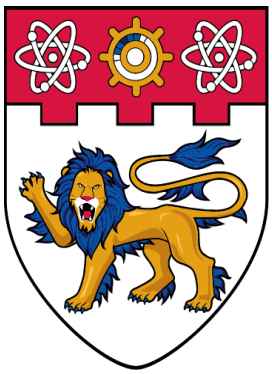

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

School of Computer Science and Engineering

CE2006 Software Engineering
Lab Project Report

Lab Group: SEP2
Group Name: 710

Group Members:
Ahmad Jazli Bin Abdul Razak (U1921863L)
Deng Zi Yang (U1923770A)
Elayne Tan Hui Shan (U1921730C)
Khin Nway Htway (U1921268F)

Table of Contents

1. Application Introduction

1.1 Purpose

Figure 1: NestScout Logo

The NestScout team aims to develop a hassle-free community platform that provides individuals with access to an online property market. This application will allow individuals to meet various needs of buying, selling, renting, or renting out a property, without the need of an agent.

1.2 Target Users

1. Buyers who wish to purchase a property
2. Sellers who wish to sell a property
3. Tenants who wish to rent a property
4. Landlords who wish to rent out a property

1.3 Dataset

We utilised government data API in our property listings database, retrieved from https://data.gov.sg/dataset/resale-flat-prices.
We also utilised a publicly available data API, Google Maps, to display relevant location information regarding the property listings.

1.4 Assumptions

1. Users should have internet access to use the application.
2. For the application to be successful, a significant number of sellers/landlords should be actively selling/renting out their properties.

1.5 Constraints

1. The application is currently available only in English.

2 Functional Requirements

2.1 Buyers

1.1 Buyers must be able to search for property they wish to purchase by entering their preferred location.

1.1.1 The preferred location data must be text of at least one character and less than 512 characters.

1.1.2 The system must display all available property with keywords relevant to the preferred location data.

1.2 Buyers must be able to browse for property they wish to purchase by selecting at least one option in one of the categories provided.

1.2.1 Buyers should be able to choose as many options as they wish for the categories provided (Town, Type, Flat Model).

1.2.2 Buyers should be able to specify the price range they want by entering the minimum and maximum data.

1.2.2.1 The minimum and maximum data must be an integer between 0 and 9,999,999. The data shall be in Singapore Dollars (SGD).

1.2.3 The system must display all available property based on the buyer's inputs.

2.2 Tenants

2.1 Tenants must be able to search for property they wish to rent by entering their preferred location.

2.1.1 The preferred location data must be text of at least one character and less than 512 characters.

2.1.2 The system must display all available property with keywords relevant to the preferred location data.

2.2 Tenants must be able to browse for property they wish to rent by selecting at least one option in one of the categories provided.

2.2.1 Tenants should be able to choose as many options as they wish for the categories provided (Town, Type, Flat Model).

2.2.2 Tenants should be able to specify the price range they want by entering the minimum and maximum data.

2.2.2.1 The minimum and maximum data must be an integer between 0 and 9,999,999. The data shall be in SGD.

2.2.3 The system must display all available property based on the tenant's inputs.

2.3 Sellers

3.1 When sellers wish to sell their property, they must be able to create new property listings by entering information about their property in the system.

3.1.1 Sellers shall enter the address of their property.

3.1.1.1 The address data must be text of at least one character and less than 512 characters.

3.1.1.2 The system must display at least a part of the property address under the search and browse pages, and the property address in full under the property's individual page.

3.1.2 Sellers shall enter the type of their property.

3.1.2.1 The property type data must be only one of the types given to them to choose from (e.g. 1 Room, 2 Room, etc.).

3.1.2.2 The system must display the property type data under the search and browse pages, as well as the property's individual page.

3.1.3 Sellers shall enter the flat model of their property.

3.1.3.1 The property flat model data must be only one of the models given to them to choose from (e.g. Adjoined Flat, Apartment, etc.).

3.1.3.2 The system must display the property flat model data under the search and browse pages, as well as the property's individual page.

3.1.4 Sellers shall enter the price they would like to sell their property at.

3.1.4.1 The property price data must be an integer between 0 and 9,999,999. The property price shall be in SGD.

3.1.4.2 The system must display the property price data in SGD in the range $0 to $9,999,999 to the nearest dollar under the search and browse pages, as well as the property's individual page.

3.1.5 Sellers shall enter descriptions about their property.

3.1.5.1 The description data must be text of at least one character and less than 3000 characters.

3.1.5.2 The system must display the description data under the property's individual page.

2.4 Landlords

4.1 When landlords wish to rent out their property, they must be able to create new property listings by entering information about their property in the system.

4.1.1 Landlords shall enter the address of their property.

4.1.1.1 The address data must be text of at least one character and less than 512 characters.

4.1.1.2 The system must display at least a part of the property address under the search and browse pages, and the property address in full under the property's individual page.

4.1.2 Landlords shall enter the type of their property.

4.1.2.1 The property type data must be only one of the types given to them to choose from (e.g. 1 Room, 2 Room, etc.).

4.1.2.2 The system must display the property type data under the search and browse pages, as well as the property's individual page.

4.1.3 Landlords shall enter the flat model of their property.

4.1.3.1 The property flat model data must be only one of the models given to them to choose from (e.g. Adjoined Flat, Apartment, etc.).

4.1.3.2 The system must display the property flat model data under the search and browse pages, as well as the property's individual page.

4.1.4 Landlords shall enter the price they would like to sell their property at.

4.1.4.1 The property price data must be an integer between 0 and 9,999,999. The property price shall be in SGD.

4.1.4.2 The system must display the property price data in SGD in the range $0 to $9,999,999 to the nearest dollar under the search and browse pages, as well as the property's individual page.

4.1.5 Landlords shall enter descriptions about their property.

4.1.5.1 The description data must be text of at least one character and less than 3000 characters.

4.1.5.2 The system must display the description data under the property's individual page.

2.5 Map Generation

5. When a property is listed by sellers or landlords, the system must be able to generate maps under the property's individual page (using Street Directory API) that show information about the property and its surroundings.

2.6 Accounts

6.1 To sell or rent out a property, sellers and landlords must have accounts. To buy or rent a property, buyers and tenants must have accounts. They may not have accounts when only searching or browsing.

6.1.1 Users without existing accounts must be able to create accounts via the sign up page.

6.1.1.1 Users must enter their preferred username.

6.1.1.2 Users must enter their preferred password.

6.1.1.3 Users must enter their email. The system shall verify the user's email during the signup process.

6.1.1.4 Users must enter their phone number. The system shall verify the user's phone number during the signup process.

6.1.1.5 Users must enter their name.

6.1.1.6 Users must enter their gender.

6.1.1.7 Users must enter their date of birth.

6.1.1.8 Buyers must enter their marital status.

6.1.2 Users with existing accounts must be able to login to their accounts with their username or email, and password.

2.7 User Functions

7.1 Users must be able to change their account information in the profile page.

7.2 Sellers and landlords should be able to view their listings via their account page.

7.2.1 Sellers and landlords must be able to view their individual property listing page when they click on the individual property in the listing page.

3 Non-Functional Requirements

3.1 Usability Requirements

1.1 The application should provide informative feedback.

1.1.1 The application should display help messages in English when the data keyed in does not meet the system requirements.

1.1.2 The application should display an appropriate error message when certain processes fail.

1.2 The application should reduce short-term memory load for users.

1.2.1 The application should keep the display simple and allow users to retrieve their saved listings easily.

1.3 The application should have a consistent user interface.

1.3.1 The application should require a consistent sequence of actions for similar situations.

1.3.2 The application should adopt a consistent visual layout and use identical terminologies throughout.

1.4 The application should permit easy reversal of actions.

1.4.1 Users should be able to save and remove property listings easily.

3.2 Performance Requirements

2.1 The application should be maintained regularly with minimal downtime incurred.

2.2 The application should be free of critical errors (e.g. crashing).

2.3 The application should be able to support internal locus of control.

2.3.1 The application should be fully responsive always, with reasonably short loading time and latency.

2.3.1.1 When a search query is entered, the system must display the results in 5 seconds.

2.3.2 The application should be able to give users a sense of control of events occurring and that the application will behave as expected.

3.3 Security Requirements

3.1 The application will authenticate users through a login process before granting them access to the application, except for the browsing function where guests can browse properties without an account.

3.2 The application will mask the password field to prevent any shoulder surfing.

3.3 The application will only allow users to access and edit their own account/property information.

3.4 Extendibility Requirements

4.1 The application should be designed with the Model-View-Controller (MVC) architecture and proper design patterns to support future enhancements.

4.2 The application should separate the data collected from the application and store this data in an online database to facilitate data access using other platforms.

4 Use Case Model

4.1 Use Case 1 – Guest Search/Browse

Use Case ID:	1		
Use Case Name:	Guest Search/Browse		
Created By:	Elayne	Last Updated By:	Elayne
Date Created:	2nd September 2020	Date Last Updated:	22nd October 2020

Actor:	Guest
Description:	Guest search or browse properties using NestScout
Preconditions:	1. The guest has internet access to the MainMenuUI.
Postconditions:	1. Guest manages to find out if there are suitable properties for him.
Priority:	High
Frequency of Use:	0-10 times per day
Flow of Events:	1. The guest accesses the MainMenuUI through the internet. 2. The guest chooses to search or browse properties. 3. The guest types in address to search or select categories to browse. These inputs are sent to the BrowseSearchMgr. 4. The BrowseSearchUI displays available properties based on the guest's inputs. 5. The guest finds suitable properties.
Alternative Flows:	AF-S4: If there are no available properties based on the guest's inputs 1. The BrowseSearchUI displays the message "There are no listed properties that match your descriptions. Please try again." 2. The BrowseSearchUI returns to step 3.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

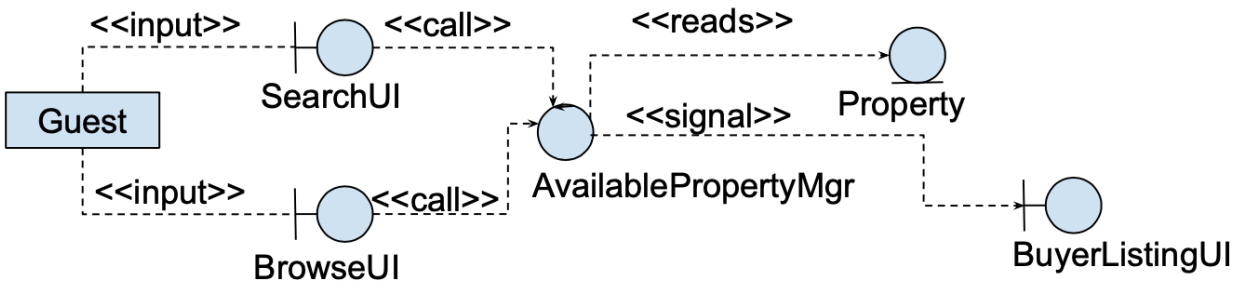

Figure 2: Use Case 1 Stereotype Classes

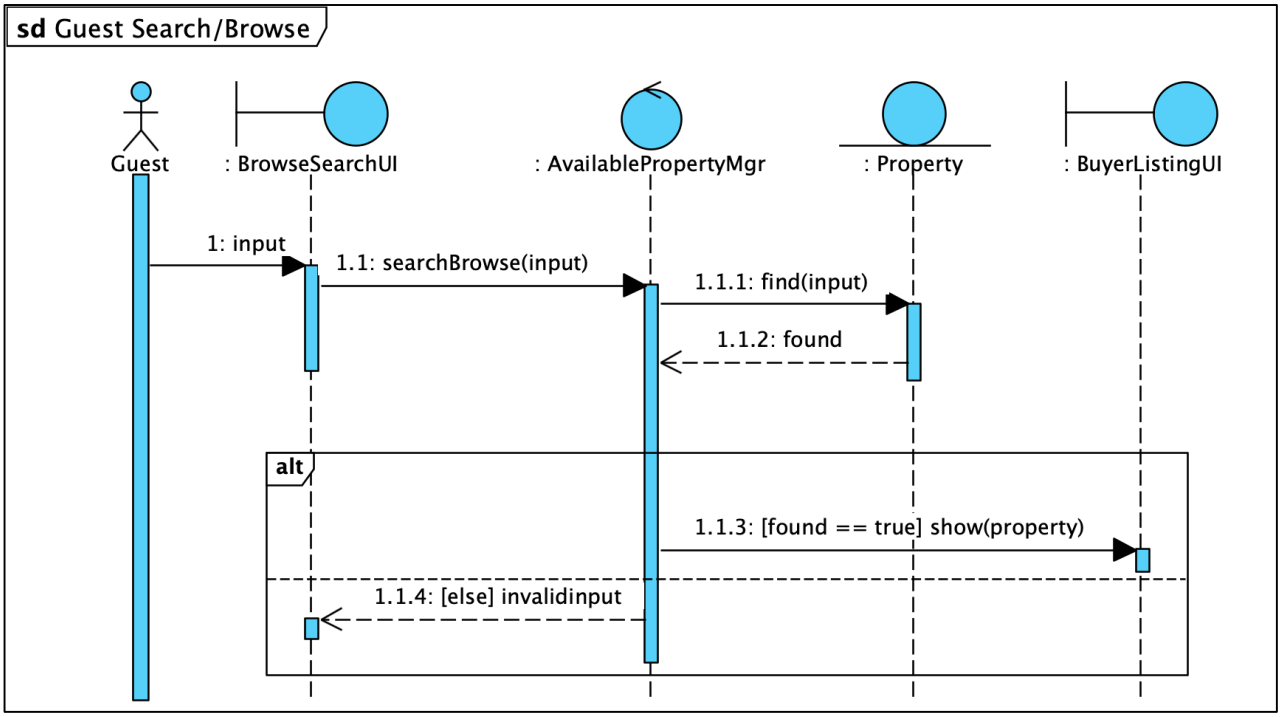

Figure 3: Use Case 1 Sequence Diagram

4.2 Use Case 2 – Buyer/Tenant Search

Use Case ID:	2		
Use Case Name:	Buyer/Tenant Search		
Created By:	Khin	Last Updated By:	Elayne
Date Created:	2nd September 2020	Date Last Updated:	22nd October 2020

