Software Requirements Specification

for

CZ2006 PropNow

Version 1.0 approved

Prepared by

Team Haha (DS1) CZ2006 Course of Nanyang Technological University

Date created: 17/04/21

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed description about the requirements needed to successfully complete the PropNow Android mobile application for the benefit of the project management, client, and development team. This is version 1 of the document, and this software requirements specification describes the entire PropNow Android mobile application. This document reports the requirements based on the official meetings of the members of Haha from CZ2006 Lab Group DS1. This document will cover the purpose and complete declaration for the development of the system.

1.2 Document Conventions

This document follows the IEEE SRS formatting requirements. This document uses the following conventions:

1. Agents: These are the users who are registered into the system as agents and refer to agents who aid in selling houses in Singapore. They have their own house listings and they can make changes like adding or deleting their house listings.
2. Non-agents: These are the users who are registered into the system as a non-agent to browse for houses available for sale in Singapore and to access features such as home calculator and view eligibility for grants.
3. General users: These are the users which are not registered into the system i.e they don't have their own personal account. They will have limitations on accessible functions.

1.3 Intended Audience and Reading Suggestions

This project is a prototype for a housing rental and selling management system integrated within the PropNow Android mobile application. This has been implemented under the guidance of the CZ2006 Software Engineering course in Nanyang Technological University. This document is intended for developers, project managers, marketing staff, users, testers, and documentation writers interested in implementing or using a housing rental and selling management system.

Throughout this document, this document will provide a general description of the project, the external interface requirements, system features, non functional requirements, and additional features of the PropNow Android mobile application. We suggest that this document be read in detail with the order specified in the table of contents (pg. 2) for developers, project managers, documentation writers, and marketing staff. Readers such as users are suggested to view Section 4 in order to view the system features of this application. Readers such as testers are suggested to view Section 4 and Section 5 in order to view the system features and nonfunctional requirements of this application.

1.4 Product Scope

The purpose of this product is to create a functional application for the general public to make an informed decision on the purchase of houses in Singapore, as well as providing a platform to aid in the purchase process. The system is based on a relational database as it has a unique key which allows access to another piece of data in the database. Our database includes the information of every non-agent, agent and the house listing owned by them.

More specifically, this system is designed to allow the non-agents to review houses available for sale as well as to communicate with the agents to get information about the houses that they are interested in purchasing. This app also has certain features like home calculator and view eligibility for grants which can assist non-agent users in making an educated choice while buying a house in Singapore.

1.5 References

1. Bandakkanavar, R., January 5. (2019, October 17). Software requirements specification document with example. Retrieved April 18, 2021, from https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database
2. M., A. (2018, December 21). Software requirement specification - building a successful application. Retrieved April 18, 2021, from https://magora-systems.com/software-requirement-specification/
3. Meet android studio : Android developers. (n.d.). Retrieved April 18, 2021, from https://developer.android.com/studio/intro/

2. Overall Description

2.1 Product Perspective

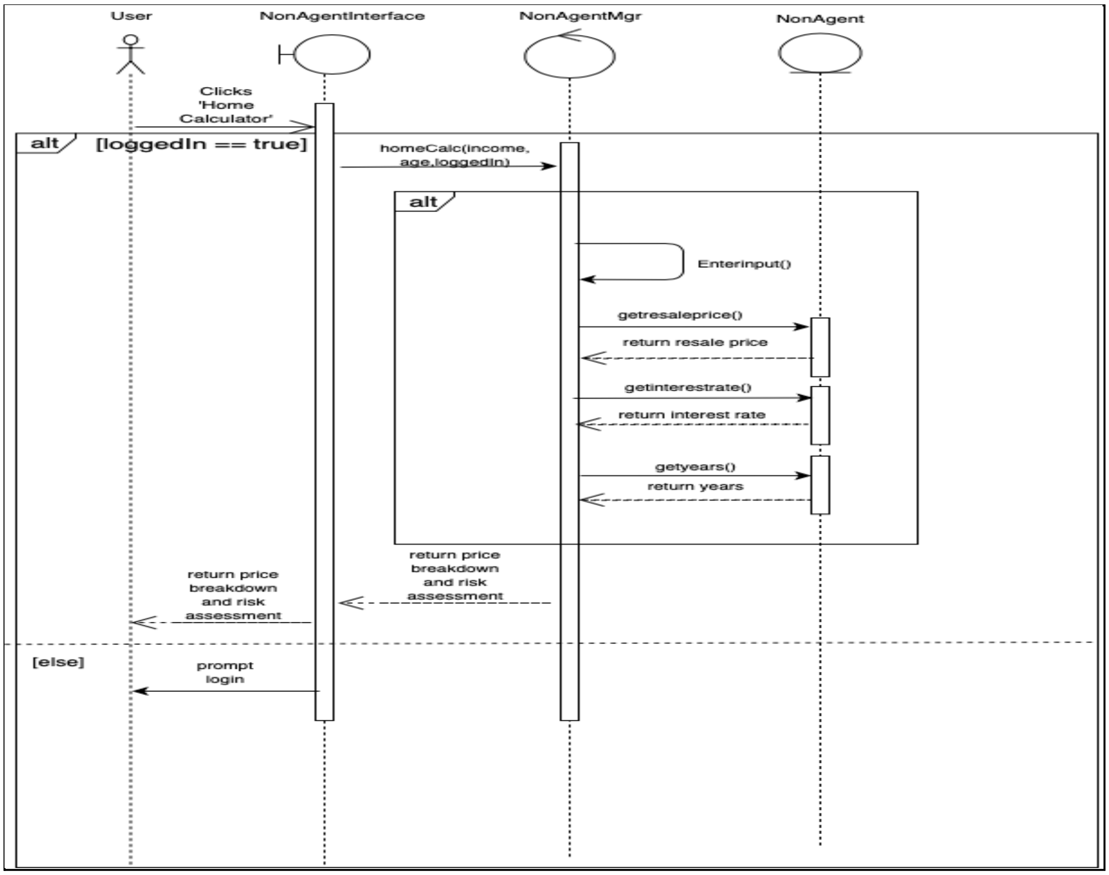
This product is a lifestyle mobile application. This product is an independent system and intends to serve as a platform to aid in the buying, selling, and renting process of houses in Singapore. Not only does it serve these purposes, it also allows users to calculate the total interest payable and the equated monthly installment according to the property price, interest rate, number of years added by the user. This product aims to provide a convenient platform by implementing a simple user interface allowing all types of users to gain a satisfying user experience.

2.2 Product Functions

Listed below are some of the major functions of our mobile application:

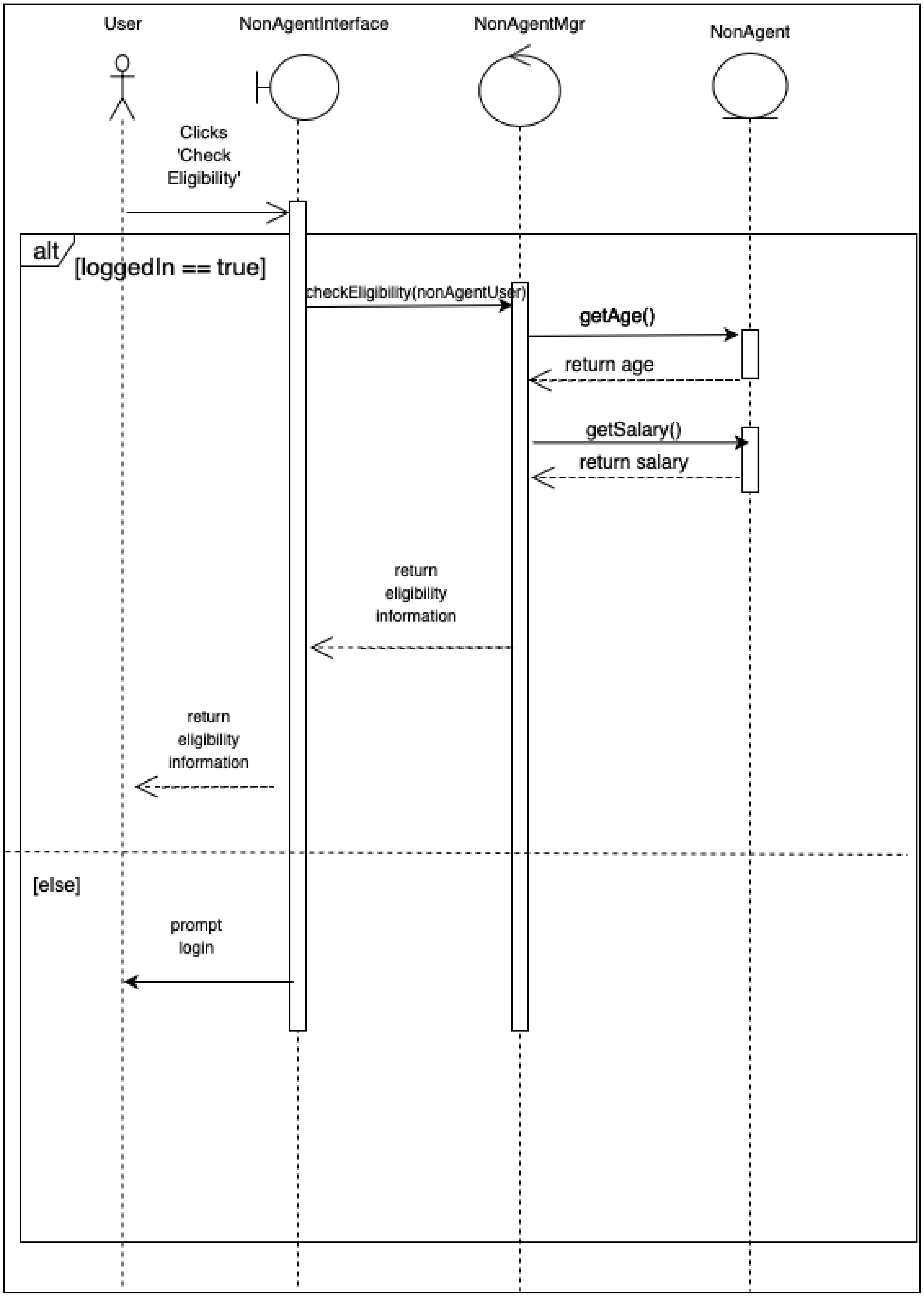
2.2.1 Home calculator

This function allows the user to calculate the equated monthly installment and the total interest payable on the property price.



