginPage
Start
Check if the user is already authenticated
Yes
No
Display Login page
Enter username and password
Submit the login form
Check if the form submission is successful (authenticated)
Yes
No
Display successful login
Display wrong email/password message
Navigate to Home Page
End

Test Cases for CurrentListing function:

Test Input	Expected Output	Actual Output
The user submits the desired property form and searches by clicking 'Search using my desired property attributes'	Able to view the attributes of the user's desired property	 
The user submits the desired property form and searches by clicking 'Search using my desired property attributes'	Able to view all the current listings	
The user submits the desired property form and searches by clicking 'Search using my desired property attributes'	Able to view the current listings with a percentage match	

The user searches using the 'Search' button	Should display all the current listings with the attributes matched	

Test Cases for LoginPage function:

Test Input	Expected Output	Actual Output
The user is authenticated (Successful)	Directed to the Homepage	Directed to the Homepage
The user is not authenticated (Unsuccessful)	Directed to Login Page	Directed to Login Page
Input Validation Valid Input and Password	Directed to the Homepage after displaying 'Login Success'	
Input Validation Invalid Input and/or Password	Directed to Login Page	
Clicked 'Forget Password' to reset the password	Directed to forget password page to enter new password after clicking the verification email sent to the user's email	 

References

BairesDev. (n.d.). The Psychology Behind UX/UI Design. Retrieved from https://www.bairesdev.com/blog/psychology-behind-ux-ui-design/#

Visual Paradigm. (n.d.). Code First vs. Design First: Which One Is Better? Retrieved from https://www.visual-paradigm.com/guide/development/code-first-vs-design-first/#:~:text=The%20Design%20First%20approach%20advocates,use%20of%20OpenAPI%20specification%20formats.