Actor:	Buyer/Tenant
Description:	Buyer/Tenant searching for a new property
Preconditions:	1. Buyer/Tenant has internet access to the MainMenuUI. 2. Buyer/Tenant has a verified account.
Postconditions:	1. Buyer/Tenant manages to find out if there are suitable properties for him.
Priority:	High
Frequency of Use:	0-10 times per day
Flow of Events:	1. Buyer/Tenant registers and verifies an account with SignUpLoginUI. 2. Buyer/Tenant logs into the SignUpLoginUI. 3. Buyer/Tenant chooses to search or browse properties. 4. BrowseSearchUI displays properties from user inputs. 5. Buyer/Tenant finds a suitable listing and chooses if he wants to save them.
Alternative Flows:	AF-S4: If user input does not produce any results 1. System displays message “There are no listed properties that match your descriptions. Please try again.” 2. System returns to step 3.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

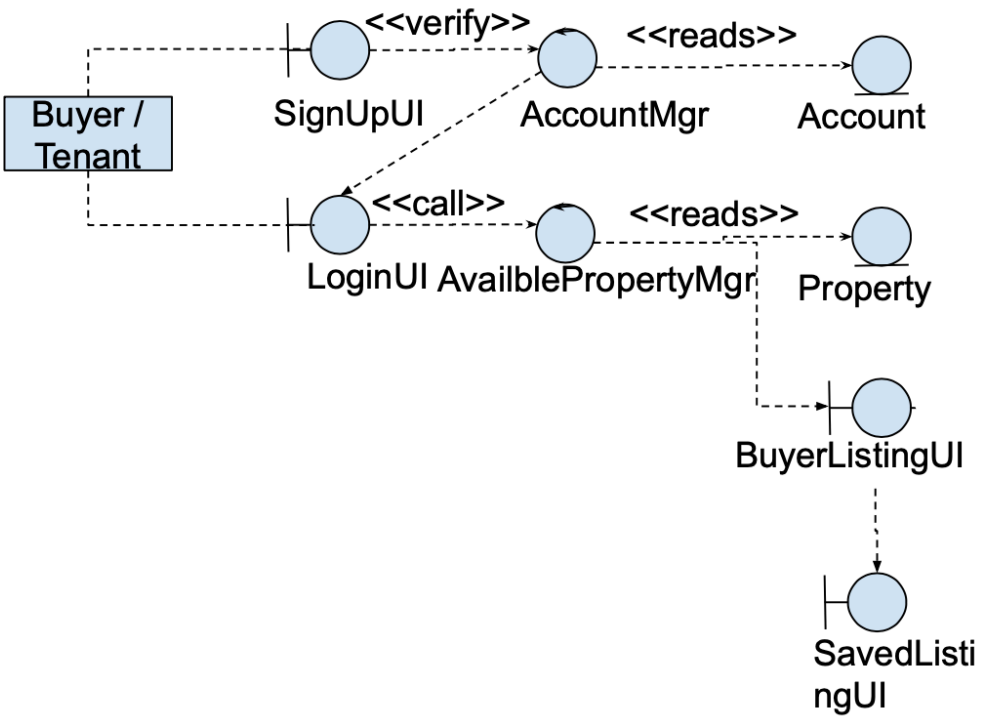

Figure 4: Use Case 2 Stereotype Classes

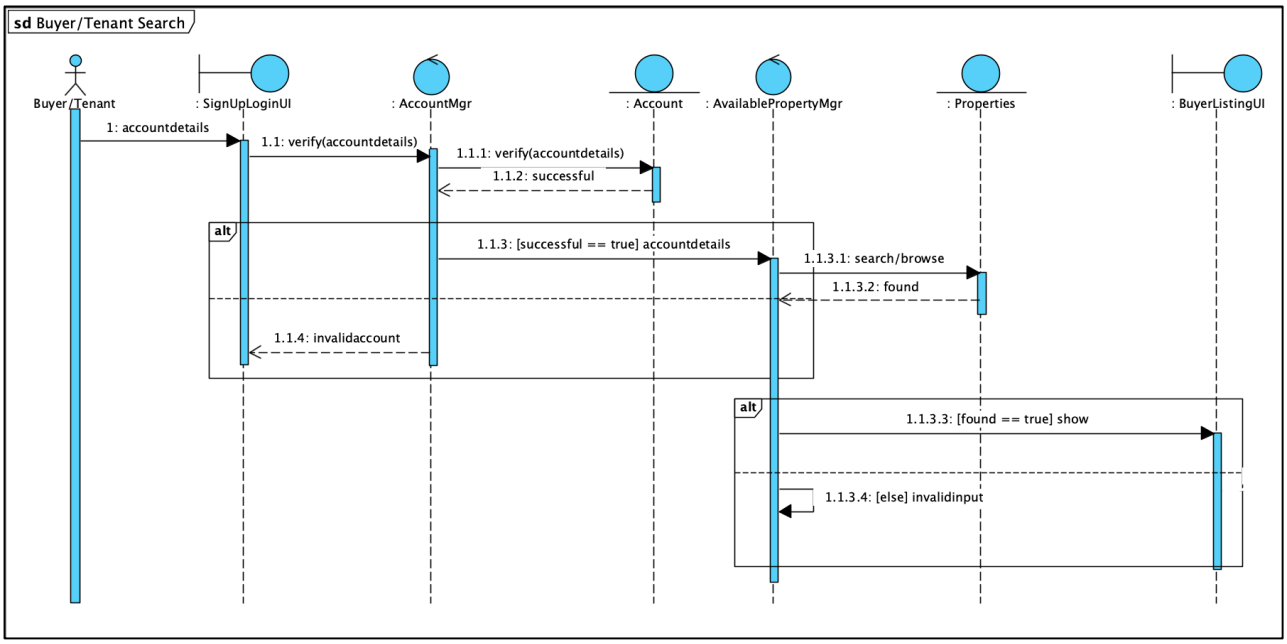

Figure 5: Use Case 2 Sequence Diagram

4.3 Use Case 3 – Seller/Landlord Sell/Rent Out

Use Case ID:	3		
Use Case Name:	Seller/Landlord Sell/Rent Out		
Created By:	Jazli	Last Updated By:	Elayne
Date Created:	2nd September 2020	Date Last Updated:	22nd October 2020

Actor:	Seller/Landlord
Description:	Seller/Landlord selling or renting out a property
Preconditions:	1. Seller/Landlord has internet access to the MainMenuUI 2. Seller/Landlord has a verified account 3. Seller/Landlord has a place to sell/rent out
Postconditions:	1. Seller/Landlord successfully lists a place to sell/rent out
Priority:	High
Frequency of Use:	0-20 times per day
Flow of Events:	1. Seller/Landlord registers and verifies account or login to existing account via SignUpLoginUI. 2. AccountMgr processes and logs in Seller/Landlord. 3. Seller/Landlord creates a new listing with SellerListingUI for apartment to rent/sell 3.1 Seller/Landlord includes location, type of apartment, description, features, amenities etc. 4. Seller/Landlord publishes listing. 5. Seller updates/deletes listing.
Alternative Flows:	AF-S4: Address listed is not a valid address 1. SellerListingUI displays the message “This address is invalid. Please try again.” 2. SellerListingUI returns to step 3. AF-S4: Some information about apartment is missing 1. SellerListingUI displays the message “Please fill in all required information.” 2. SellerListingUI returns to step 3.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	We assume that if the user wants to update a listing, they will have to delete and recreate a new listing.
Notes and Issues:	-