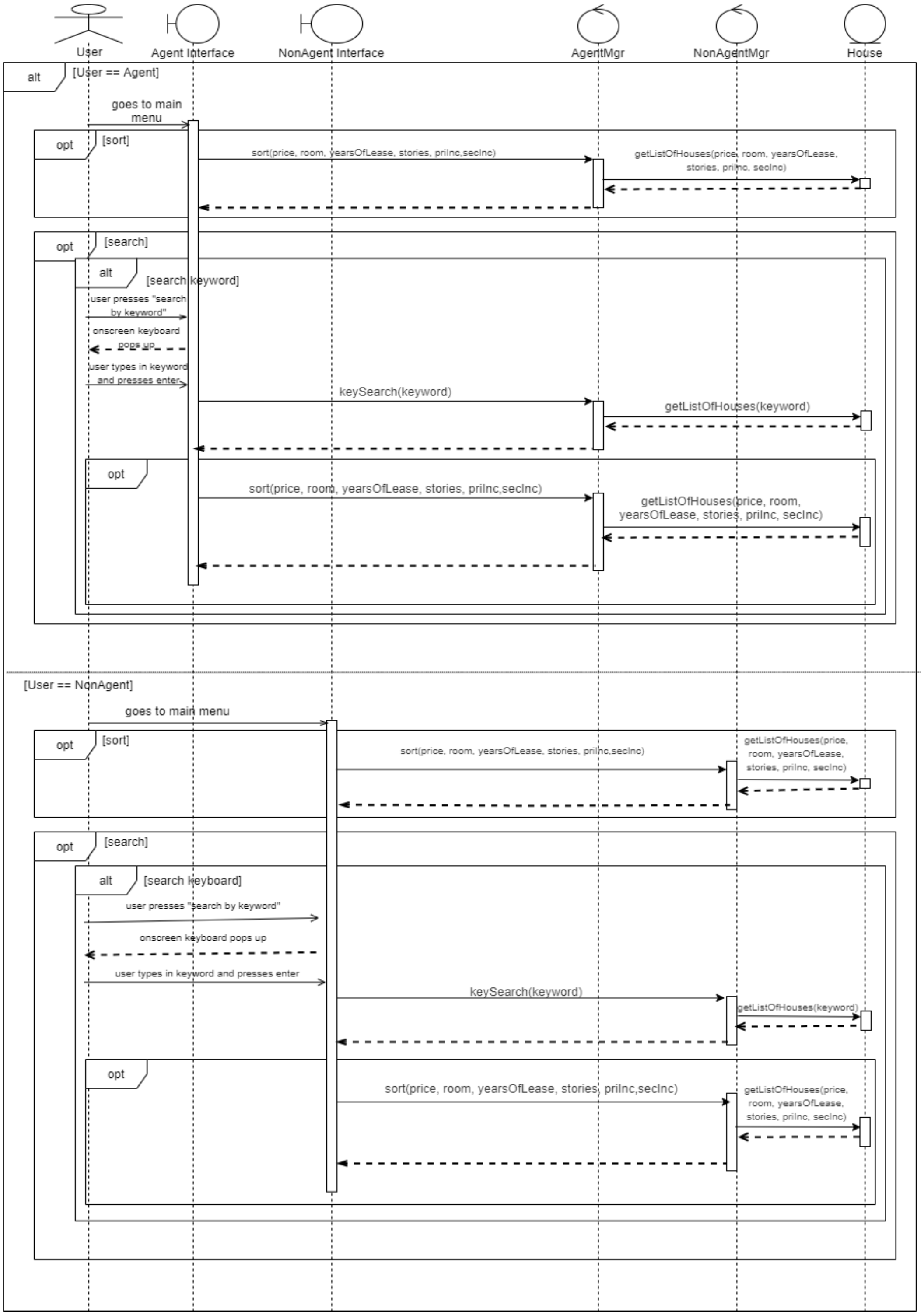
2.2.2 Check eligibility

This function allows the user to know which grants they are eligible for by looking at their monthly income.



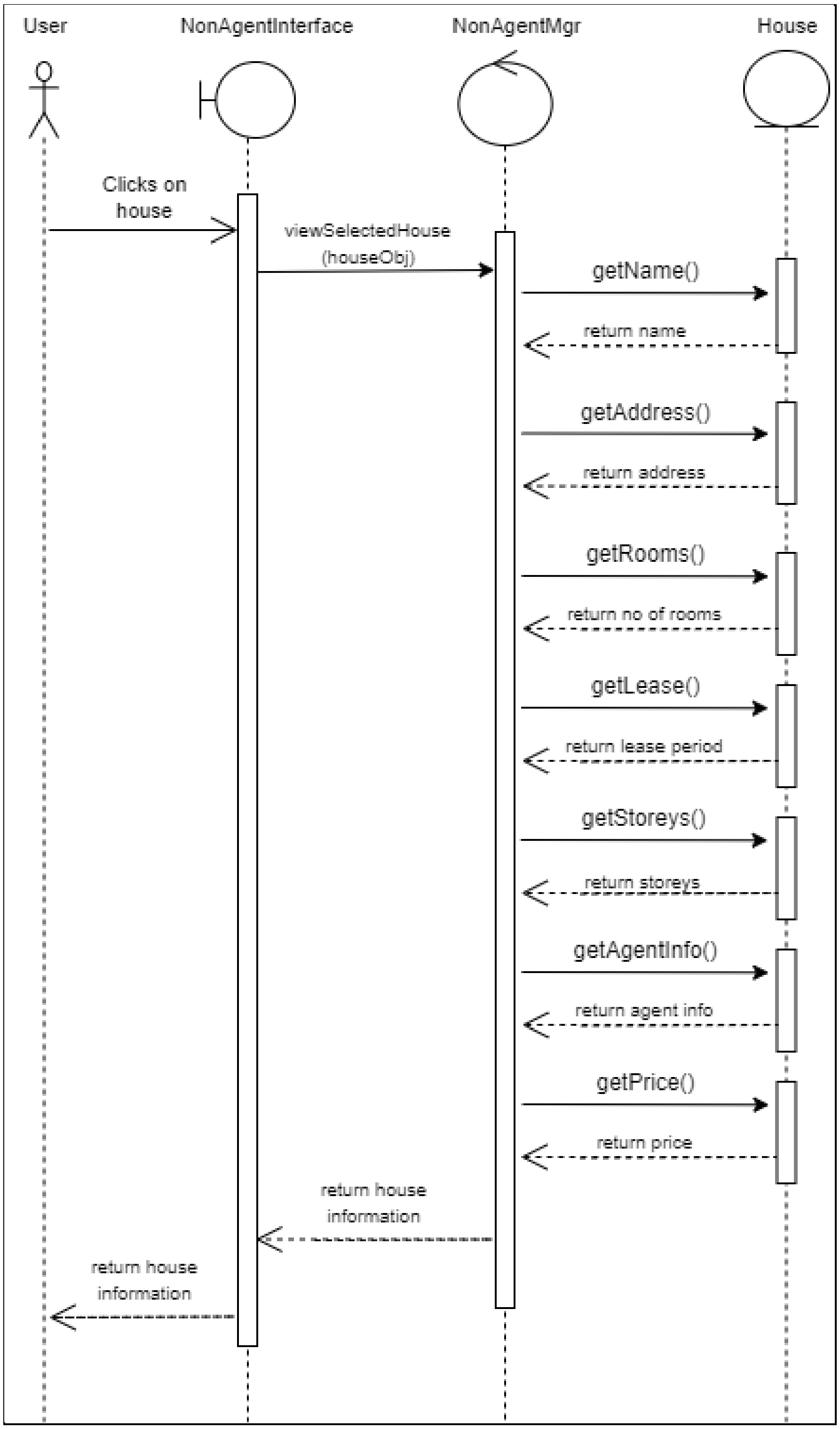
2.2.3 Access house listings

This function allows the user to access different house listings by searching using keywords. This feature can be accessed without logging in by the general users.


User
Agent Interface
NonAgent Interface
AgentMgr
NonAgentMgr
House
alt
[User == Agent]
goes to main menu
opt
[sort]
sort(price, room, yearsOfLease, stories, priInc,secInc)
getListOfHouses(price, room, yearsOfLease, stories, priInc, secInc)
opt
[search]
alt
[search keyword]
user presses "search by keyword"
onscreen keyboard pops up
user types in keyword and presses enter
keySearch(keyword)
getListOfHouses(keyword)
opt
sort(price, room, yearsOfLease, stories, priInc,secInc)
getListOfHouses(price, room, yearsOfLease, stories, priInc, secInc)
[User == NonAgent]
goes to main menu
opt
[sort]
sort(price, room, yearsOfLease, stories, priInc,secInc)
getListOfHouses(price, room, yearsOfLease, stories, priInc, secInc)
opt
[search]
alt
[search keyboard]
user presses "search by keyword"
onscreen keyboard pops up
user types in keyword and presses enter
keySearch(keyword)
getListOfHouses(keyword)
opt
sort(price, room, yearsOfLease, stories, priInc,secInc)
getListOfHouses(price, room, yearsOfLease, stories, priInc, secInc)

