

1.0 Project Introduction

1.1 Purpose

Our group, Underdog, intends to create a web application directed at both landlords and tenants who will be able to list out flats that are for resale or rent out respectively. Our web application will serve as a platform for landlords who are interested in reselling or renting out their properties directly to interested tenants.

1.2 Intended Audience

Our intended audience for the application is those who are seeking a platform to buy, rent or sell properties.

2.0 Documentation of Functional and Non-functional requirements

2.1 Functional Requirements

1. Account registration

- 1.1. Users must be able to select options to register as a Landlord or Tenant.
- 1.2. Users must be able to create a user account.
 - 1.2.1. Account must have a username.
 - 1.2.1.1. Username must be between 8 to 30 characters long.
 - 1.2.1.2. Username must not be a repeat of an already existing user in the database. In other words, it must be unique.
 - 1.2.2. Users must provide the following details during registration:
 - 1.2.2.1. A valid email address.
 - 1.2.2.2. A password that includes 1 uppercase letter, 1 lowercase letter, 1 number, 1 special character and must be at least 8 characters long.
 - 1.2.2.3. User's name.
 - 1.2.2.4. A valid user phone number.

2. User Account login

- 2.1. Users must be able to select options to login as a Landlord or Tenant.
- 2.2. Users must be able to login into the account that they created using the credentials that they have keyed in during account registration.
 - 2.2.1. The system must validate the username and password for account security.
 - 2.2.2. System must display an error message if the login credentials are wrong.
 - 2.2.3. The system must allow existing users who forgot their password to reset their password.
 - 2.2.3.1. The system must send an email to users to reset the password.
 - 2.2.3.2. The system must prompt the user to change passwords if the new password has been used before.

3. Admin Account login

- 3.1. Admins must be able to login as an Admin at a separate URL from the Landlords and Tenants.
- 3.2. Admins must be able to login into the account that they created using the credentials that they have keyed in during account registration.
 - 3.2.1. The system must validate the username and password for account security.
 - 3.2.2. System must display an error message if the login credentials are wrong.
 - 3.2.3. The system must allow existing users who forgot their password to reset their password.
 - 3.2.3.1. The system must send an email to Admins to reset the password.
 - 3.2.3.2. The system must prompt the Admins to change passwords if the new password has been used before.
- 3.3. Admins must be able to use public API to get Master data.
- 3.4. Admins must be able to update Master data into the database.
- 3.5. Admins must be able to refresh Master data to update details on the main webpage.

4. Personal user details
 - 4.1. Users must fill up the compulsory personal details.
 - 4.2. Users must be able to edit their personal details.
 - 4.3. Users must have a profile picture. If a picture is not provided by the user, a default profile picture will be provided.
 - 4.4. Users must be able to view their profile details.
5. Property details
 - 5.1. The system must allow the user registered as Landlords to provide details of the property.
 - 5.1.1. Details of the property must consist of the following:
 - 5.1.1.1. For Sale/For rental
 - 5.1.1.2. Address
 - 5.1.1.2.1. Address must be valid.
 - 5.1.1.3. Unit number
 - 5.1.1.3.1. Unit Number must be valid.
 - 5.1.1.4. Price
 - 5.1.1.5. Property Type
 - 5.1.1.6. Property Age
 - 5.1.1.7. Floor Size
 - 5.1.1.8. Furnishing
 - 5.1.1.9. Bedrooms
 - 5.1.1.10. PSF (Per Square Foot)
 - 5.1.1.11. Tenure
 - 5.1.1.12. Amenities
 - 5.2. The system must allow the users the choice to add additional details
 - 5.2.1. The system must allow the additional details to be optional.
 - 5.2.2. Additional details must consist of the following:
 - 5.2.2.1. Wheelchair accessible
 - 5.2.2.2. Proximity to MRT station
 - 5.3. The system must allow the user the option to edit the details of each individual property at any time in the future.
 6. Property filtering
 - 6.1. Tenants must be able to fill in their housing preferences.
 - 6.2. Tenants must be able to select the housing based on a list of filters.
 - 6.2.1. The system must also allow users to only view property that are:
 - 6.2.1.1. For sale
 - 6.2.1.2. For rental
 - 6.2.1.2.1. Users must be able to state the specific dates for their rental.
 - 6.2.2. The system must provide the users with the following as sorting criteria:
 - 6.2.2.1. Price
 - 6.2.2.1.1. Users must be able to provide a price range given as an option.
 - 6.2.2.2. Property type
 - 6.2.2.2.1. Users must be able to provide specified property type (ie. HDB, Condo, Landed) given as an option.
 - 6.2.2.3. Property Age