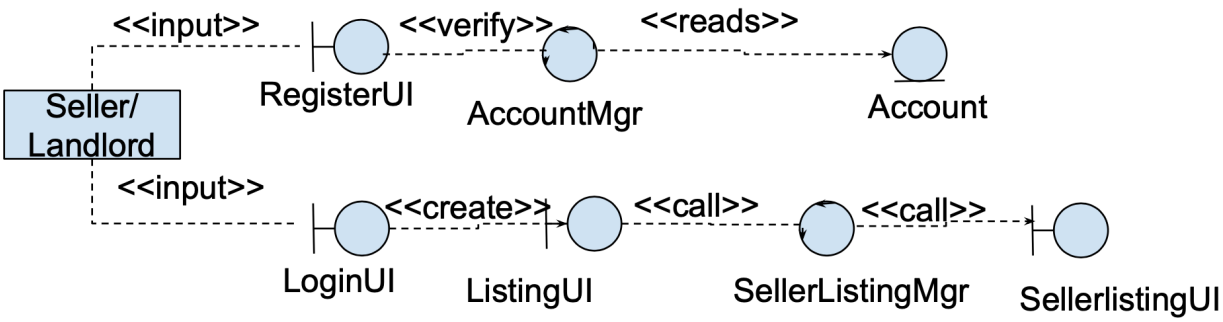

Figure 6: Use Case 3 Stereotype Classes

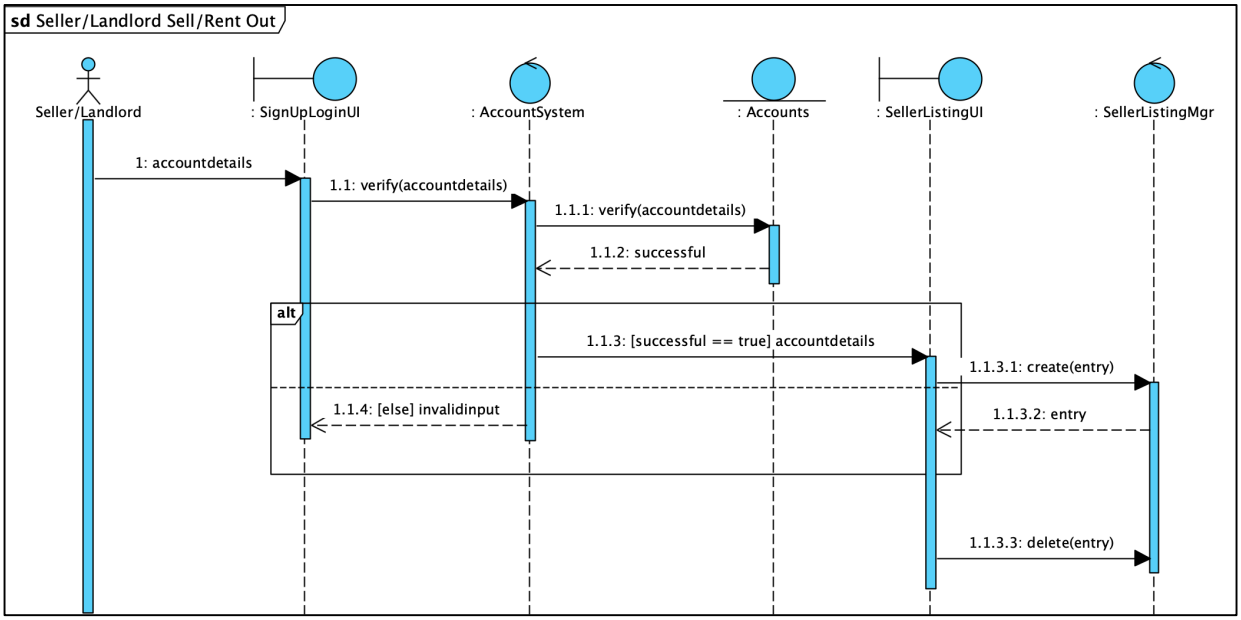

Figure 7: Use Case 3 Sequence Diagram

4.5 Use Case Diagram

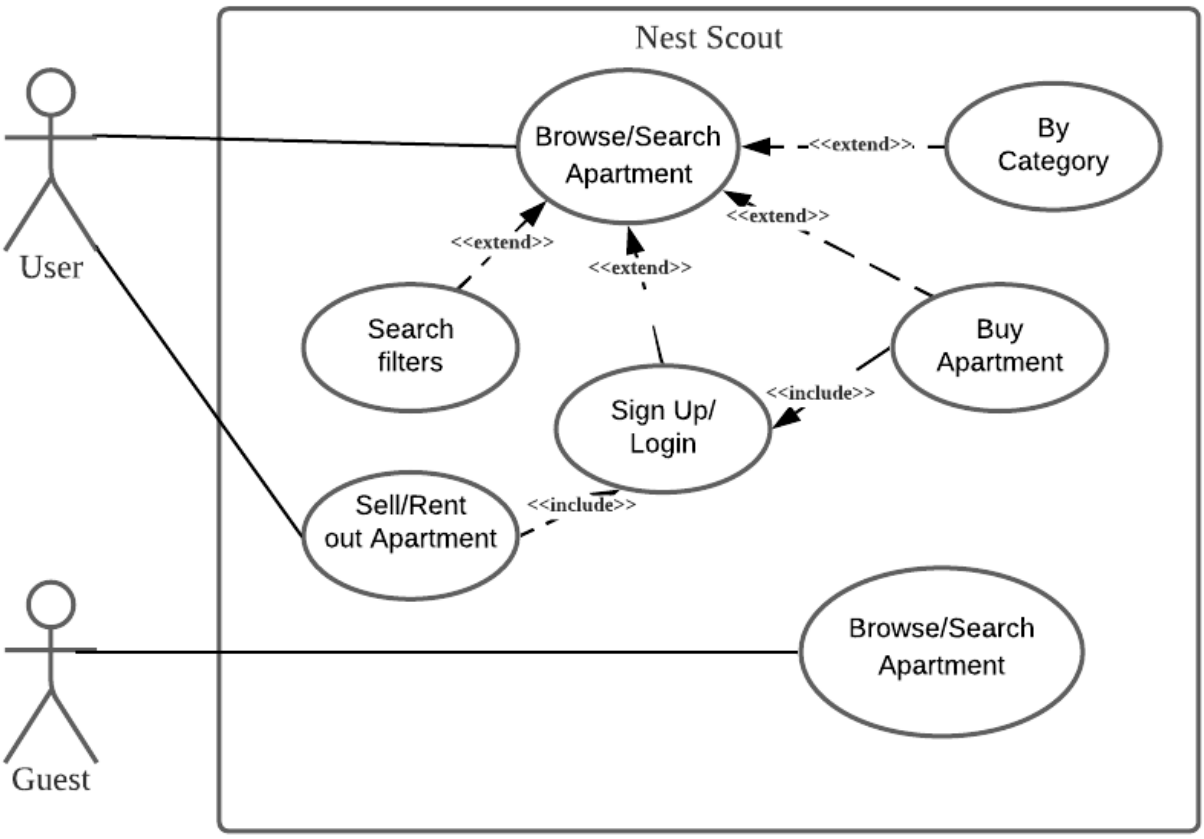

Figure 8: Use Case Diagram

4.6 Dialog Map

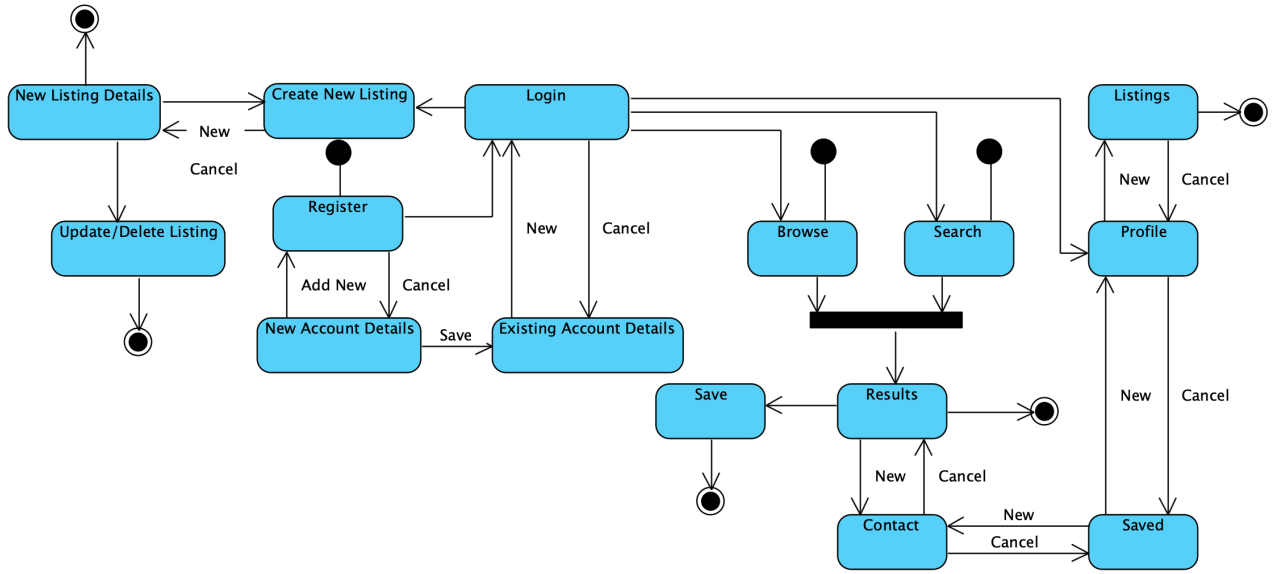

Figure 9: Dialog Map

4.7 Class Diagram

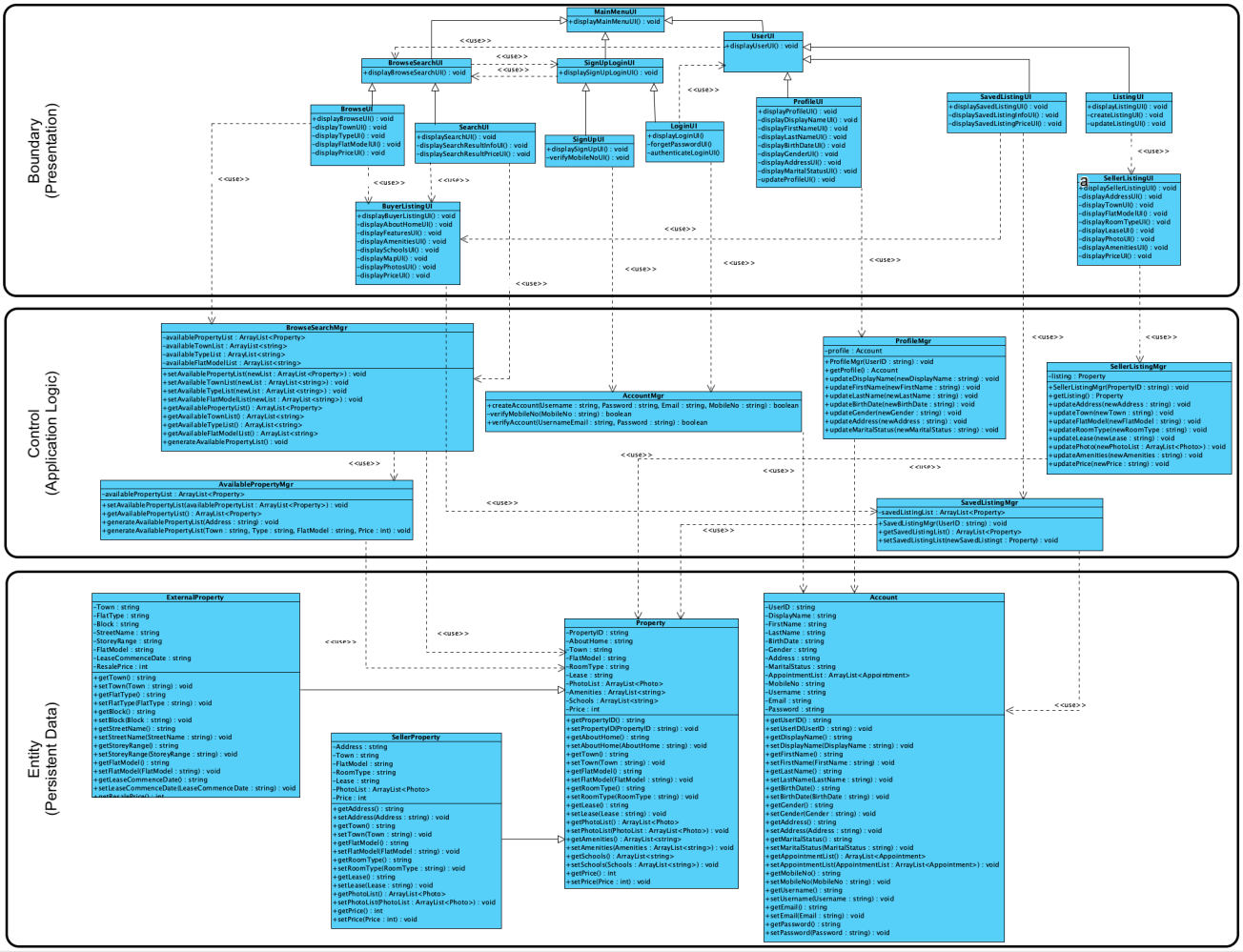

Figure 10: Class Diagram

4.8 System Architecture

4.8.1 Layer Architecture

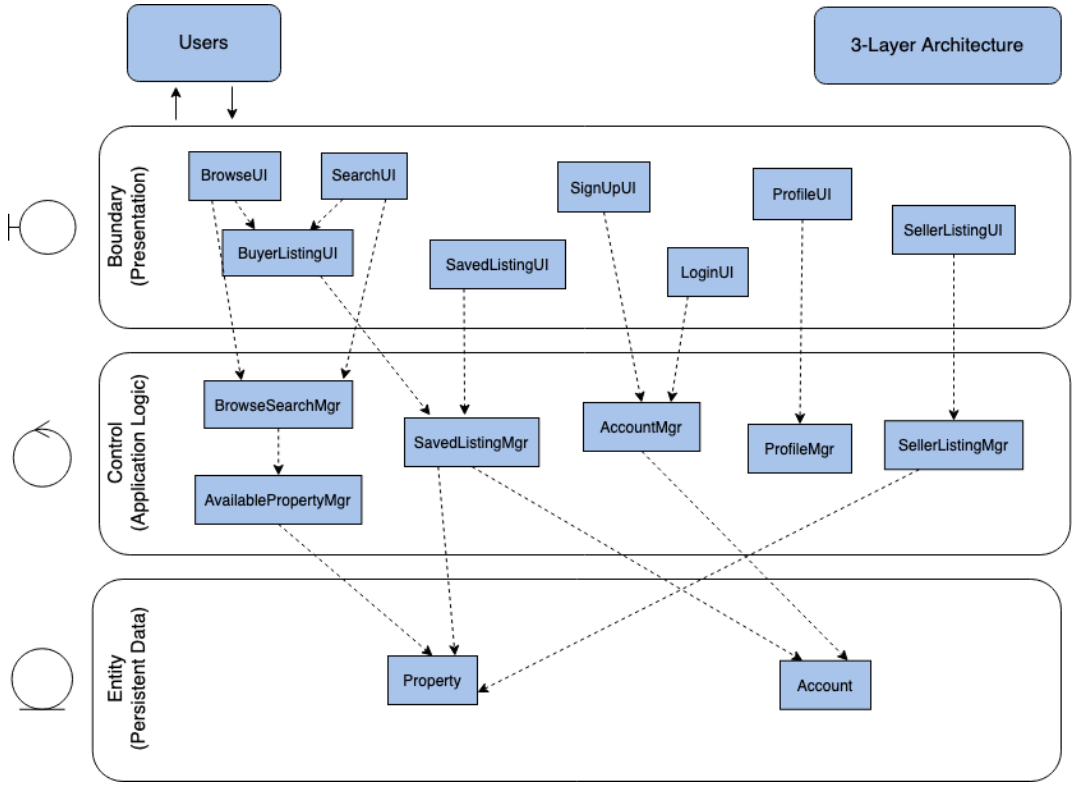

Figure 11: Layer Architecture

4.8.2 System Architecture

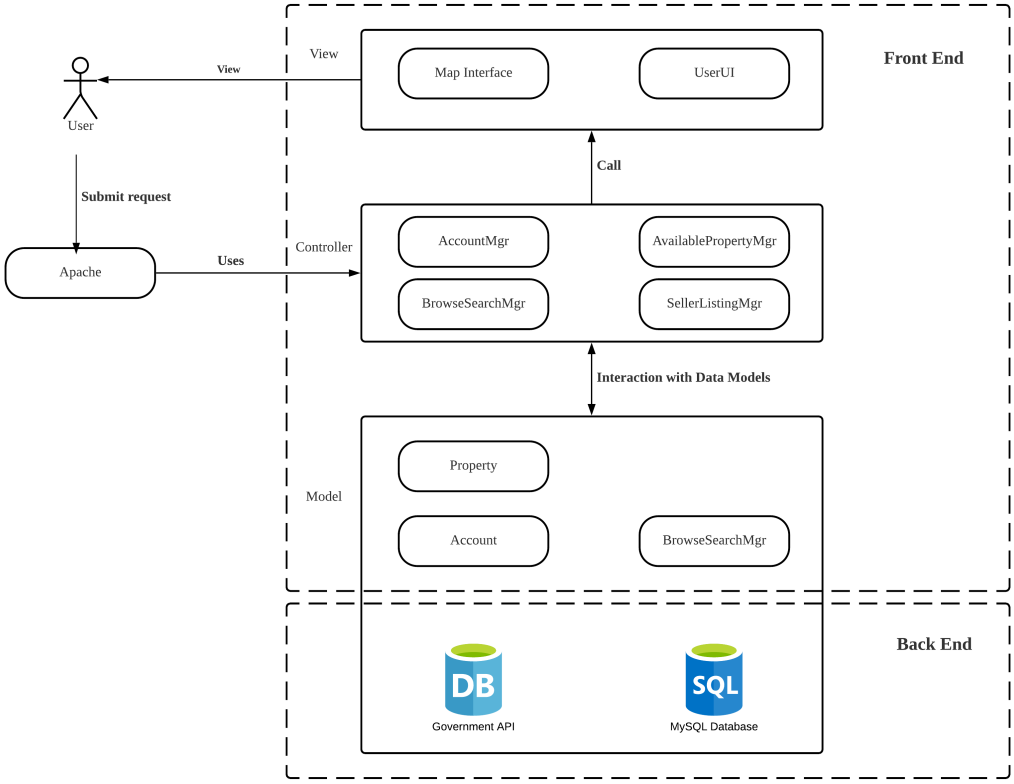

Figure 12: System Architecture

5 Design Patterns and Practices

5.1 Design Considerations

5.1.1 Façade Design Pattern

The Facade Design Pattern helps to improve readability and usability of a software library through masking interactions with complex components behind an API. It provides context-specific interfaces to more generic functionalities. It also helps to make our code more loosely-coupled. In our project, we noticed several User Interfaces where the Facade Design Pattern could be implemented.