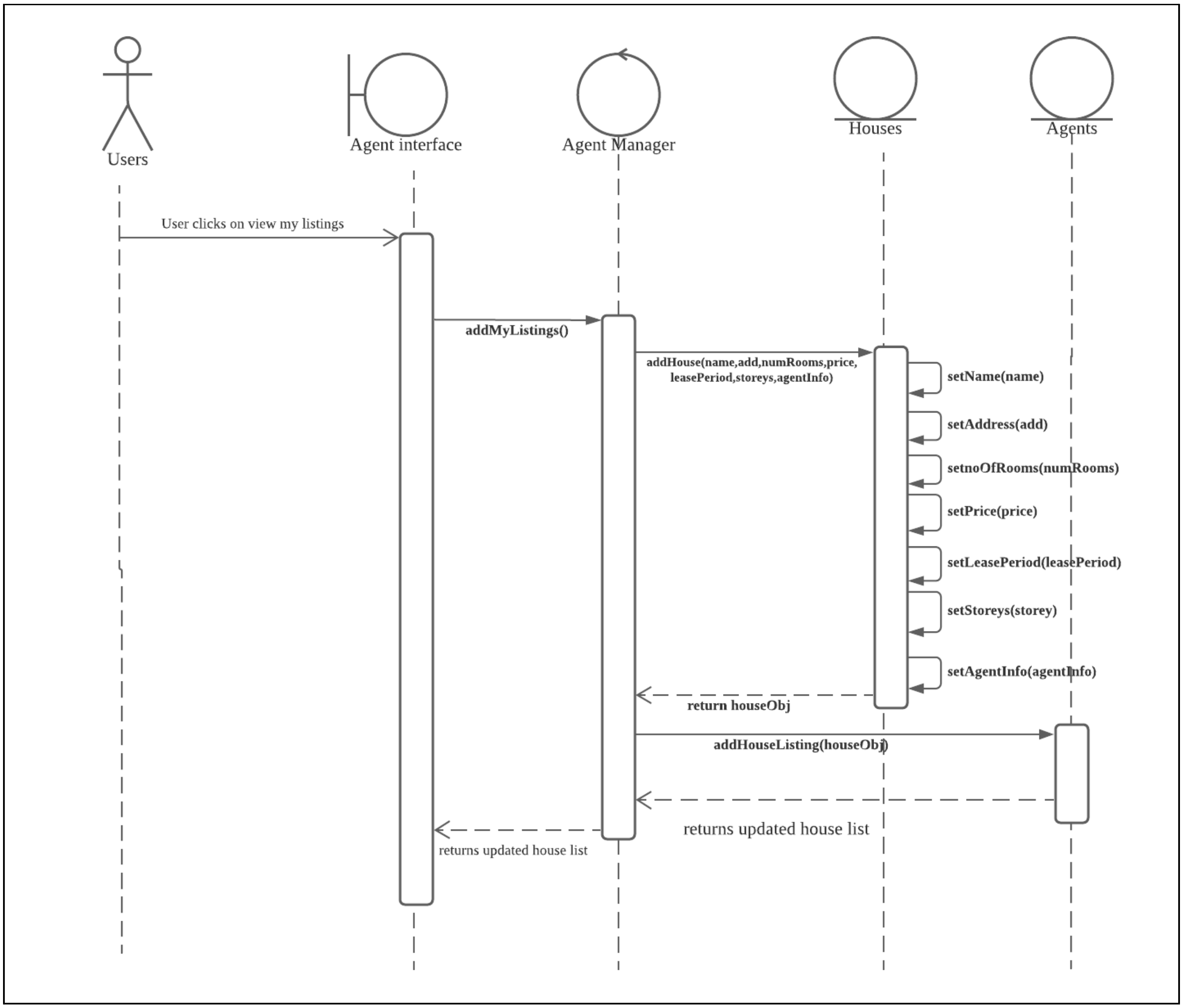
2.2.4 View house information

This function allows the user to view house information like address, number of bedrooms, nearest MRT by clicking on a particular house listing.



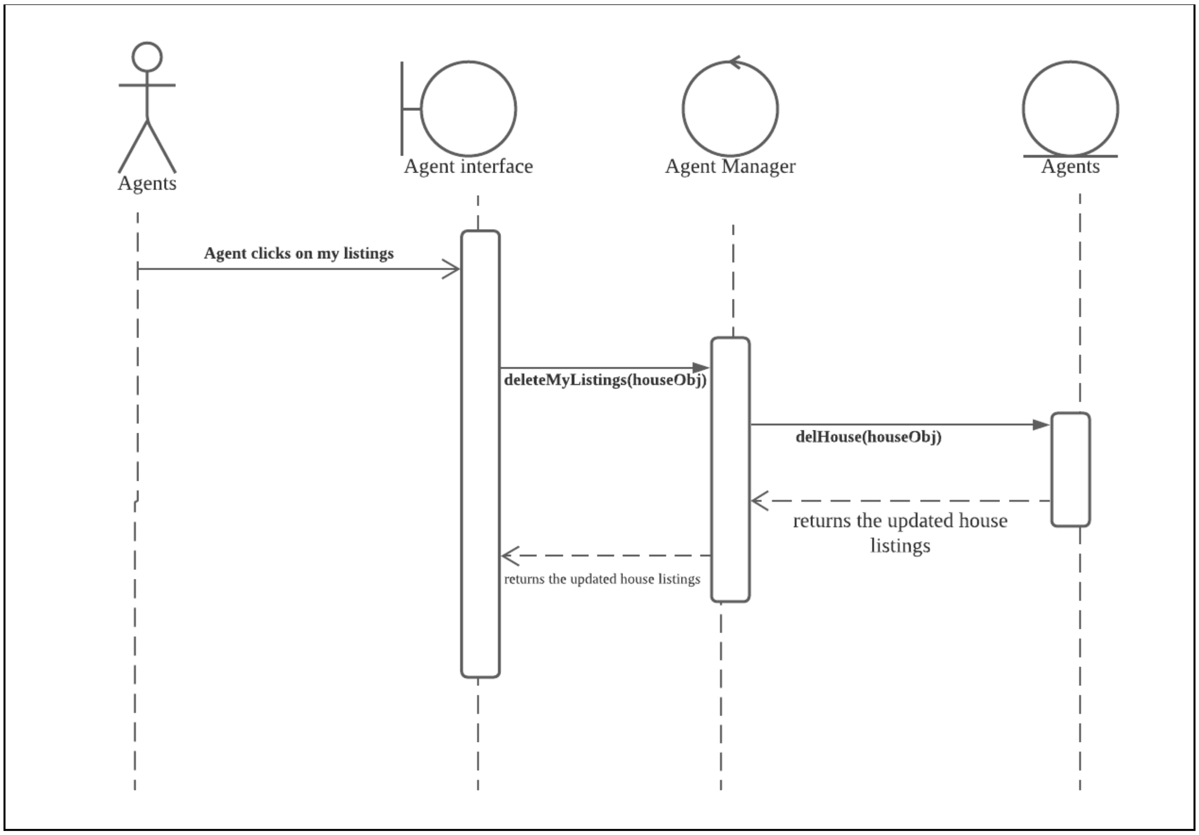
2.2.5 Add house listing

This function allows the agents to add to their own listings. This function requires the agent to login and add house information which will then get saved into the database.


Users
Agent interface
Agent Manager
Houses
Agents
User clicks on view my listings
addMyListings()
addHouse(name,add,numRooms,price,
leasePeriod,storeys,agentInfo)
setName(name)
setAddress(add)
setnoOfRooms(numRooms)
setPrice(price)
setLeasePeriod(leasePeriod)
setStoreys(storey)
setAgentInfo(agentInfo)
return houseObj
addHouseListing(houseObj)
returns updated house list
returns updated house list

2.2.6 Delete house listing

Similar to the add house listing function, this function allows users logged in as agents to delete specific listings that they have once added which are sold or not available for sale.


Agents
Agent interface
Agent Manager
Agents
Agent clicks on my listings
deleteMyListings(houseObj)
delHouse(houseObj)
returns the updated house listings
returns the updated house listings

2.3 User Classes and Characteristics

Users of this system will be divided into three categories: non-agents (general, non-login users), non-agents (logged-in users), and agents. User classes are differentiated based on function accessibility levels. Users of this product should be able to access a housing buying/renting (non-agents) and selling (agents) platform.

- Non-agents (general, non-login users): All general non-agent users can use this application. Their knowledge level on software engineering should not matter as the user interface follows user design principles, allowing users to follow a conventional process.
- Non-agents (logged-in users): All logged in non-agent users can use this application. Their knowledge level on software engineering should not matter as the user interface follows user design principles, allowing users to follow a conventional process.

- Agents: All agent users can use this application. Their knowledge level on software engineering should not matter as the user interface follows user design principles, allowing users to follow a conventional process.

Non-agents(general users) should be able to perform the following functions:

1. Access house listings
    a. Search house listings using keywords
    b. Sort house listings using advanced filters
2. Search furniture using AR
3. View house information
4. View agent profiles
5. View grant information

Non-agents (logged-in users) should be able to perform the following functions:

1. Login
2. Access house listings
    a. Search house listings using keywords
    b. Sort house listings using advanced filters
3. Search furniture using AR
4. View house information
    a. Favorite houses
    b. Access financial calculator
5. Communicate with agents via chat system
6. View agent profiles
7. View grant information
    a. Check eligibility for grants

Agents should be able to perform the following functions:

1. Login
2. Manage house listings
    a. Add listing
    b. Delete listing
3. Access house listings
    a. Search house listings using keywords
    b. Sort house listings using advanced filters
4. Search furniture using AR
5. Communicate with non-agents via chat system

2.4 Operating Environment

This product is an Android based mobile application using Android Studio for development. Android Studio is an Integrated Development Environment (IDE) which allows users to develop Android based applications. App modules including manifests (.xml files), java (.java source code files), and res (non-code resources) folders were used for the creation of this product.

2.5 Design and Implementation Constraints

1. The system database uses csv files which are fast and do not consume large amounts of memory. Files in csv format are easy to programmatically manipulate and can be opened with any text editor in Windows, allowing the database to be conveniently migrated on different machines.
2. This system will be developed using Android Studio.
3. As part of standard operating procedures, test cases were performed during the design phase. Test cases were designed based on features, use cases, flow, required tasks, and expected outcomes.
4. Database Design: The database structure should be as complete as possible during the test case performance stage, but room for modification is left depending on the outcomes of the test cases.
5. Security considerations: This app uses Singpass which is a secure digital identity for Singapore citizens. All information of users comes from Singpass, therefore, editing of salary, password etc. must be done through Singpass. To prevent leaking of data, the product gives each user 5 chances to enter the correct username and password.
6. Design conventions: Design conventions and elements can be easily modified and implemented due to the res file formats provided in Android Studio.

2.6 User Documentation

There are no needed user documentation components involved in this product.

2.7 Assumptions and Dependencies

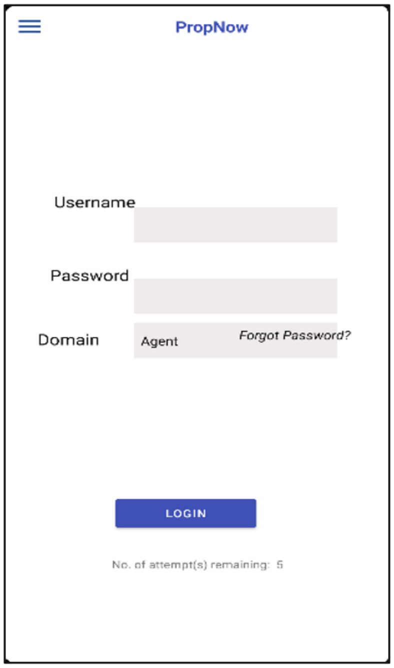
1. Since this application is an android application, a phone with android OS is required.
2. It is assumed that all users have good internet connection.
3. It is assumed that all logged-in non-agent/agent users will have an account in Singpass.
4. It is assumed that all users know the English language, as the user interface is provided in English.

3. External Interface Requirements

3.1 User Interfaces

The user interface would be straightforward and clear, with language that is familiar to the system's intended users. To remove the need for user training for infrequent users, the device would have a basic gui that is compatible with industry standard interfaces. Our team will assess related systems' user interfaces and adapt the findings accordingly. Application testing can be used to ensure that the user interface is straightforward (commonly understood vocabulary), complete (users will do all operations from the interface), and reliable (buttons and wording are the same throughout the system).

Here are some of user interface samples which shows the sequence of actions needed to add a new house listing :



Domain: Agent

No. of attempt(s) remaining: 5

Figure 1
The user needs to be registered into the system to login as an agent. The user can login by clicking on the login button after adding the correct username and password.

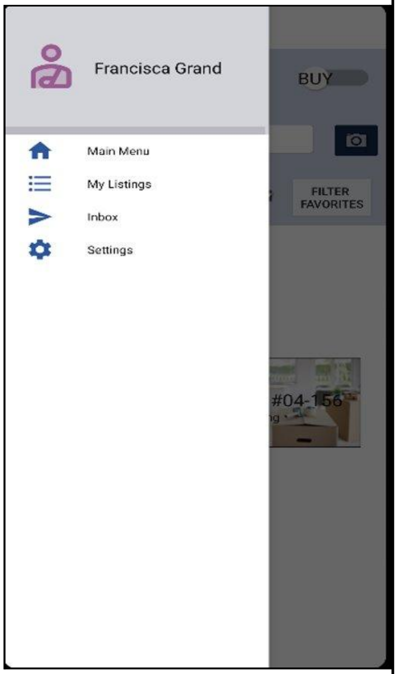
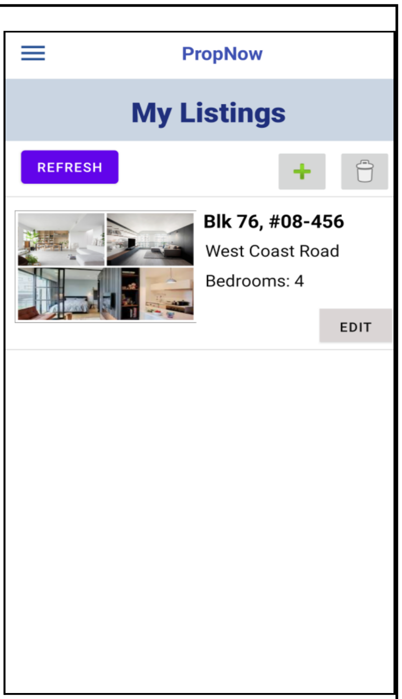
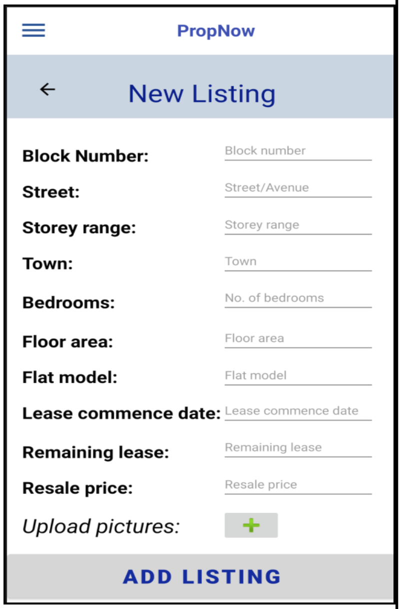


Figure 2
After the user is logged in as an agent, her name will appear on the sidebar and then she can access the my listings function.



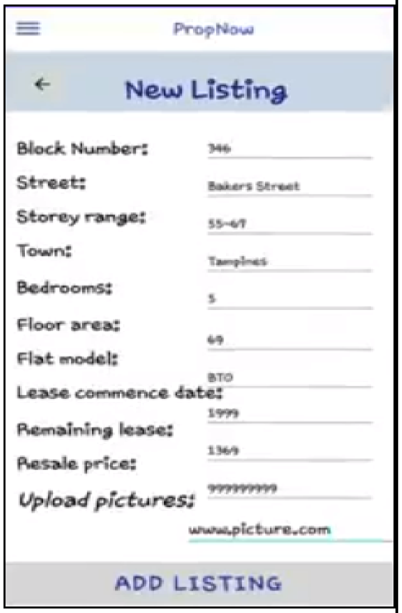
Blk 76, #08-456
West Coast Road
Bedrooms: 4

Figure 3
The my listings page has the option to add or delete a particular listing. The agent clicks on the add button to add a new listing.



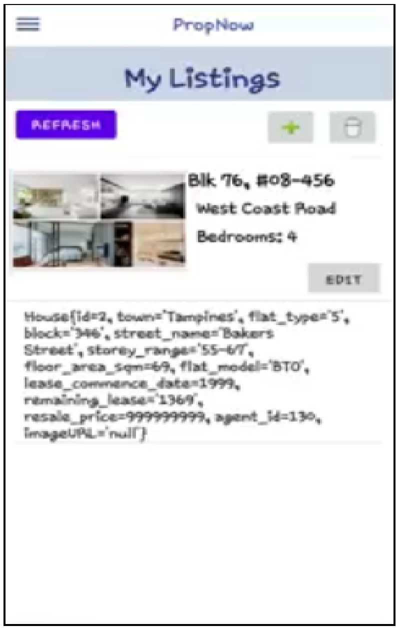
Block Number: Block number
Street: Street/Avenue
Storey range: Storey range
Town: Town
Bedrooms: No. of bedrooms
Floor area: Floor area
Flat model: Flat model
Lease commence date: Lease commence date
Remaining lease: Remaining lease
Resale price: Resale price
Upload pictures:

Figure 4
After clicking on the add button the agent arrives on this page and then inputs the details of the house she wants to add.



Block Number: 346
Street: Bakers Street
Storey range: 55-67
Town: Tampines
Bedrooms: 5
Floor area: 69
Flat model: BTO
Lease commence date: 1999
Remaining lease: 1369
Resale price: 999999999
Upload pictures: www.picture.com

Figure 5
Agent adds in the information in the given sections. All sections must be filled in in order to add listing.



Blk 76, #08-456
West Coast Road
Bedrooms: 4

```
House{id=2, town='Tampines', flat_type='5',
block='346', street_name='Bakers
Street', storey_range='55-67',
floor_area_sqm=69, flat_model='BTO',
lease_commence_date=1999,
remaining_lease='1369',
resale_price=999999999, agent_id=130,
imageURL='null'}
```

Figure 6
After all information has been added, the user clicks on the button ‘Add Listing’ and ‘Refresh’ to see the added listing.

3.2 Hardware Interfaces

There are no additional hardware interfaces needed. Standard hardware and computer communications resources can be included in the device.

3.3 Software Interfaces

The framework can make use of widely distributed computing tools on the Internet. This includes, but is not limited to, the Android studio (Gradle plugin version 4.1.2, Gradle version 6.5) , Java, XML and other necessary resources needed by the mobile application.

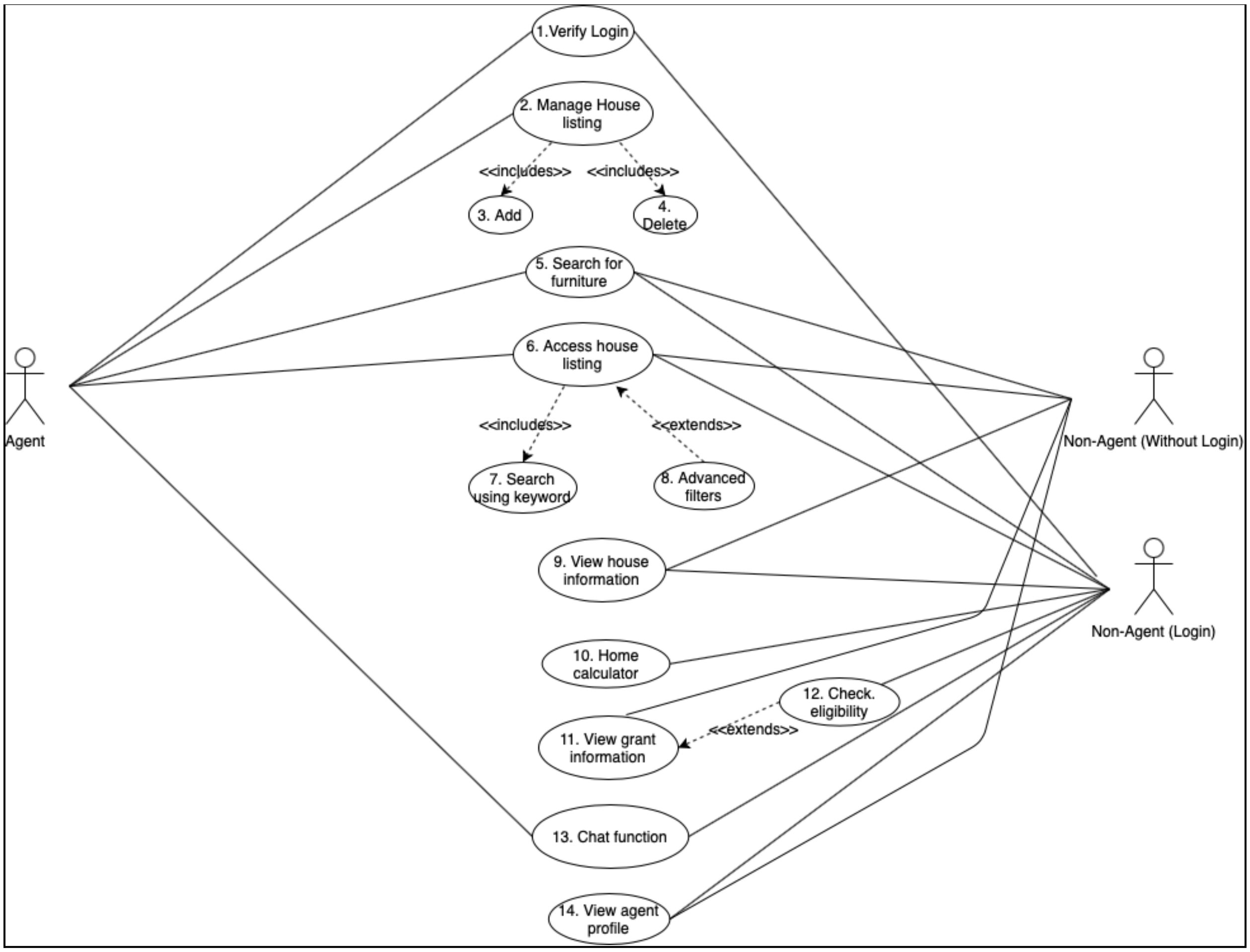
This application also used 4 Comma-Separated Values (CSV) files to store the information of agents, non-agents, messages and houses. SQLite was used to store the different entity class objects in separate tables.

Several functions used in the application use external API such as TensorflowLite object detection and machine learning functionalities for the AI furniture recognition, where images from Google’s database were used to train the model (using this functionality consumes RAM on mobile). Databases implemented for houses are implemented in SQLite as compared to agent and non-agent databases which use only csv load in since they do not need to be edited(Singpass assumption). SQLite implementation uses an extension from SQLiteOpenHelper which is built into Android studio(import android.database.sqlite.SQLiteOpenHelper).

3.4 Communications Interfaces

The system uses common communication resources. For the View Grants Information page, there are hyperlinks which will redirect the users to the relevant pages for more information on the specific grants. This uses HTTP protocol for communication with the web browser and the web server.

4. System Features



Use case diagram

4.1 Verify login feature

4.1.1 Description and Priority

[Priority: Must Have]

This feature allows the user to log-in as an agent or a non agent user, depending on the domain chosen by the user, certain functions will be made available. For example, when a user logs in as a non-agent, he can access features like home calculator and check eligibility for grants.