5.1.1.1 Façade Problem 1

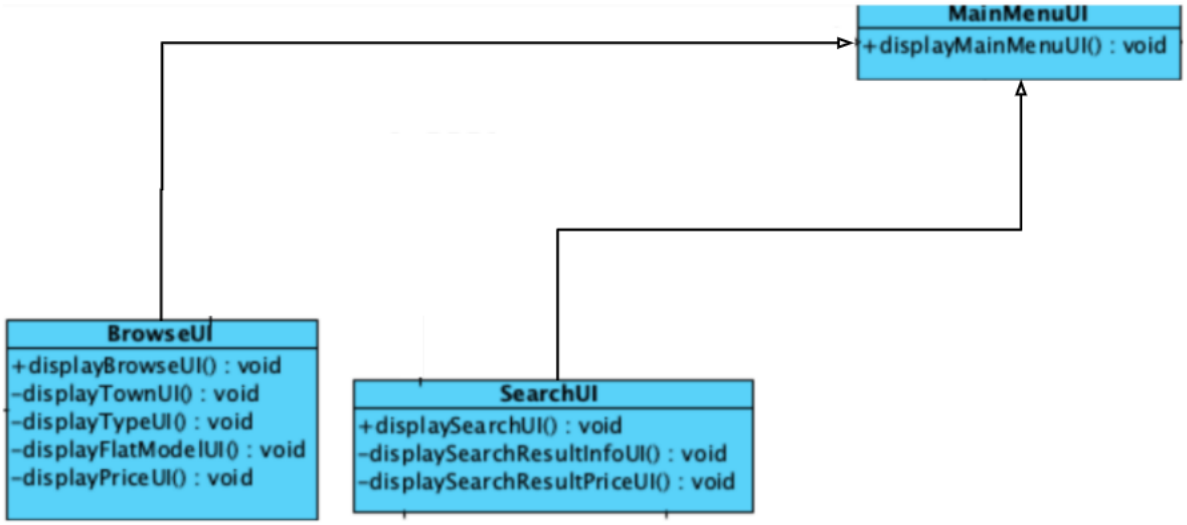

Figure 13: Facade Problem 1 - Before

The figure above shows MainMenuUI displaying both BrowseUI and SearchUI. A facade class, BrowseSearchUI, was introduced to improve the code, as shown below.

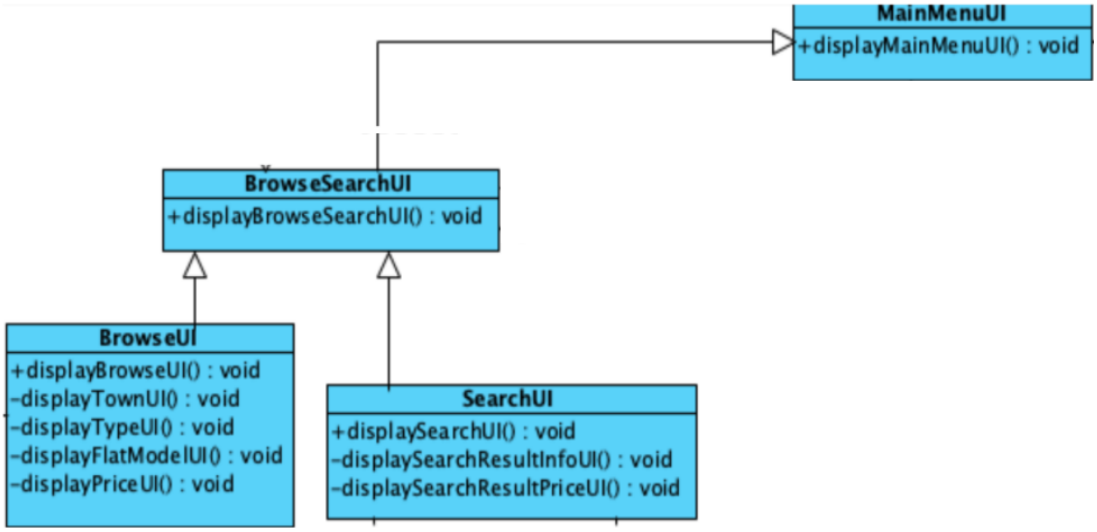

Figure 14: Facade Problem 1 - After

5.1.1.2 Façade Problem 2

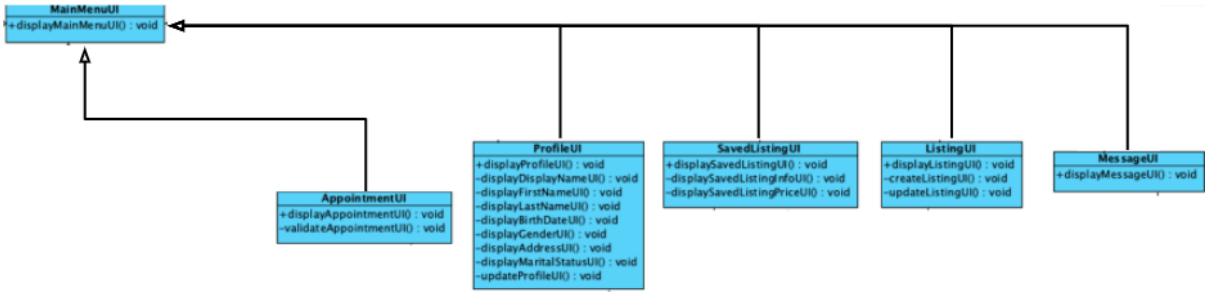

Figure 15: Facade Problem 2 - Before

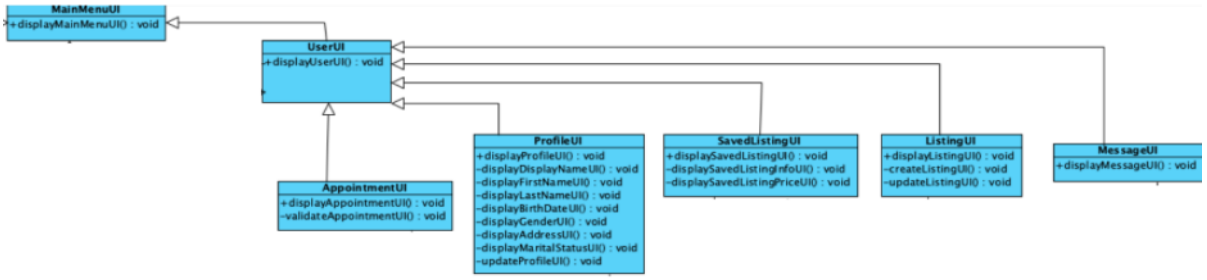

Figure 16: Facade Problem 2 - After

The same concept is applied in the second example.

5.1.2 MVC Architecture

The usage of Model-View-Controller (MVC) architecture facilitates and optimises the implementation of interactive intensive systems. With reference to Figure 10 (Class Diagram) and Figure 11 (Layer Architecture), the separation of the Model (in this case, Entity classes) from View (in this case, Boundary classes) and Controller (in this case, Control classes) components allow multiple views of the same model.

This separates presentation (Boundary classes) and interaction (Control classes) from the data. This makes it easy to support multiple different user interfaces or changes in interface details, and ensures that the core functionality is reusable across all different interfaces.

MVC architecture enables high cohesion as it allows for logical grouping of related actions on a controller together and grouping of views for specific models. It also ensures low coupling among models, views or controllers.

5.1.3 Single Responsibility Principle

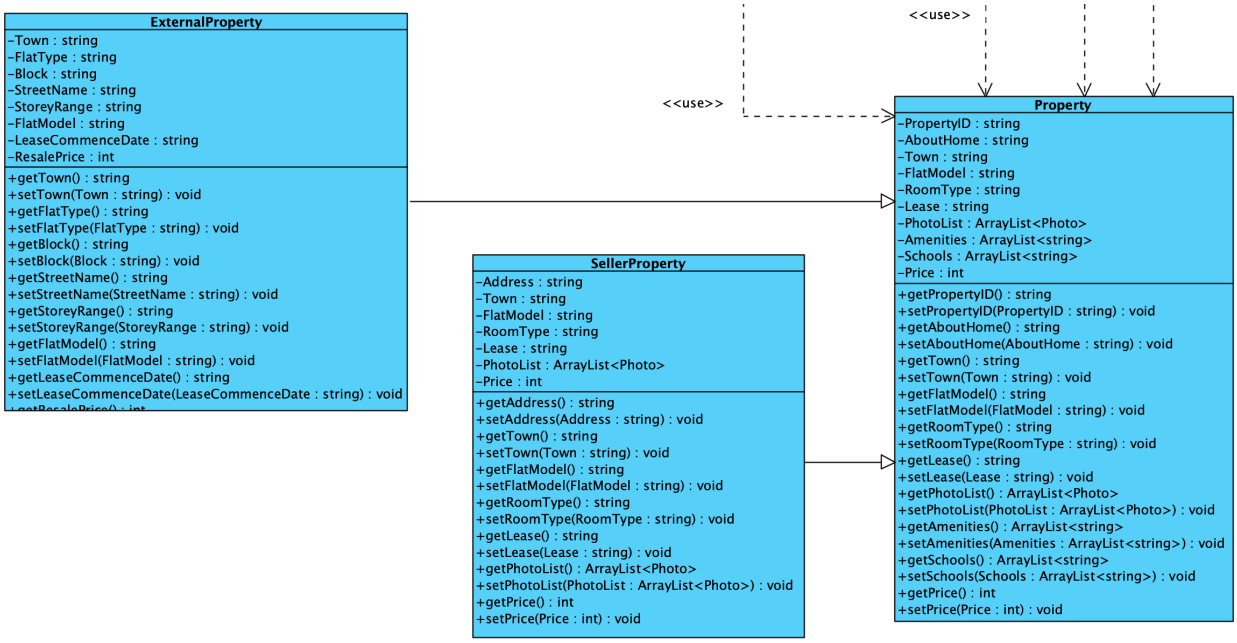

Figure 17: Extracted part from Class Diagram

With reference to the figure above, in our storage of data, we split the entity class for Property into ExternalProperty and SellerProperty. This is in line with the Single Responsibility Principle (SRP), which states that each class should only assume one responsibility. In this case, the data from ExternalProperty is retrieved from the government data API, while the SellerProperty data is obtained from the users. Originally, we only had one generic class for Property, but we eventually decided to split it up so that the external and internal data will have their own class that assumes their own responsibility, and so that a modification in one of them will not affect the other. This will also support further extensions if other forms of data are added to the system.

5.2 Software Engineering Practices

5.2.1 GitHub Branch Management

GitHub's branching management helped our team manage our workflow. Our team placed our project on GitHub and worked on it from there. When one team member creates a branch in our GitHub project, changes made on that branch does not affect the main branch, and that branch will only be merged when it is reviewed by other team members in the same collaboration.

Any updates made by team members are added as commits, which helps us keep track of our progress and ensure that we are on track with our planned schedule. Commits also create a transparent history of everyone's work that other team members can follow and keep up with, and allow us to revert changes if needed. After going through pull requests and reviews, commits can then be merged into the main branch.
Our team made use of GitHub's branch management system to aid us in collaborating in a systematic manner, and to ensure that everything is documented properly.

GitHub Repository Link: https://github.com/khinwayway/710-SE

5.2.2 Gantt Chart

Our team made use of a Gantt Chart to set up a timeline for our project, and to document our progress. This helped us stay on track and ensured that we generally follow the timeline and not let our progress lag too far behind the proposed one.
During the completion of our project, we filled up the Gantt Chart with the actual timeline, which helped to document our progress and once again allow us to keep ourselves in check.
The Gantt Chart can be found in the Appendix.