4.1.2 Stimulus/Response Sequences

Use Case ID:	1
Use Case Name:	Verify Login

Actor:	Agent, Non-Agent (Login)
Description:	This use case allows the user to log in to the system
Preconditions:	1. Existing user account
Postconditions:	1. Login successful and profile page displayed
Priority:	Must have
Frequency of Use:	High
Flow of Events:	1. System displays login page 2. User enters username and password and their domain 3. User clicks login button 4. System verify the login 5. If the login is verified, system shows main page depends on domain of user
Alternative Flows:	AF-S4: If user inputs wrong password 1. System display error message 2. System return to step 2
Exceptions:	EX1 : If user inputs wrong password 5 times 1. User is unable to attempt login and has to reboot the application again
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	Agents cannot change passwords on their own, they need to contact the developers/support. Non-Agent cannot update salary etc. as information is retrieved from SingPass. Non Agent password is same as that of SingPass, thus password

	reset for them is at SingPass.

4.1.3 Functional Requirements

1. Users are able to log in (Users - Agent/Non-agent)

- **1.1.** Users can choose to log in optionally through their domain, username, and password. Those who are not logged in have limited access to functions
  - **1.1.1.** If user is logged in as an agent, they can access the agent main menu
    - **1.1.1.1.** Agent users can access and manage the listings of houses(selling/renting)
    - **1.1.1.2**. Agents cannot change the password on their own, they need to contact the developers/support.
  - **1.1.2.** If the user is logged in as non-agent, they can access the non-agent main
    - **1.1.2.1.** Non-Agent cannot update salary etc. as information is retrieved from SingPass. Non Agent password is the same as that of SingPass, thus password reset for them is at SingPass.

2. Users are not able to log in

- **2.1.** The user gets a total of 5 trials before getting blocked from accessing their account i.e the user can input a wrong username or password or domain a maximum of five times.

4.2 Manage house listings

4.2.1 Description and Priority

[Priority: Must Have]

This feature allows agent users to manage house listings which are available for sale.

4.2.2 Stimulus/Response Sequences

Use Case ID:	2
Use Case Name:	Manage House Listing

Actor:	Agent (Initiating Actor)

Description:	This use case allows agent to manage the house listing
Preconditions:	1. Agent already login to the system 2. The house listing is owned by the agent
Postconditions:	1. Information of the house listing is changed
Priority:	Must have
Frequency of Use:	High
Flow of Events:	1. Agent click on the page of the houses listing they owned 2. Agent can choose either to add OR update OR delete any one of the house listing 3. System show the changed information of the house listing
Alternative Flows:	-
Exceptions:	EX1 : In the case of a network failure, 1. The message for the developer is not sent 2. Instead, a pop-up saying ‘No network connection’ will appear
Includes:	Add House Listing, Delete House Listing

Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.2.3 Functional Requirements

1. User is logged in as an agent

1.1. Agent users can access and manage the listings of houses(selling/renting)

2. User is not logged in

2.1. The user gets a total of 5 trials before getting blocked from accessing their account i.e the user can input a wrong username or password a maximum of five times.

4.3 Add house listing

4.3.1 Description and Priority

[Priority: Must Have]

This feature allows the user who is logged in as an agent, to add to his listings.

4.3.2 Stimulus/Response Sequence

Use Case ID:	3
Use Case Name:	Add House Listing

Actor:	Agent (Initiating Actor)
Description:	This use case allows agent to add into the house listing
Preconditions:	1. Agent already login to the system 2. The new house listing is owned by the agent

Postconditions:	1. New house listing is added
Priority:	Must have
Frequency of Use:	Medium
Flow of Events:	1. Agent click on the page of the houses listing they owned 2. Agent click the plus icon 3. Agent inputs information of the new house listing 4. Agent click the submit button 5. System show information of added house listing
Alternative Flows:	AF-S5: There is important information that have not been filled by Agent 1. System show error message "This information is required" 2. System return to step 3
Exceptions:	EX1 : In the case of a network failure, 1. The message for the developer is not sent 2. Instead, a pop-up saying 'No network connection' will appear
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.3.3 Functional Requirements

1. User is logged in as an agent

1.1. Agent users can access and manage the listings of houses(selling/renting)

1.1.1. Agent users can add a listing

1.1.1.1. Every listing should have an auto-generated unique reference number

2. User is not logged in

2.1. The user gets a total of 5 trials before getting blocked from accessing their account i.e the user can input a wrong username or password a maximum of five times.

4.4 Delete house listing

4.4.1 Description and Priority

[Priority : Must have]
This feature allows the user logged in as an agent to delete his listings if they are not available for sale.

4.4.2 Stimulus/Response Sequences

Use Case ID:	4
Use Case Name:	Delete House Listing

Actor:	Agent (Initiating Actor)
Description:	This use case allows agent to add into the house listing
Preconditions:	1. Agent already login to the system 2. The house listing is owned by the agent
Postconditions:	The house listing is deleted
Priority:	Must have
Frequency of Use:	Medium
Flow of Events:	1. Agent click on the page of the houses listing they owned 2. Agent click on the bin button 3. Agent choose one of the house listing they want to remove 4. System show the confirmation to delete 5. If agent click “Yes” to the confirmation, system delete the information of chosen house listing
Alternative Flows:	AF- S5: If the agent clicks “No” to the confirmation 1. System return to page of list of houses

Exceptions:	EX1 : In the case of a network failure, 1. The message for the developer is not sent 2. Instead, a pop-up saying ‘No network connection’ will appear
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.4.3 Functional Requirements

1. User is logged in as an agent

1.1. Agent users can access and manage the listings of houses(selling/renting)

1.1.1. Agent users can delete a listing

1.1.1.1. Every listing should have an auto-generated unique reference number

2. User is not logged in

2.1. The user gets a total of 5 trials before getting blocked from accessing their account i.e the user can input a wrong username or password a maximum of five times.

4.5 Search furniture using AR

4.5.1 Description and Priority

[Priority : Low]
This feature allows the user to search furniture using augmented reality. Users will be able to search objects like chairs, desk, bed etc using this function.

4.5.2 Stimulus/Response Sequences

Use Case ID:	5
Use Case Name:	Search Using AR

Actor:	Agent(Initiating actor), Non-Agent(login)(Initiating actor), Non-Agent(Without login)(Initiating actor)

Description:	Allows users to search for furniture by scanning an image.
Preconditions:	Camera permission is required.
Postconditions:	Result is displayed.
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1. User selects the AR camera button. 2. System changes to camera view. 3. User aims the phone camera at the object/picture desired for the query. 4. System identifies the object. 5. System proceeds to return the identified object.
Alternative Flows:	-
Exceptions:	EX1 : In the case of no camera available, 3. The AR search can neither work nor recognize the object and the user cannot use this function. 4. Instead, a pop-up saying 'No camera access' will appear
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.5.3 Functional Requirements

1. Users can access the AR feature from the sidebar

1.1 They can point at objects or scan an image to get the desired results.

2. In case of no camera access

2.1 Users can not access AR feature

2.2 A pop up saying "No camera access will appear".

4.6 Access house listings

4.6.1 Description and Priority

[Priority : Must have]
This feature allows the users to view house listings which are available for sale, they can be accessed without logging in.

4.6.2 Stimulus/Response Sequences

Use Case ID:	6
Use Case Name:	Access House Listing

Actor:	Agent(Initiating actor), Non-Agent(login)(Initiating actor), Non-Agent(Without login)(Initiating actor)
Description:	Allows user to access lists of houses available for rent/sale
Preconditions:	-
Postconditions:	Users can view/select house listings
Priority:	Must Have
Frequency of Use:	High
Flow of Events:	The main menu will show all houses available for sale. 1. If the user presses the button which toggles “buy” to “rent”, it will show all houses available for renting instead. User can toggle between selling and renting modes 2. User can scroll down and up to view all houses available from the list 3. User can decrease the number of houses available in the listing with conditions by using search functions in included use cases Search Using Keyword 4. Users can proceed and select a particular listing which brings them to specific information regarding the listing which is the use case View House Information.
Alternative Flows:	

Exceptions:	EX1 : In the case of a network failure, 1. Listings will not be shown in the main menu because it cannot be fetched. 2. Instead, a pop-up saying ‘No network connection’ will appear
Includes:	Search Using Keyword
Special Requirements:	-
Assumptions:	-
Notes and Issues:	Has extended use case Advanced filters.

4.6.3 Functional Requirements

1.Users can access the buying or renting housing list

1.1. Users must search up houses for sale/rent by either keyword/address or augmented reality to detect buildings

1.1.1. Users searching through keyword should use region (North/West etc), address or parts of the address

1.1.2. Users searching through augmented reality should point their device camera to building, which will cause device to recognise that building and return a list of available houses in that building

1.2.. Users can arrange listings based on decrementing or incrementing primary and secondary filters such as pricing, number of rooms etc.

4.7 Search using keyword

4.7.1 Description and Priority

[Priority : High]
This feature allows the user to search houses which are available for sale using the search bar which in turn displays results according to keywords.

4.7.2 Stimulus/Response Sequences

Use Case ID:	7
Use Case Name:	Search Using Keyword

Actor:	Agent(Initiating actor), Non-Agent(login)(Initiating actor), Non-Agent(Without login)(Initiating actor)
Description:	Allows user to search for houses available for sale/rent through a search bar, which uses keywords to identify relevant results
Preconditions:	Requires user to access the use case Access House Listing
Postconditions:	System return list of houses that fulfils keyword search
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User press on search bar. 2. System triggers the keyboard to pop up for input by the user. 3. Users can type keywords specifying type/location of houses for viewing or even the address of a specific house. 4. User presses enter to trigger search. 5. System will then show a reduced list of houses in the main menu
Alternative Flows:	-
Exceptions:	EX1 : In the case of a network failure, 5. The user is unable to search 6. Instead, a pop-up saying 'No network connection' will appear
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.7.3 Functional Requirements

1. Users are able to log in (Users - Agent/Non-agent)

1.1. Users can choose to log in optionally through their domain, username, and password. Those who are not logged in have limited access to functions

1.1.2. If user has not yet logged in or is logged in as a non-agent, they can access the non-agent main menu

1.1.2.1. Users can access the buying or renting housing list

1.1.2.1.1. Users must search up houses for sale/rent by either keyword/address

1.2.1.1.1.1. Users searching through keyword should use region (North/West etc), address or parts of the address

2. User is not logged in

2.1. The user gets a total of 5 trials before getting blocked from accessing their account i.e the user can input a wrong username or password a maximum of five times.

4.8 Advanced filters

4.8.1 Description and Priority

[Priority : High]
This feature allows the user to sort houses based on primary and secondary options.

4.8.2 Stimulus/Response Sequences

Use Case ID:	8
Use Case Name:	Advanced filters

Actor:	Agent(Initiating actor), Non-Agent(login)(Initiating actor), Non-Agent(Without login)(Initiating actor)
Description:	Allows users to sort all houses in the main menu base 'rimary/Secondary conditions and/or favorites.
Preconditions:	Requires user to access the use case Access House Listing
Postconditions:	System return list of houses available after arranging it with sort filter as well as limiting range of listings from filter restrictions
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User presses button “Advanced filter” 2. System pops up a list of filters for users to apply to the search list. This includes a primary and secondary category to arrange the house listing(e.g. By price then by years of lease available).