6 UI Mock-ups

6.1 Buy/Rent

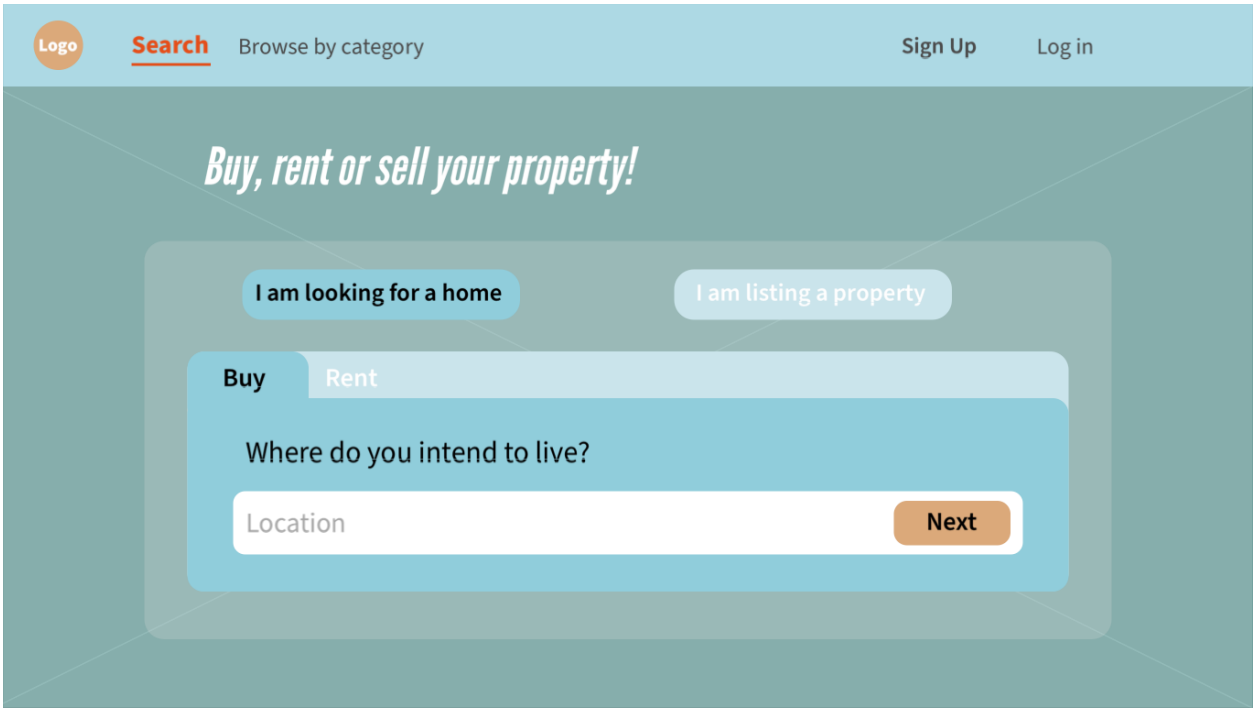

Figure 18: Buy/Rent UI Mock-up

6.2 Sell/Rent Out

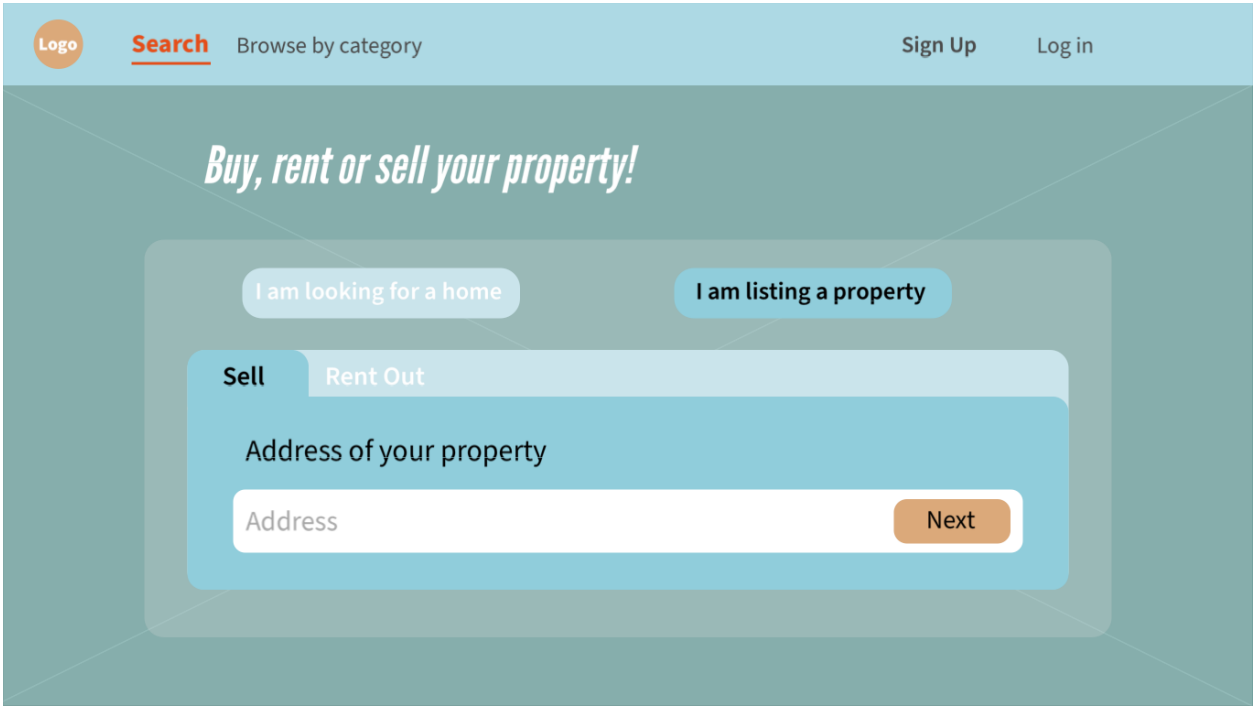

Figure 19: Sell/Rent Out UI Mock-up

6.3 Login

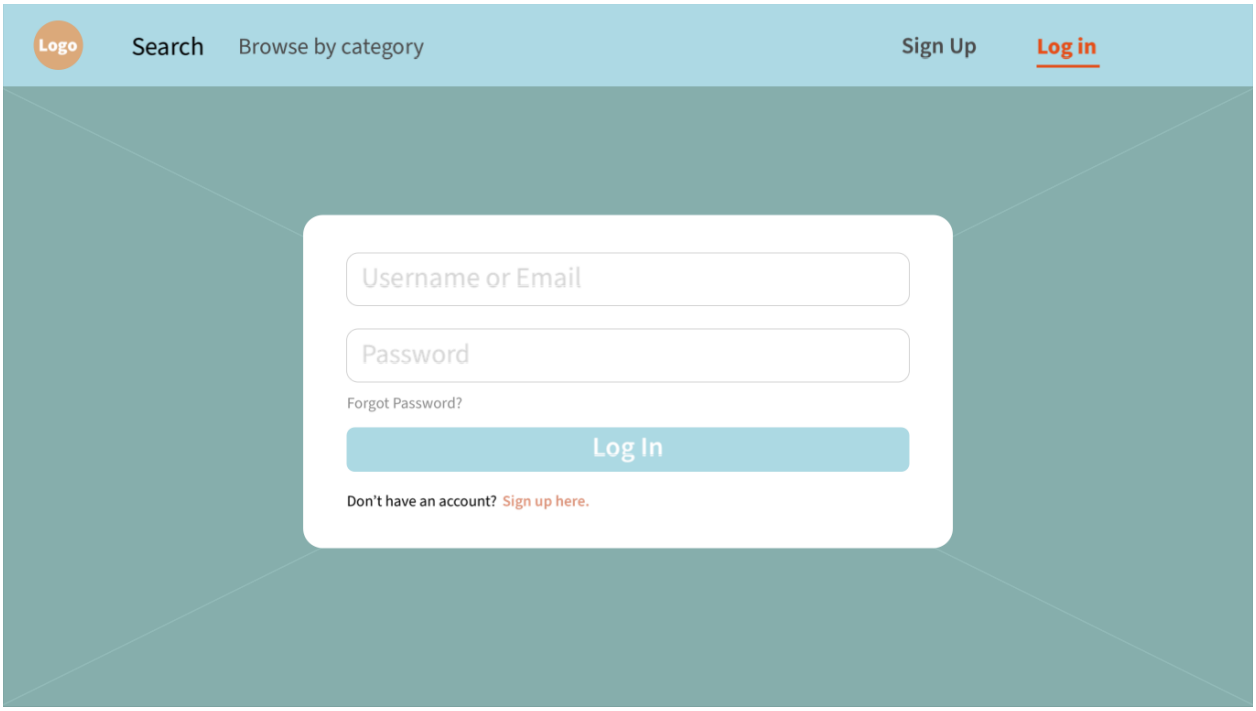

Figure 20: Login UI Mock-up

6.4 Sign Up

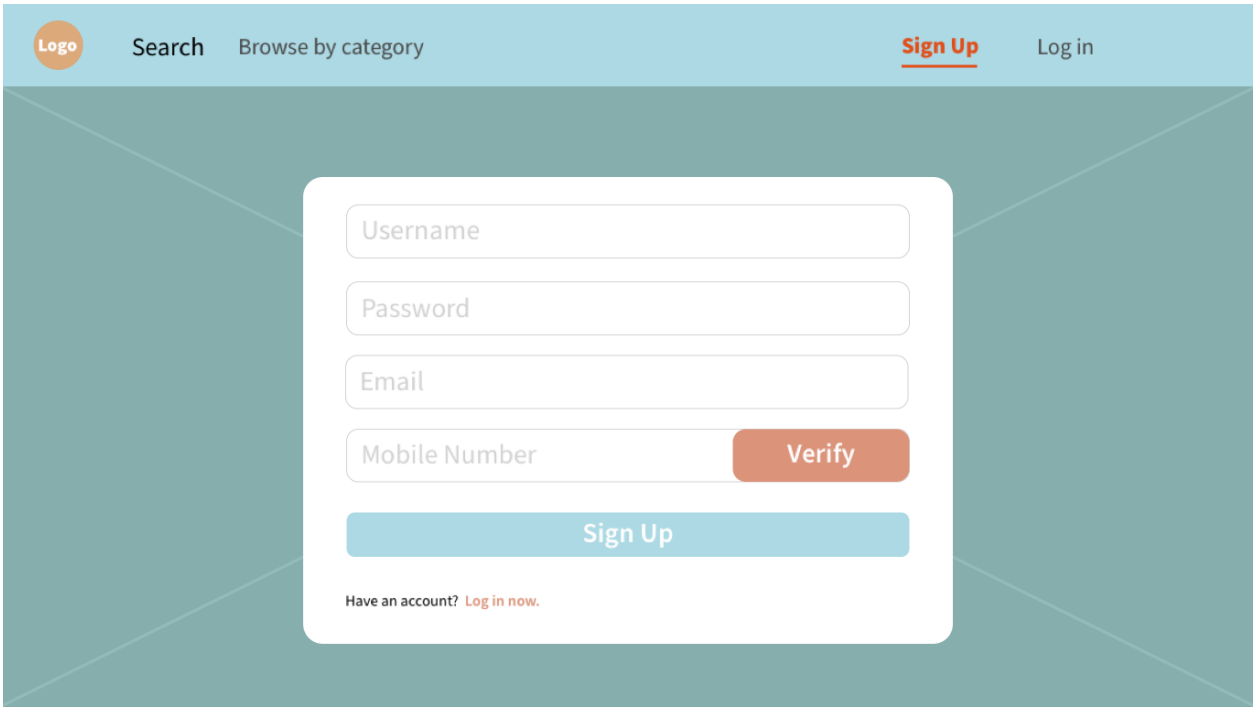

Figure 21: Sign Up UI Mock-up

6.5 Search

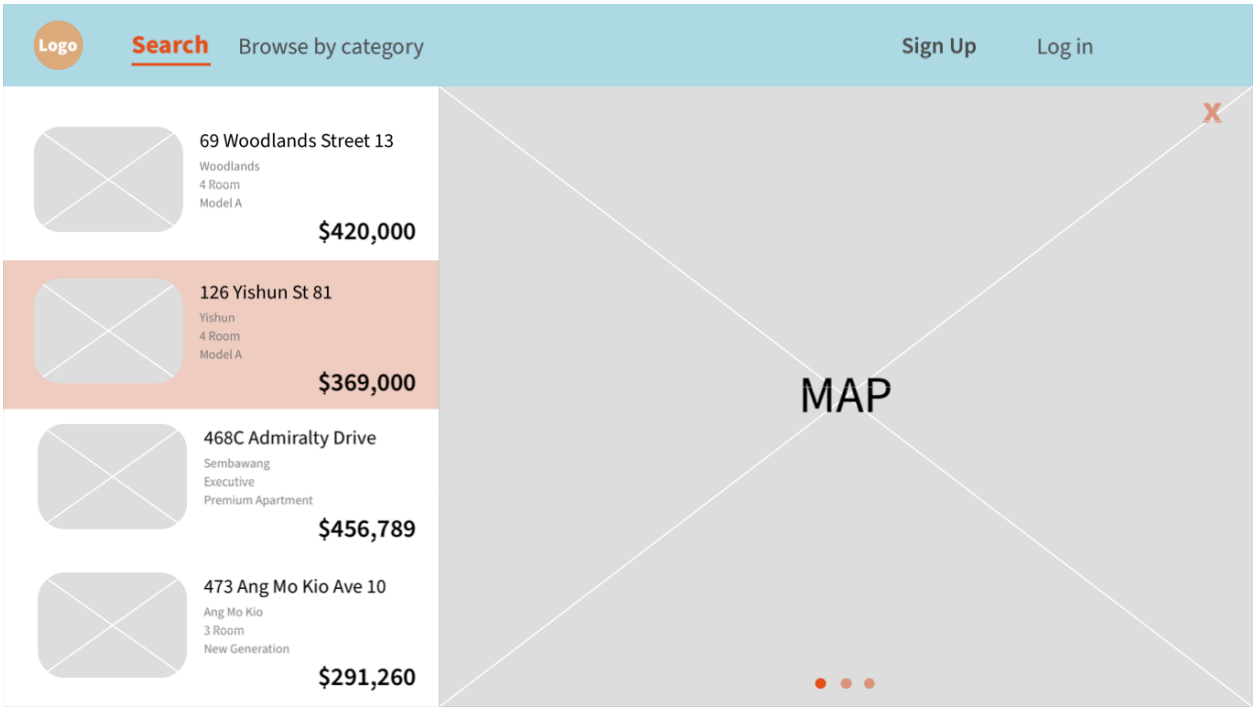

Figure 22: Search UI Mock-up 1

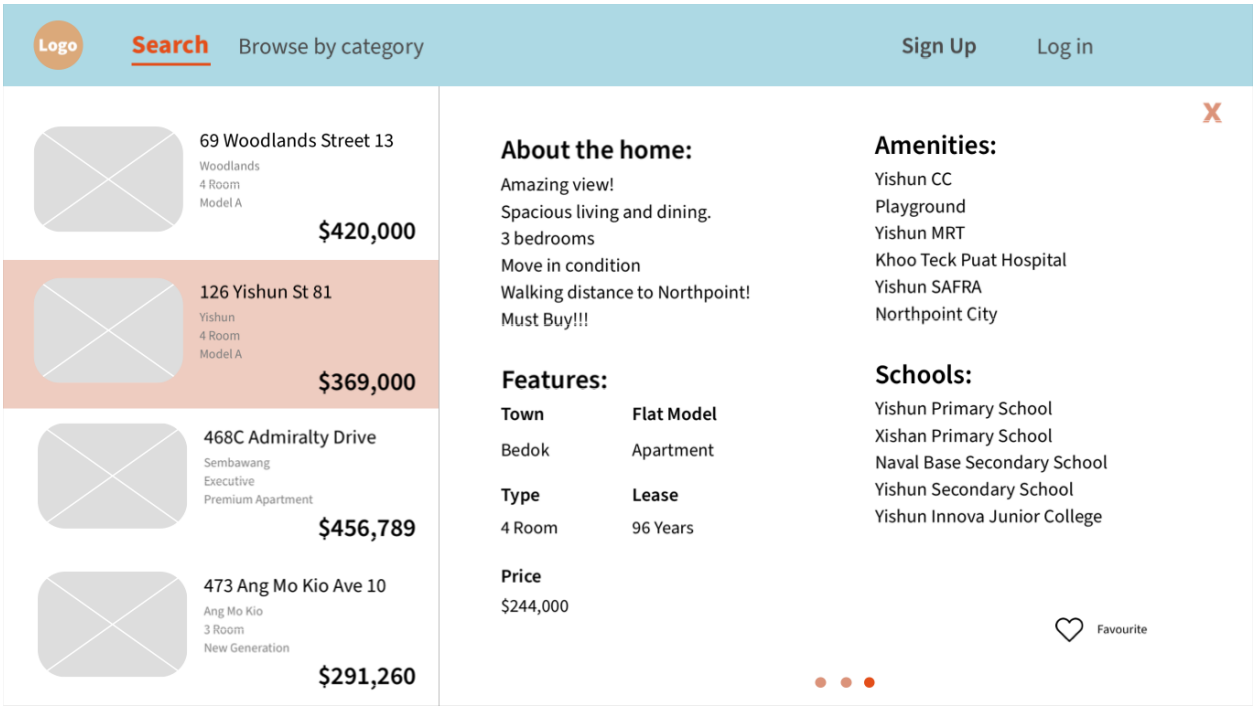

Figure 23: Search UI Mock-up 2

6.6 Browse

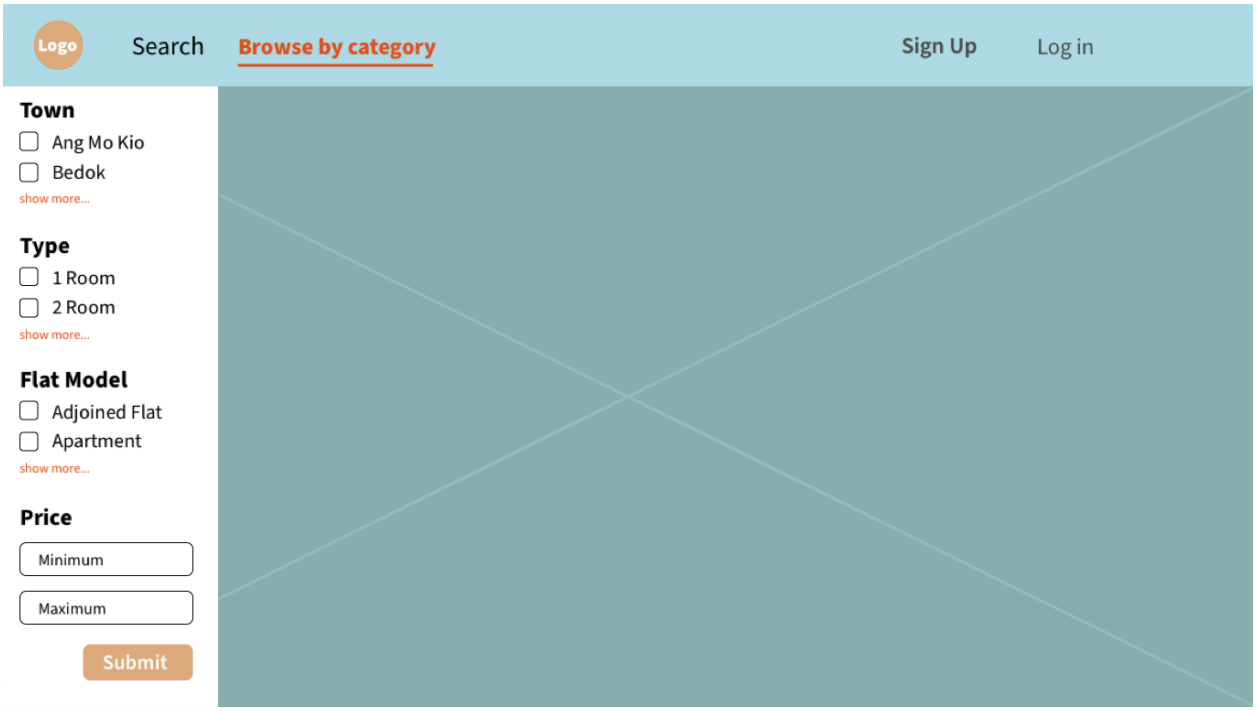

Figure 24: Browse UI Mock-up 1

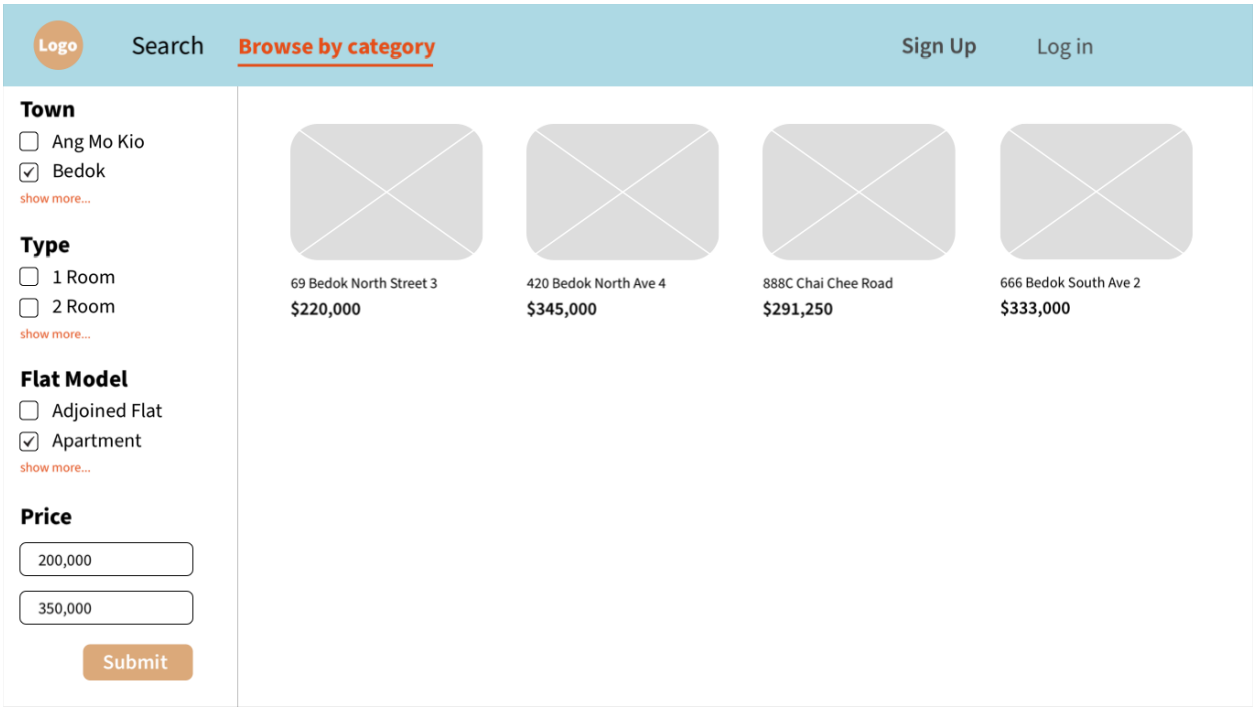

Figure 25: Browse UI Mock-up 2

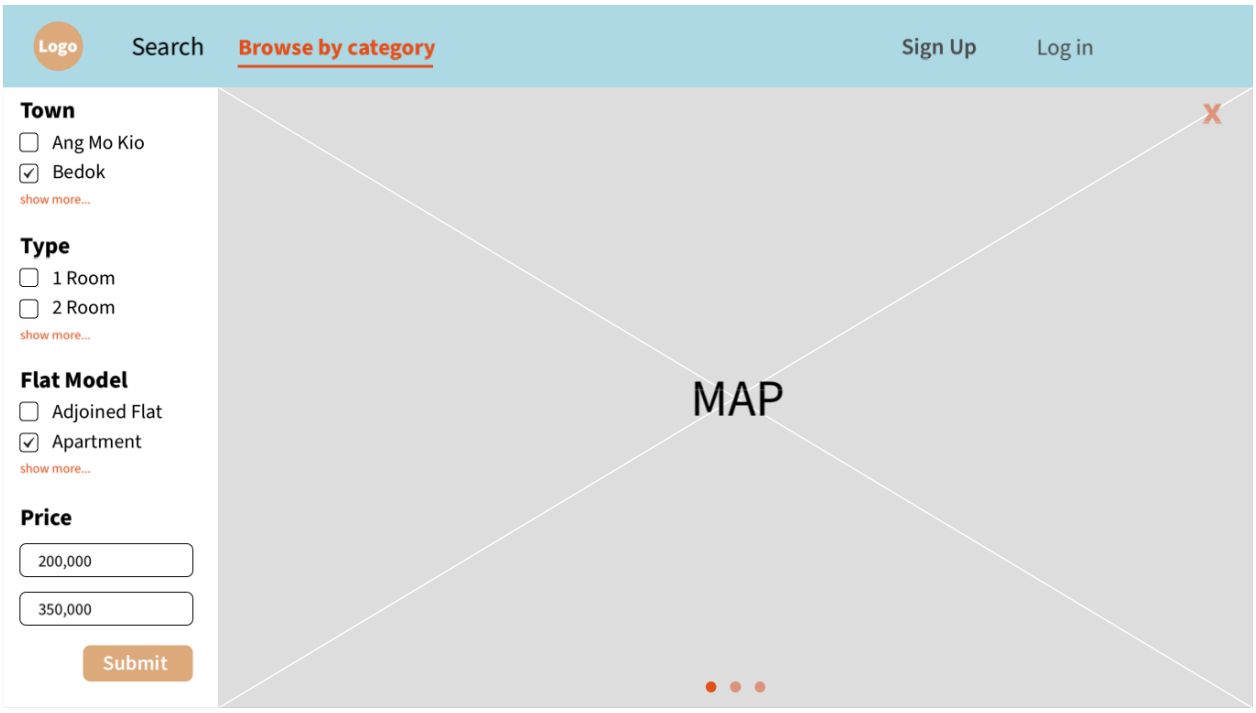

Figure 26: Browse UI Mock-up 3

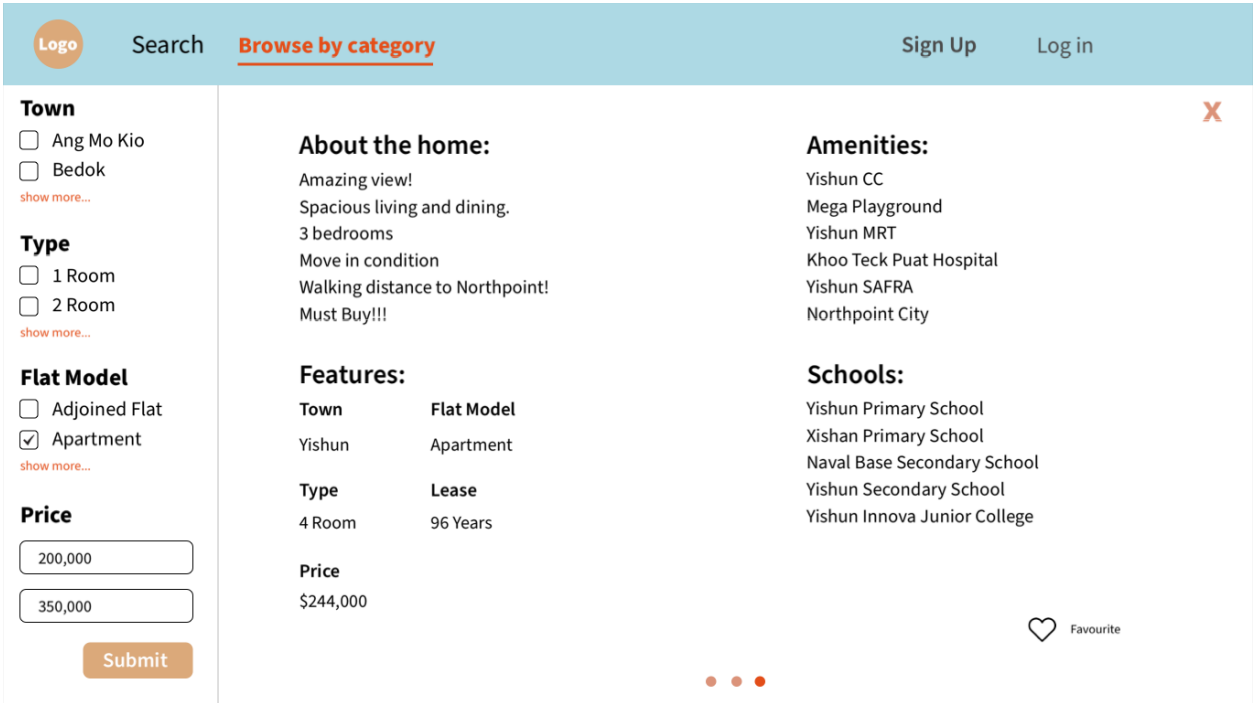

Figure 27: Browse UI Mock-up 4

6.7 Profile

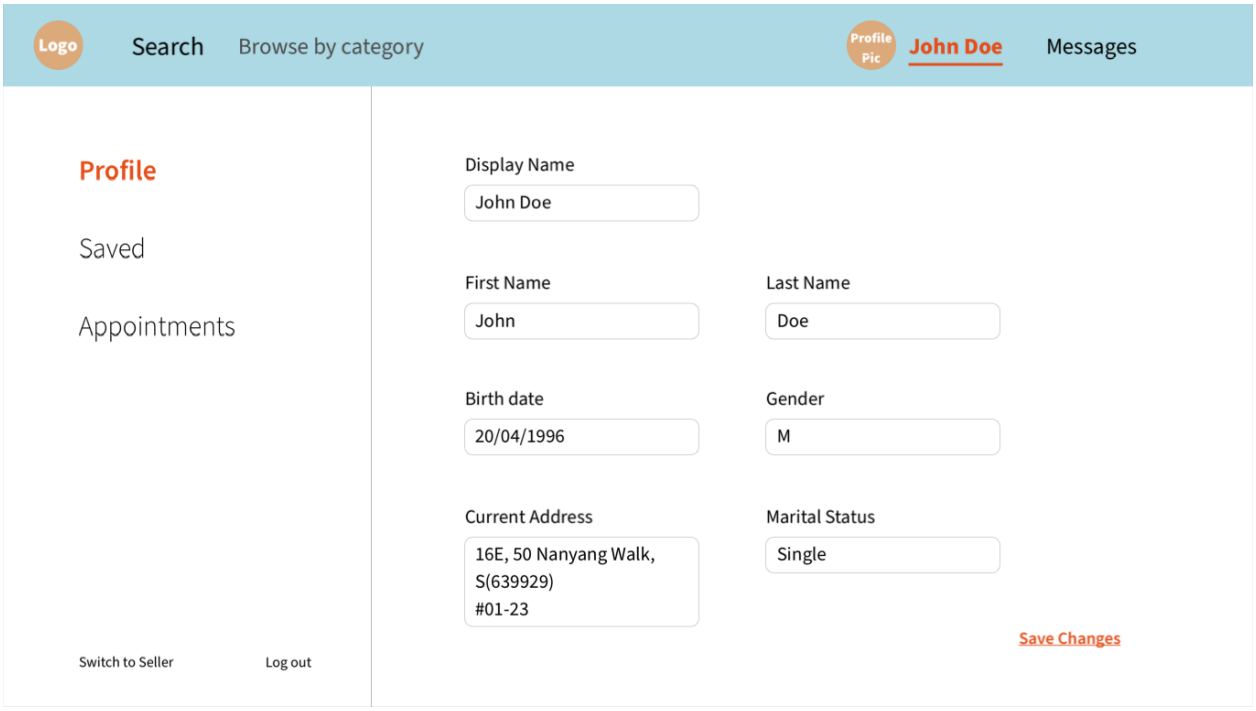

Figure 28: Profile UI Mock-up

6.8 Seller Listings

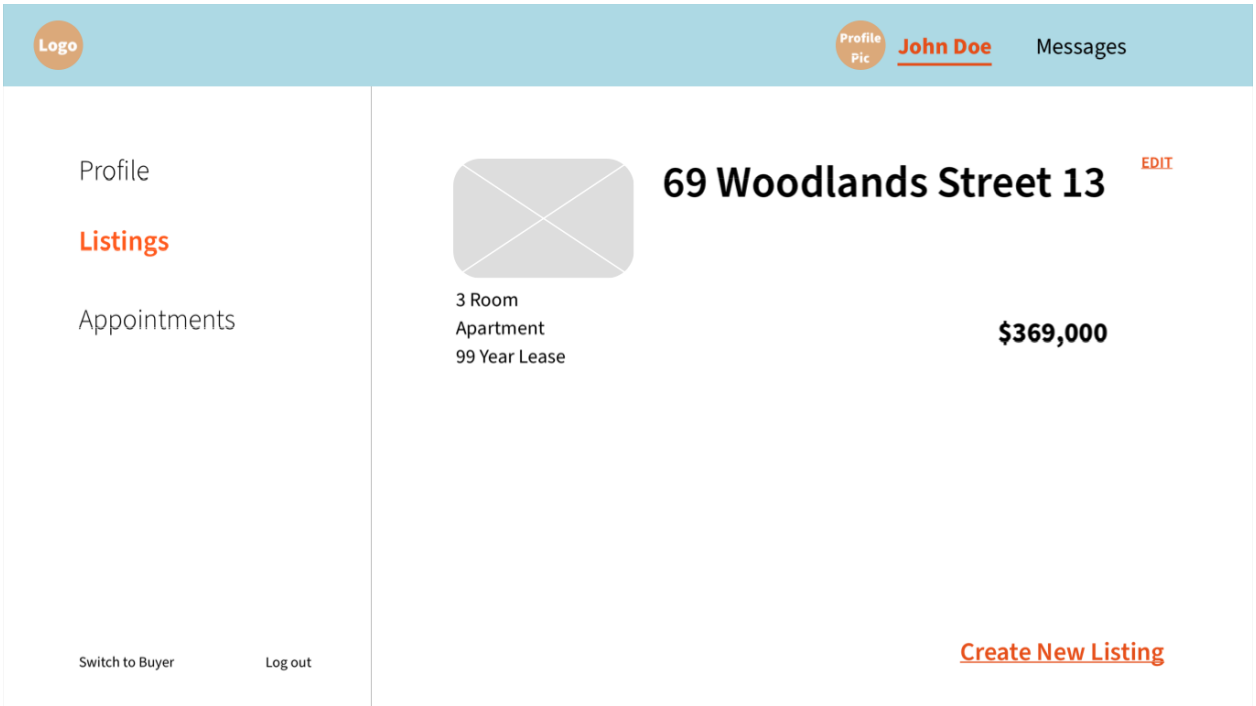

Figure 29: Seller Listing UI Mock-up

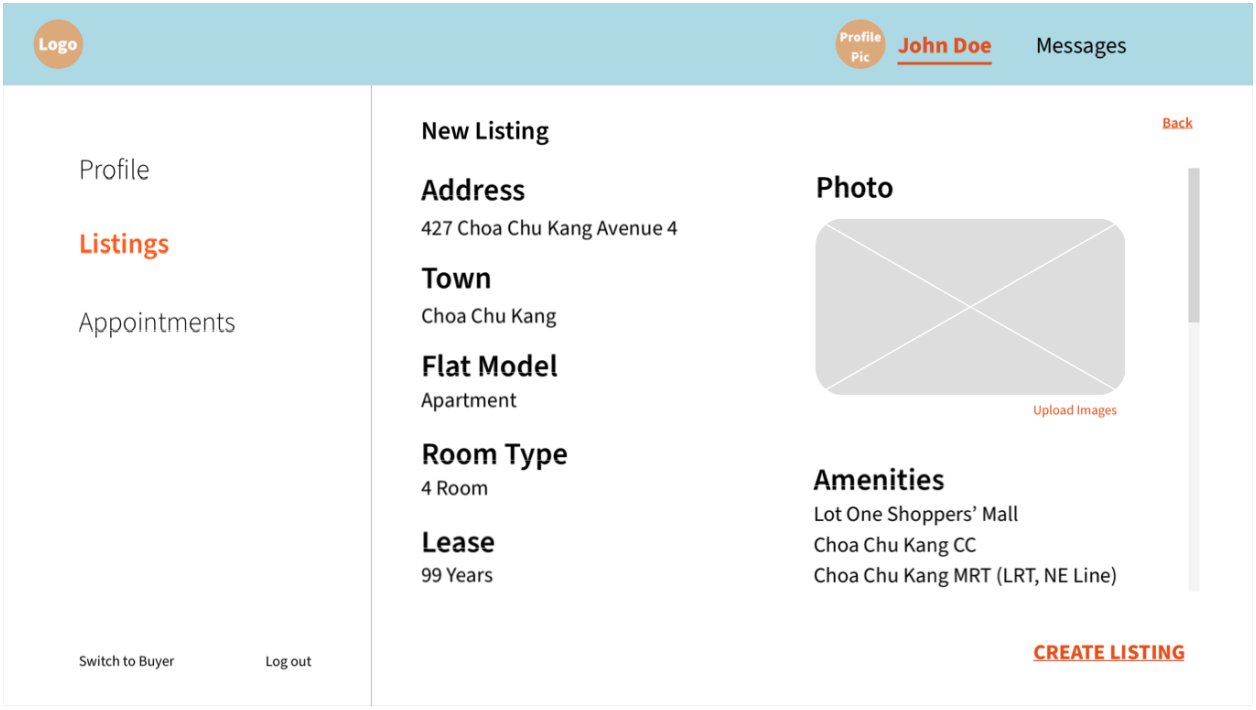

Figure 30: Create New Listing UI Mock-up

7 Testing

7.1 Blackbox Testing

7.1.1 Login

7.1.1.1 Generic Cases

Test ID	Scenario	Expected Result	Actual Result
1	Login with valid account username and password	The system displays the main menu for user to continue the operation	The system displays the main menu for user to continue the operation
2	Login without valid credentials	The system prompts the user to enter valid credentials	The system prompts the user to enter valid credentials
3	Login without filling up the required fields	The system prompts the user to enter required fields	The system prompts the user to enter required fields

7.1.1.2 Specific Cases

Username	Password	Expected Result	Actual Result
Testuser	Testpassword	Successful login	Successful login
Wronguser	Testpassword	System prompt: “Please Enter Correct Username and Password”	System prompt: “Please Enter Correct Username and Password”
Testuser	Wrongpassword	System prompt: “Please Enter Correct Username and Password”	System prompt: “Please Enter Correct Username and Password”
“” (Empty)	Testpassword	System prompt: “Please fill in this field”	System prompt: “Please fill in this field”
Testuser	“” (Empty)	System prompt: “Please fill in this field”	System prompt: “Please fill in this field”

7.1.2 Sign Up

7.1.2.1 Generic Cases

Test ID	Scenario	Expected Result	Actual Result
1	Signup with valid account username and password	The system creates an account with valid username and password	The system creates an account with valid username and password
2	Login without filling up the required fields	The system prompts the user to enter required fields	The system prompts the user to enter required fields

7.1.1.2 Specific Cases

Username	Password	Email	Mobile Number	Expected Result	Actual Result
Testuser	Testpassword	Testemail	Testmobilenumber	Successful sign up	Successful sign up
“” (Empty)	Testpassword	Testemail	Testmobilenumber	System prompt: “Please enter an username”	System prompt: “Please enter an username”
Testuser	“” (Empty)	Testemail	Testmobilenumber	System prompt: “Please enter a password”	System prompt: “Please enter a password”
Testuser	Testpassword	“” (Empty)	Testmobilenumber	System prompt: “Please enter an email”	System prompt: “Please enter an email”
Testuser	Testpassword	Testemail	“” (Empty)	System prompt: “Please enter a mobile number”	System prompt: “Please enter a mobile number”

7.1.3 Browse

Test ID	Scenario	Expected Result	Actual Result
1	Browse properties with no user input made (no selections)	The system displays all properties available for sale	The system displays all properties available for sale
2	Browse properties with at least one user input (e.g. town/type/flat model etc.)	The system displays properties available that fit within user’s input. The system prompts the user if no properties are found	The system displays properties available that fit within user’s input. The system prompts the user if no properties are found