	Users can further restrict listing by sliding the number of rooms desired, or by toggling to view only favourites. 3. After applying their desired filters, the user presses the back button to return to the main menu. 4. System then shows arranged listing with filters applied 5. Users can switch between increasing/decreasing sort for both primary and secondary categories in the main menu.(e.g. Increasing price followed by decreasing lease years) 6. After changing between incremental/decremental sorting, the system would arrange the listing accordingly.
Alternative Flows:	-
Exceptions:	EX1 : In the case of a network failure, 7. There will be nothing to sort since there is no listings to be fetched 8. Instead, a pop-up saying ‘No network connection’ will appear
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.8.3 Functional Requirements

1. Users can access the main menu of the application

1.1. User clicks on the button ‘Advanced Filters’

1.1.1. A list of filters for users to apply are shown including primary and secondary drop down menus and sliders for ranges

1.1.1.1 User presses on ‘Apply’ button to apply desired filters

1.1.1.1.1.System returns listings with filters applied

4.9 View House Info

4.9.1 Description and Priority

[Priority : Must have]
This feature allows non-agent users to view information of houses.

4.9.2 Stimulus/Response Sequences

Use Case ID:	9
Use Case Name:	View House Information

Actor:	Non-agent(login) (Initiating actor), non-agent(without login) (Initiating actor)
Description:	This use case allows the non-agents, with or without login, to view the full information of a listing
Preconditions:	Non-agent clicks on the listing
Postconditions:	Non-agent leaves the page by clicking on the back tab
Priority:	Must have
Frequency of Use:	High
Flow of Events:	1. Non-agent clicks on the house listing 2. The page with the full information of the house opens up 3. The non-agent can scroll down until the end of the page to view all the information 4. Non-agent leaves the page by clicking on the back tab
Alternative Flows:	
Exceptions:	In the case of a network failure, 1. The page with the house information will not open up 2. Instead, a pop-up saying ‘No network connection’ will appear
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.9.3 Functional Requirements

1. Users are able to log in (Users - Agent/Non-agent)

1.1. Users can choose to log in optionally through their domain, username, and password. Those who are not logged in have limited access to functions

1.1.2. If user has not yet logged in or is logged in as a non-agent, they can access the non-agent main menu

1.1.2.1. User can click on a house listing that they want to access

1.1.2.2. System returns page with information of house

4.10 Home Calculator

4.10.1. Description and Priority

[Priority : High]
This feature allows the non-agent users who are logged in, to calculate the equated monthly installment and interest payable after inputting property price, interest rate and the number of years.

4.10.2. Stimulus/Response Sequences

Use Case ID:	10
Use Case Name:	Home Calculator

Actor:	Non-agent(login) (Initiating actor)
Description:	This use case allows the non-agents, with login, to calculate the final price they have to pay for the house and give a breakdown of the payment
Preconditions:	1. Non-agent must be in the sidebar 2. Non-agent must have an account in the system and must be logged in
Postconditions:	Non-agent receives a breakdown of the payment they have to make if they want to purchase the house
Priority:	High
Frequency of Use:	Medium
Flow of Events:	1. Non-agent clicks on the financial calculator tab in the sidebar 2. System shows the details of the non-agent(buyer 1) and the option to enter details of buyer 2 with a add icon 3. After confirming the details are right, the non-agent clicks

	on the calculate button 4. The system then provides a breakdown of the final payment for the house
Alternative Flows:	[AF-S2] 1. The non-agent clicks on the add icon to add details of buyer 2 2. If the profile of the non-agent contains the details of buyer 2, the details will appear on the home calculator page. OR If the profile of the non-agent does not contain the details of buyer 2, the non-agent will automatically be brought to the edit profile page. 3. The non-agent inputs the information [AF-S1] If the non-agent is not logged in, 1. Non-agent clicks on the financial calculator tab 2. The system shows a login pop-up, prompting the non agent to log in 3. If login is successful, non-agent can then access the financial calculator
Exceptions:	EX1 : After 5 tries of unsuccessful login 1. The non-agent's account will be blocked and an email will be sent to him/her for security reasons. EX2 : In the case of a network failure, 1. The page with the financial calculator will not open up 2. Instead, a pop-up saying 'No network connection' will appear
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.10.3 Functional Requirements

1. Users are able to log in (Users - Agent/Non-agent)

1.1. Users can choose to log in optionally through their domain, username, and password. Those who are not logged in have limited access to functions

1.1.2. If user has not yet logged in or is logged in as a non-agent, they can access the non-agent main menu

1.1.2.1. Users click on the Home Calculator in the sidebar to access home calculator

1.1.2.1.1 Users enter property price, interest rate, and years in the provided spaces

1.1.2.1.2. Users press on the calculate button

1.1.2.1.3. System returns calculated EMI and total interest

2. User is not logged in

2.1. The user gets a total of 5 trials before getting blocked from accessing their account i.e the user can input a wrong username or password a maximum of five times.

4.11 View grant information

4.11.1 Description and Priority

[Priority : Low]
This feature allows the non-agents, with or without login, to view the details of all the available grants.

4.11.2 Stimulus/Response Sequences

Use Case ID:	11
Use Case Name:	View Grant Information

Actor:	Non-agent(login) (Initiating actor), non-agent(without login) (Initiating actor)
Description:	This use case allows the non-agents, with or without login, to view the details of all the available grants
Preconditions:	Non-agent must have opened the side bar of the system
Postconditions:	The system shows the page containing the grant information
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1. Non-agent clicks on the ‘View Grant Information’ tab 2. The page with the full information of all the available grants opens up

	3. The non-agent can scroll down until the end of the page to view all the information 4. Non-agent leaves the page by clicking on the back tab
Alternative Flows:	-
Exceptions:	EX1 : In the case of a network failure, 1. The page with the grant information will not open up 2. Instead, a pop-up saying ‘No network connection’ will appear
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.11.3 Functional Requirements

1. Users are able to log in (Users - Agent/Non-agent)

1.1. Users can choose to log in optionally through their domain, username, and password. Those who are not logged in have limited access to functions

1.1.2. If user has not yet logged in or is logged in as a non-agent, they can access the non-agent main menu

1.1.2.1. Non-agent users can view grant information by accessing the sidebar

2. User is not logged in

2.1. The user gets a total of 5 trials before getting blocked from accessing their account i.e the user can input a wrong username or password a maximum of five times.

4.12 Check eligibility

4.12.1 Description and Priority

[Priority : Low]
This use case allows the non-agents, with login, to check which grants they are eligible for.

4.12.2 Stimulus/Response Sequences

Use Case ID:	12
Use Case Name:	Check Eligibility

Actor:	Non-agent(login) (Initiating actor)
Description:	This use case allows the non-agents, with login, to check which grants they are eligible for
Preconditions:	1. Non-agent must be in the ‘View Grant Information’ page 2. Non-agent must have an account in the system and must be logged in
Postconditions:	The system shows a list of grants that the non-agent is eligible for
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1. Non-agent must click the ‘Check eligibility for grant’ button 2. The system then provides a list of eligible grants for the non-agent
Alternative Flows:	[AF-S1] If the non-agent is not logged in, 1. Non-agent clicks on the ‘Check grant eligibility’ link 2. The system shows a login pop-up, prompting the non agent to log in 3. If login is successful, non-agent can then access the grant eligibility feature OR If non-agent has no account, he/she can set up an account by signing up
Exceptions:	EX1 : After 5 tries of unsuccessful login 1. The non-agent’s account will be blocked and an email will be sent to him/her for security reasons. EX2 : In the case of a network failure, 1. The page with the list of eligible grants will not open up

	2. Instead, a pop-up saying ‘No network connection’ will appear
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	Extends from ‘View Grant Information’ use case

4.12.3 Functional Requirements

1. Users are able to log in (Users - Agent/Non-agent)

1.1. Users can choose to log in optionally through their domain, username, and password. Those who are not logged in have limited access to functions

1.1.2. If user has not yet logged in or is logged in as a non-agent, they can access the non-agent main menu

1.1.2.1. Non-agent users can view grant information by accessing the sidebar

1.1.2.1.1. Non-agents can check eligibility for grants by clicking the check eligibility button

2. User is not logged in

2.1. The user gets a total of 5 trials before getting blocked from accessing their account i.e the user can input a wrong username or password a maximum of five times.

4.13 Communication using chat system

4.13.1 Description and Priority

[Priority : High]
This feature allows both non agents and agents to communicate by text. They can only access this feature when they are logged in as agent or non agent.

4.13.2 Stimulus/Response Sequences

Use Case ID:	13
Use Case Name:	Communication Using Chat System

Actor:	Agent(Initiating actor), Non-Agent(login)(Initiating actor)
Description:	Allows both Agent and Non-Agent users to communicate via chat rooms
Preconditions:	User must have logged in as Agent/Non-Agent to be able to access Inbox
Postconditions:	System sends message from user to intended recipient
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User has logged in as a Non-Agent user 2. User has accessed the view House Listing function to view a particular house information which shows house information and Agent in-charge 3. User presses chat icon to initiate chat with agent-in charge 4. System goes to a chatroom with information about that particular listing in the chat. 5. User can choose to write more under the listing information to communicate their intention before pressing send button 6. After the send button is pressed, the system sends the message with the listing information and the user's additional remarks to the agent in-charge of that specific listing
Alternative Flows:	AF-S1: 1. User has logged in as Agent in use case LogIn, which the system uses to identify the user as actor: Agent. 2. User opens side bar and select "Inbox" 3. System shows a list of chat room where they have been contacted by non-agent users 4. User can choose to reply to received messages before pressing send button 5. After send button is pressed, system sends the reply to the initial non-agent user who initiated the chat AF-S2: 1. User opens side bar and select "Inbox" 2. System shows a list of chatrooms of messages they have sent to agents as well as conversations with agents. 3. User can choose follow-up with a previous

	conversation or to reply to received messages before pressing send button 4. After send button is pressed, system sends the reply to the agent user
Exceptions:	EX1 : In the case of a network failure, 1. The messages will not be sent 2. Instead, a pop-up saying ‘No network connection’ will appear
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	Agents cannot initiate chat, only Non-Agents.

4.13.3 Functional Requirements

1. Users are able to log in (Users - Agent/Non-agent)

1.1. Users can choose to log in optionally through their domain, username, and password. Those who are not logged in have limited access to functions

1.1.2. If user has is logged in as a non-agent, they can access the non-agent main menu

1.1.2.1. After logging in they can access the house listings

1.1.2.1.1. From the house listing page they can click on the chat button to communicate with the agent responsible for that particular listing.