7.1.4 Search

Test ID	Scenario	Expected Result	Actual Result
1	Search properties with no user input made	The system prompts user to make a valid input	The system displays all properties available for sale
2	Search properties with user input made (e.g. Woodlands/Choa Chu Kang)	The system displays properties available that fit within user's input. The system prompts the user if no properties are found	The system displays properties available that fit within user's input. The system prompts the user if no properties are found

7.1.5 Profile

Test ID	Scenario	Expected Result	Actual Result
1	Update profile with valid inputs	The system updates profile based on inputs	The system updates profile based on inputs
2	Update profile without filling up the required fields	The system prompts the user to enter required fields	The system prompts the user to enter required fields

7.1.6 Create New Listing

7.1.6.1 Generic Cases

Test ID	Scenario	Expected Result	Actual Result
1	Create new listing with valid inputs	The system creates a new listing based on inputs	The system creates a new listing based on inputs
2	Create new listing without filling up the required fields	The system prompts the user to enter required fields	The system prompts the user to enter required fields

7.1.6.2 Specific Cases

All fields (Block, Town, Address, Flat Model, Room Type, Remaining Lease, About)	Expected Result	Actual Result
All fields filled with valid inputs	Successful creation of new listing with inputs	Successful creation of new listing with inputs
All fields filled except About (About is optional)	Successful creation of new listing with inputs	Successful creation of new listing with inputs
One or more field (except for About) empty	System prompt: "Please fill in this field"	System prompt: "Please fill in this field"