2. User is not logged in

2.1. The user gets a total of 5 trials before getting blocked from accessing their account i.e the user can input a wrong username or password a maximum of five times.

4.14 View Agent profiles

4.14.1 Description and Priority

[Priority : Medium]
This feature allows the non-agents, with or without login, to view the profiles of all the agents registered with the system.

4.14.2 Stimulus/Response Sequences

Use Case ID:	14
Use Case Name:	View Agent Profiles

Actor:	Non-agent(login) (Initiating actor), non-agent(without login) (Initiating actor)
Description:	This use case allows the non-agents, with or without login, to view the profiles of all the agents registered with the system.
Preconditions:	Non-agent clicks on the view agent profile tab in the side bar
Postconditions:	Non-agent leaves the page by clicking on the back tab
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. Non-agent clicks on the view agent profile tab 2. The page with all the agents' profiles opens up 3. The non-agent can scroll down until the end of the page to view all the profiles 4. Non-agent leaves the page by clicking on the back tab
Alternative Flows:	-
Exceptions:	EX1 : In the case of a network failure, 1. The page with the agents' profile will not open up 2. Instead, a pop-up saying 'No network connection' will appear
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

4.14.3 Functional Requirements

1. Users are able to log in (Users - Agent/Non-agent)

> **1.1.** Users can choose to log in optionally through their domain, username, and password. Those who are not logged in have limited access to functions
>
> > **1.1.2.** If user has not yet logged in or is logged in as a non-agent, they can access the non-agent main menu
> >
> > > **1.1.2.1.** Non-agent users can view the profiles of all agent users by accessing the sidebar

2. User is not logged in

> **2.1.** The user gets a total of 5 trials before getting blocked from accessing their account i.e the user can input a wrong username or password a maximum of five times.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The application will run on all android devices and will respond to the size of the screen the application is running in. Functions like AR to search furniture should take less than 3 seconds to identify objects. Code submitted on SVN should be run on mobile(enable developer mode and USB debugging) than android studio for smooth performance.

5.2 Reliability Requirements

Most functions will not require internet connectivity. System components that require authentication through SingPass and network connectivity will function as long as the systems maintained by SingPass are available. The house listings are updated real time and the information on Non-Agent users are updated real time based on the Singpass database. Agent users have to be registered by the company before opening an account to increase reliability. Information/data entered by users is stored in the system's database so users don't have to key in the details every time they login/use the app.

5.3 Security Requirements

Password Verification occurs during log-in, and Singpass Authentication is used in this process since the account used to log in is the person's Singpass account.

a. Login limitations,
b. Ensuring Camera permission required before using application,
c. Only Singpass users can log-in; Thus only Singaporeans/Permanent Residents and registered Foreigners at Singapore with legitimate ID can log-in (enable access to sensitive information such as user salary etc.).

5.4 Availability Requirements

Only Singpass users can log-in, thus only Singaporeans/Permanent Residents and Long Term pass holders in Singapore with legitimate ID can log-in.

5.5 Usability Requirements

This app can be downloaded and used by anyone even if they're not residing in Singapore. This app has increased user friendliness by giving constant feedback, for example whenever there is a network connection error, there will be a pop-up saying "No network connection" to inform the users.

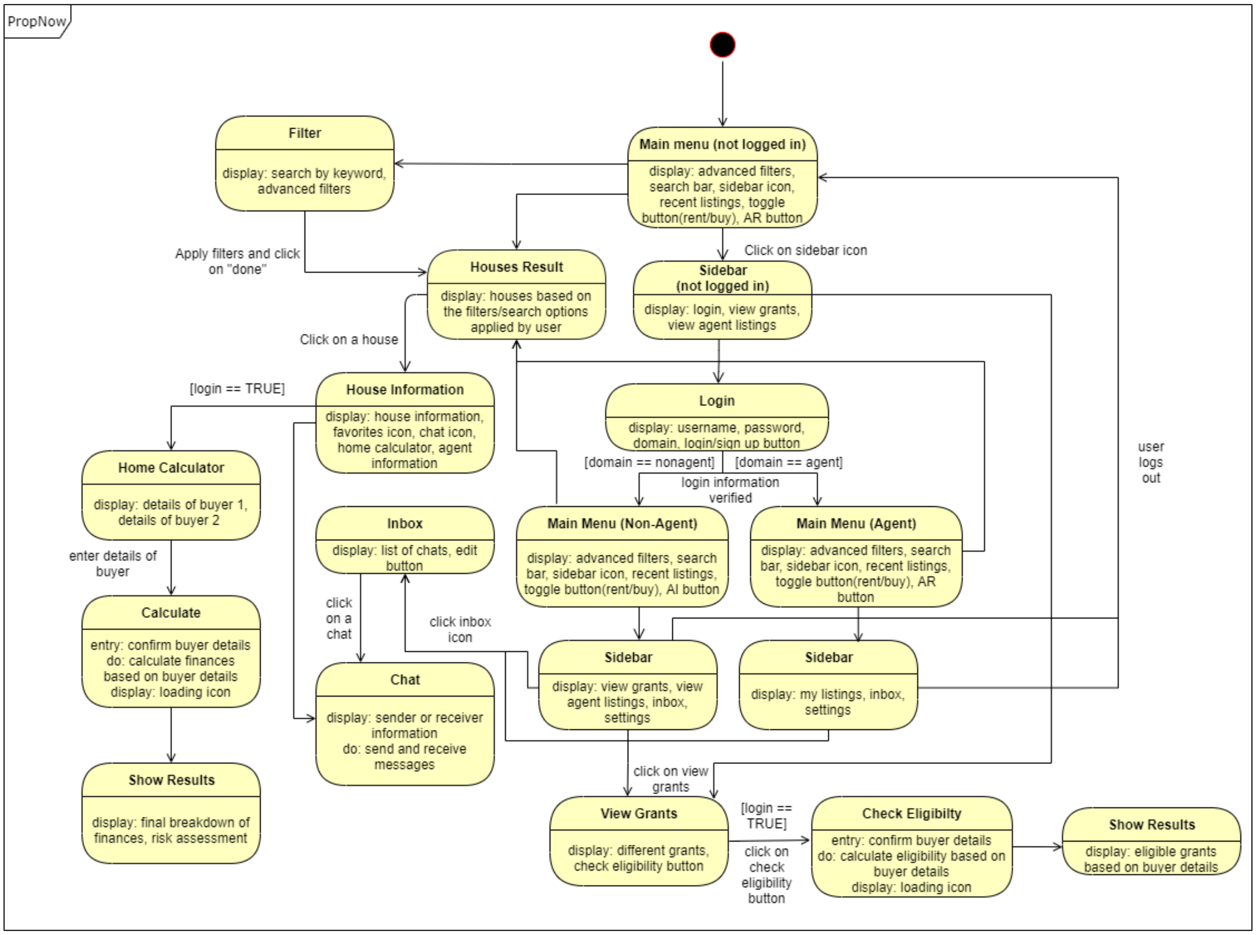
6. Appendix

6.1 Appendix A: Glossary

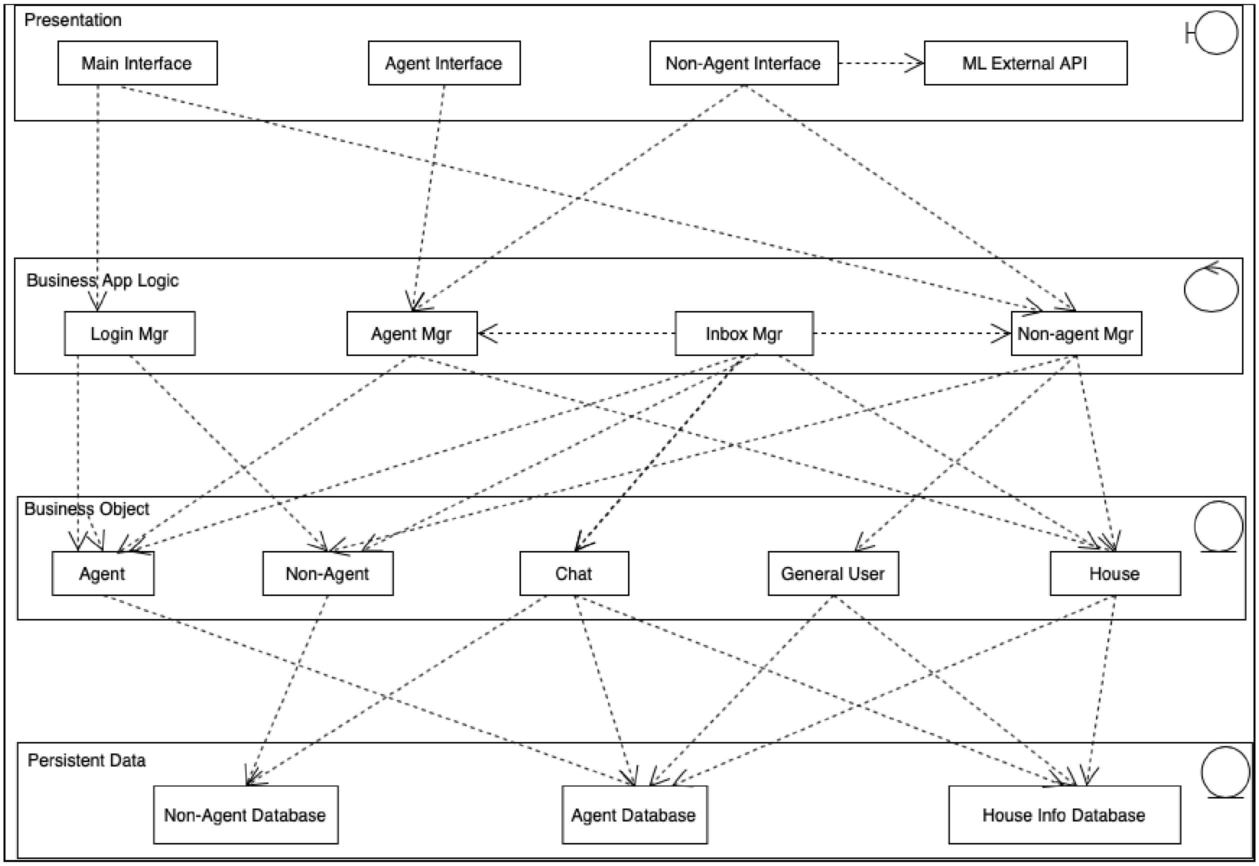
Data Dictionary

Term	Definition
Non-Agent Users	Members of the public who want to buy/rent a house
Agent Users	Those registered under agencies which are certified to mediate the selling and buying of HDBs
Listing	A posting of a house which contains house information for sale/rent
Primary Category Filter	First filter applied within given categories in the sorting of houses
Secondary Category Filter	Second filter applied within given categories in the sorting of houses
Grants	Subsidies given by the government in order to help purchase of resale/new flats.
Monthly Installment	A sum of money paid monthly in small parts in a fixed period of time in order to complete payment for purchase of a house
Eligibility for Grants	Qualifying for a grant Conditions: ● Citizenship ● Age ● Family nucleus ● Ethnic Integration Policy (EIP) and SPR quota ● Special conditions
Augmented Reality (AR)	An interactive view that shows the real-world environment where the objects detected can be analyzed for further information