7.2 Whitebox Testing

7.2.1 Login

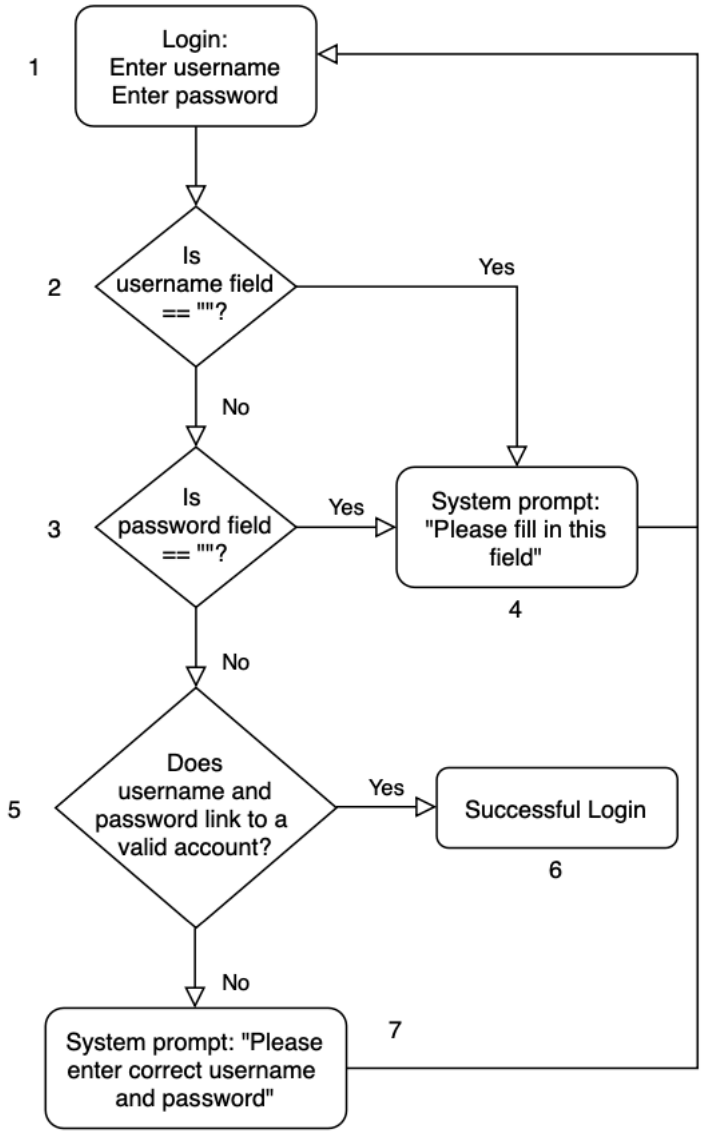

Figure 31: Login Whitebox Testing

Cyclomatic Complexity: 4

Basis Path	Expected Result	Actual Result
1,2,3,5,6	Successful login	Successful login
1,2,4,1	System prompt: “Please fill in this field”	System prompt: “Please fill in this field”
1,2,3,4,1	System prompt: “Please fill in this field”	System prompt: “Please fill in this field”
1,2,3,5,7,1	System prompt: “Please enter correct username and password”	System prompt: “Please enter correct username and password”

7.2.2 Sign Up

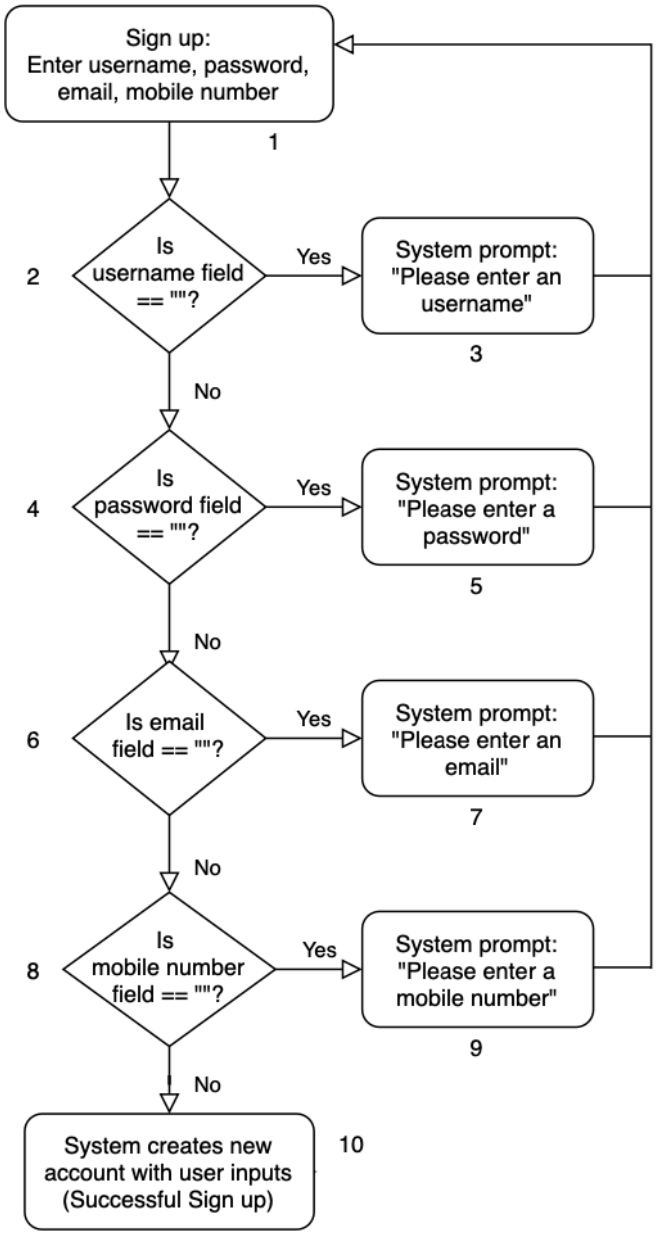

Figure 32: Sign Up Whitebox Testing

Cyclomatic Complexity: 5

Basis Path	**Expected Result**	**Actual Result**
1,2,4,6,8,10	Successful Sign up	Successful Sign up
1,2,3,1	System prompt: “Please enter an username”	System prompt: “Please enter an username”
1,2,4,5,1	System prompt: “Please enter a password”	System prompt: “Please enter a password”
1,2,4,6,7,1	System prompt: “Please enter an email”	System prompt: “Please enter an email”
1,2,4,6,8,9,1	System prompt: “Please enter a mobile number”	System prompt: “Please enter a mobile number”

7.2.3 Create New Listing

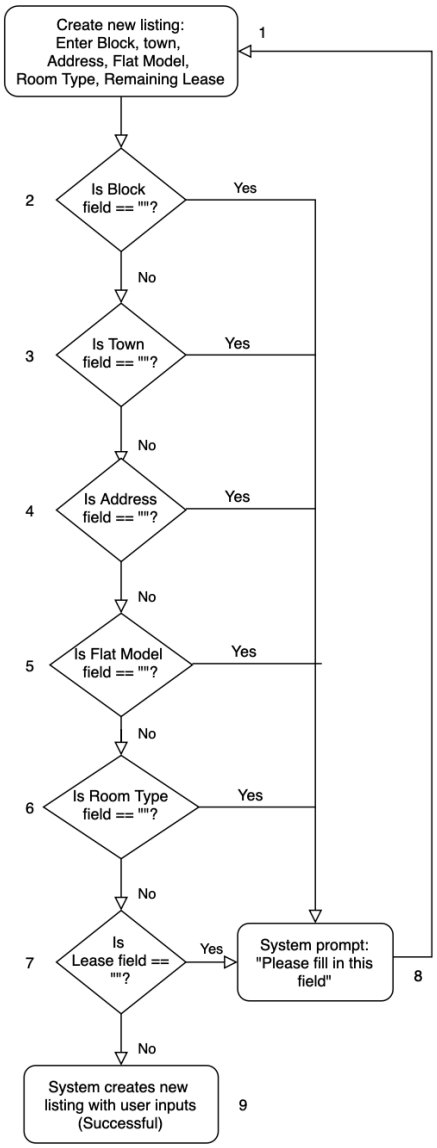

Figure 33: Create New Listing Whitebox Testing

Cyclomatic Complexity: 7

Basis Path	**Expected Result**	**Actual Result**
1,2,3,4,5,6,7,9	Successfully create new listing	Successfully create new listing
1,2,8,1	System prompt: “Please fill in this field”	System prompt: “Please fill in this field”
1,2,3,8,1	System prompt: “Please fill in this field”	System prompt: “Please fill in this field”
1,2,3,4,8,1	System prompt: “Please fill in this field”	System prompt: “Please fill in this field”
1,2,3,4,5,8,1	System prompt: “Please fill in this field”	System prompt: “Please fill in this field”
1,2,3,4,5,6,8,1	System prompt: “Please fill in this field”	System prompt: “Please fill in this field”
1,2,3,4,5,6,7,8,1	System prompt: “Please fill in this field”	System prompt: “Please fill in this field”

8 Data Dictionary

Term	Description	Attributes	Relationships with other terms
Guest	Anyone browsing without an account.		A guest may sign up or log in to be a user.
User	A user is someone browsing with an account.	Account	A user owns an account. Users with accounts can make appointments. A user with an account can list properties.
Account	A user's account allows a user to authenticate to a system. Each account has a unique username and a case-sensitive password. Each account is connected to an email address and a verified phone number. Users may upload and modify their profile.	Username:string (6-20 characters), Password:string (>8 characters, with at least one numeric character), Email:string (to be verified), PhoneNumber:string (8 numeric characters, to be verified), Profile	A user owns an account. Account has profile
Profile	Basic information provided by the user.	Avatar, DisplayName:string (1-40 characters), FirstName:string (1-20 characters), LastName:string (1-20 characters), BirthDate:string (8 characters, in format of DDMMYYYY),	Account has profile

		Gender:string (only “F” or “M”), CurrentAddress:string(1-512 characters), MaritalStatus:string (one of the options provided to them e.g. Single, Married, Divorced, etc.)	
Property	A property listed by a seller/landlord.	Address:string(1-512 characters), About the home:string(0-1000 characters), Amenities:string(0-512 characters) , PropertyID:int Town:string Type:string(one of the options provided to them e.g. 1-room, 2-room etc.) Flat Model:string(one of the options provided to them e.g. Model A, New Generation etc.) Lease:string Price:int(0-9,999,999) Amenities:string(0-512 characters) Schools:string(0-512 characters)	A user can list properties

About the home	Description of the house, to be completed by seller/landlord.		Attribute of property
Town	The town council a property belongs to. E.g. Woodlands, Yishun		Attribute of property
Type	The unit type of a house. E.g. 1-room, 2-room		Attribute of property
Flat Model	The flat type of a house.		Attribute of property
Lease	The effective period of the contract offered by the land owner.		Attribute of property
Price	Price offered by the land owner. The unit of price is Singapore Dollars (SGD). It should be an integer from 0 to 9,999,999		Attribute of property
Amenities	Infrastructure and facilities within a convenient range from the listed property.		Attribute of property
Schools	Educational institutes within a convenient range from the listed property.		Attribute of property
Appointment	A calendar activity made by the agreement of two users.	AppointmentID:int User Location:string DateTime:DateTime	Users can make appointments.

9 Appendix

For a clearer version of the figures, do refer to folder “CE2006_710_FinalReport Diagrams”.

9.1 Gantt Chart

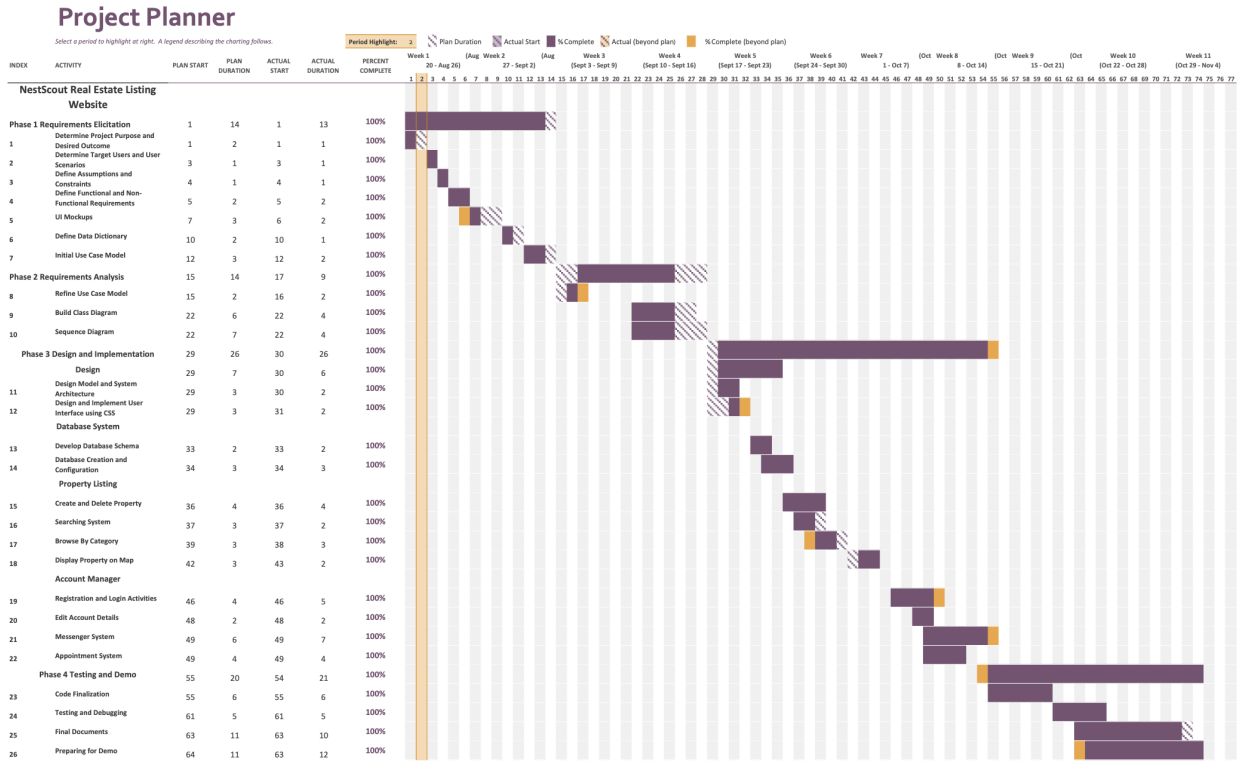

Figure 34: Gantt Chart

Note: The original Gantt Chart Excel file is attached in the folder “CE2006_710_FinalReport Diagrams”. The yellow highlighted column serves as a “period highlight”, whereby its value can be changed to highlight corresponding columns for easier viewing. For example, in the figure above, Period 2 is highlighted as the value is 2.