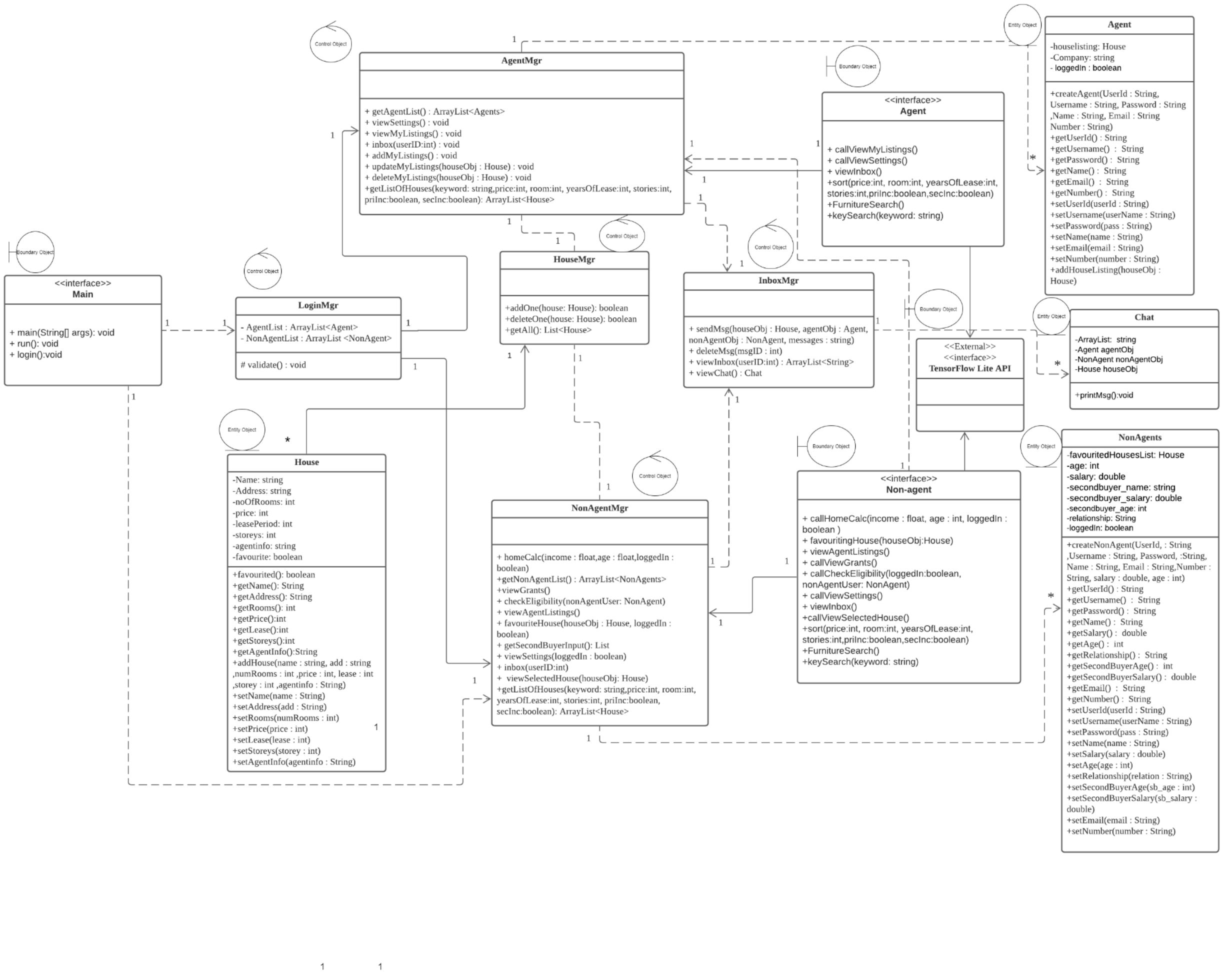
6.2 Appendix B: Analysis Models


PropNow
Filter
display: search by keyword, advanced filters
Main menu (not logged in)
display: advanced filters, search bar, sidebar icon, recent listings, toggle button(rent/buy), AR button
Apply filters and click on "done"
Houses Result
display: houses based on the filters/search options applied by user
Click on sidebar icon
Sidebar (not logged in)
display: login, view grants, view agent listings
Click on a house
[login == TRUE]
House Information
display: house information, favorites icon, chat icon, home calculator, agent information
Login
display: username, password, domain, login/sign up button
user logs out
Home Calculator
display: details of buyer 1, details of buyer 2
[domain == nonagent]
[domain == agent]
login information verified
Inbox
display: list of chats, edit button
Main Menu (Non-Agent)
display: advanced filters, search bar, sidebar icon, recent listings, toggle button(rent/buy), AI button
Main Menu (Agent)
display: advanced filters, search bar, sidebar icon, recent listings, toggle button(rent/buy), AR button
enter details of buyer
Calculate
entry: confirm buyer details
do: calculate finances based on buyer details
display: loading icon
click on a chat
click inbox icon
Chat
display: sender or receiver information
do: send and receive messages
Sidebar
display: view grants, view agent listings, inbox, settings
Sidebar
display: my listings, inbox, settings
Show Results
display: final breakdown of finances, risk assessment
click on view grants
View Grants
display: different grants, check eligibility button
[login == TRUE]
click on check eligibility button
Check Eligibilty
entry: confirm buyer details
do: calculate eligibility based on buyer details
display: loading icon
Show Results
display: eligible grants based on buyer details

Initial Dialog map



System architecture


Control Object
1
AgentMgr
+ getAgentList() : ArrayList<Agents>
+ viewSettings() : void
+ viewMyListings() : void
+ inbox(userID:int) : void
+ addMyListings() : void
+ updateMyListings(houseObj : House) : void
+ deleteMyListings(houseObj : House) : void
+getListOfHouses(keyword: string,price:int, room:int, yearsOfLease:int, stories:int, priInc:boolean, secInc:boolean): ArrayList<House>
1
1
1
1
1
1
Entity Object
Agent
-houselisting: House
-Company: string
- loggedIn : boolean
+createAgent(UserId : String, Username : String, Password : String ,Name : String, Email : String Number : String)
+getUserId() : String
+getUsername() : String
+getPassword() : String
+getName() : String
+getEmail() : String
+getNumber() : String
+setUserId(userId : String)
+setUsername(userName : String)
+setPassword(pass : String)
+setName(name : String)
+setEmail(email : String)
+setNumber(number : String)
+addHouseListing(houseObj : House)
*
Boundary Object
<<interface>>
Agent
1
+ callViewMyListings()
+ callViewSettings()
+ viewInbox()
+sort(price:int, room:int, yearsOfLease:int, stories:int,priInc:boolean,secInc:boolean)
+FurnitureSearch()
+keySearch(keyword: string)
Boundary Object
<<interface>>
Main
+ main(String[] args): void
+ run(): void
+ login():void
1
1
Control Object
LoginMgr
- AgentList : ArrayList<Agent>
- NonAgentList : ArrayList <NonAgent>
# validate() : void
1
1
1
1
Control Object
HouseMgr
+addOne(house: House): boolean
+deleteOne(house: House): boolean
+getAll(): List<House>
1
1
1
Control Object
InboxMgr
+ sendMsg(houseObj : House, agentObj : Agent, nonAgentObj : NonAgent, messages : string)
+ deleteMsg(msgID : int)
+ viewInbox(userID:int) : ArrayList<String>
+ viewChat() : Chat
1
1
Boundary Object
<<External>>
<<interface>>
TensorFlow Lite API
Entity Object
Chat
-ArrayList: string
-Agent agentObj
-NonAgent nonAgentObj
-House houseObj
+printMsg():void
*
Entity Object
*
House
-Name: string
-Address: string
-noOfRooms: int
-price: int
-leasePeriod: int
-storeys: int
-agentinfo: string
-favourite: boolean
+favourited(): boolean
+getName(): String
+getAddress(): String
+getRooms(): int
+getPrice():int
+getLease():int
+getStoreys():int
+getAgentInfo():String
+addHouse(name : string, add : string ,numRooms : int ,price : int, lease : int ,storey : int ,agentinfo : String)
+setName(name : String)
+setAddress(add : String)
+setRooms(numRooms : int)
+setPrice(price : int)
+setLease(lease : int)
+setStoreys(storey : int)
+setAgentInfo(agentinfo : String)
1
Control Object
1
NonAgentMgr
+ homeCalc(income : float,age : float,loggedIn : boolean)
+getNonAgentList() : ArrayList<NonAgents>
+viewGrants()
+ checkEligibility(nonAgentUser: NonAgent)
+ viewAgentListings()
+ favouriteHouse(houseObj : House, loggedIn : boolean)
+ getSecondBuyerInput(): List
+ viewSettings(loggedIn : boolean)
+ inbox(userID:int)
+ viewSelectedHouse(houseObj: House)
+getListOfHouses(keyword: string,price:int, room:int, yearsOfLease:int, stories:int, priInc:boolean, secInc:boolean): ArrayList<House>
1
1
1
1
Boundary Object
1
<<interface>>
Non-agent
1
+ callHomeCalc(income : float, age : int, loggedIn : boolean)
+ favouritingHouse(houseObj:House)
+ viewAgentListings()
+ callViewGrants()
+ callCheckEligibility(loggedIn:boolean, nonAgentUser: NonAgent)
+ callViewSettings()
+ viewInbox()
+callViewSelectedHouse()
+sort(price:int, room:int, yearsOfLease:int, stories:int,priInc:boolean,secInc:boolean)
+FurnitureSearch()
+keySearch(keyword: string)
1
Entity Object
NonAgents
-favouritedHousesList: House
-age: int
-salary: double
-secondbuyer_name: string
-secondbuyer_salary: double
-secondbuyer_age: int
-relationship: String
-loggedIn: boolean
+createNonAgent(UserId, : String ,Username : String, Password, :String, Name : String, Email : String,Number : String, salary : double, age : int)
+getUserId() : String
+getUsername() : String
+getPassword() : String
+getName() : String
+getSalary() : double
+getAge() : int
+getRelationship() : String
+getSecondBuyerAge() : int
+getSecondBuyerSalary() : double
+getEmail() : String
+getNumber() : String
+setUserId(userId : String)
+setUsername(userName : String)
+setPassword(pass : String)
+setName(name : String)
+setSalary(salary : double)
+setAge(age : int)
+setRelationship(relation : String)
+setSecondBuyerAge(sb_age : int)
+setSecondBuyerSalary(sb_salary : double)
+setEmail(email : String)
+setNumber(number : String)
*
1
1

Class diagram