- 6.2.2.3.1. Users must be able to provide a Property Age range given as an option.
 - 6.2.2.4. Floor Size
 - 6.2.2.4.1. Users must be able to provide a floor size range given as an option.
 - 6.2.2.5. Furnishing
 - 6.2.2.5.1. Users must be able to provide specified furnishing (ie. Unfurnished, Partially Furnished, Fully Furnished) given as an option.
 - 6.2.2.6. PSF (Per Square Foot)
 - 6.2.2.6.1. Users must be able to provide a PSF range given as an option.
 - 6.2.2.7. Tenure
 - 6.2.2.7.1. Users must be able to provide specified tenure (ie. Freehold, 99-year Leasehold etc) given as an option.
 - 6.2.2.8. Location
 - 6.2.2.8.1. Users must be able to select the location by district.
 - 6.2.2.9. Amenities
 - 6.2.2.9.1. Users must be able to provide specified amenities (ie. WiFi, Washing machine, Kitchen etc) given as an option.
- 7. Property bookmarking
 - 7.1. Users must be able to save properties in the system under their account.
 - 7.2. Users must be able to remove properties in the system under their account.
 - 7.3. Users must be able to view the list of saved properties in the system under their account.
- 8. Property Rating
 - 8.1. After using the property for sale or for rental, Tenants must be able to give a rating and review of the housing they accommodate to the Landlords.
 - 8.1.1. The ratings will be of the format of a 5 point rating scale with additional review given as an option.
 - 8.1.2. The system must update any ratings and reviews in the Landlord's profile.
 - 8.2. After using the property for sale or for rental, Landlords must be able to give a rating and review of the sale to the Tenants.
 - 8.2.1. The ratings will be of the format of a 5 point rating scale with additional review given as an option.
 - 8.2.2. The system must update any ratings and reviews in the Tenant's profile.
- 9. Contact system
 - 9.1. Tenants must be able to contact the Landlords from the Property viewing Page.
- 10. Forum Page
 - 10.1. Users must be able to post guides related to housing content.
 - 10.1.1. Users must be able to report post under different categories (ie. Irrelevant, Advertisements, Defamatory etc).

2.2 Non-functional requirements

- 1. Usability

- 1.1. Error messages with proper indication of the respective issue must be displayed clearly to help the user.
 - 1.2. Help messages must be displayed in the local language according to the user's locale.
2. Reliability
 - 2.1. The system must be able to sustain with minimal crashing.
 - 2.2. Upon Server reboot, full website functionality must resume within 15 minutes.
 - 2.3. The website must have an uptime of 99%.
3. Performance
 - 3.1. The website must be able to load within 3 seconds.
 - 3.2. The registration and login page must be able to load within 2 seconds.
 - 3.3. The validation email for registration must be able to be received within 1 minute.
 - 3.4. The resetting of the password must be able to be updated in 3 seconds.
 - 3.5. Each return search result must be able to load within 10 seconds.
 - 3.6. The listing of properties by Landlords must be updated on the website within 10 seconds.
4. Supportability
 - 4.1. The database must be replaceable with any commercial product supporting standard SQL queries.
 - 4.2. The web application must be able to run on Google Chrome, Safari, Firefox browsers.

3.0 Data Dictionary

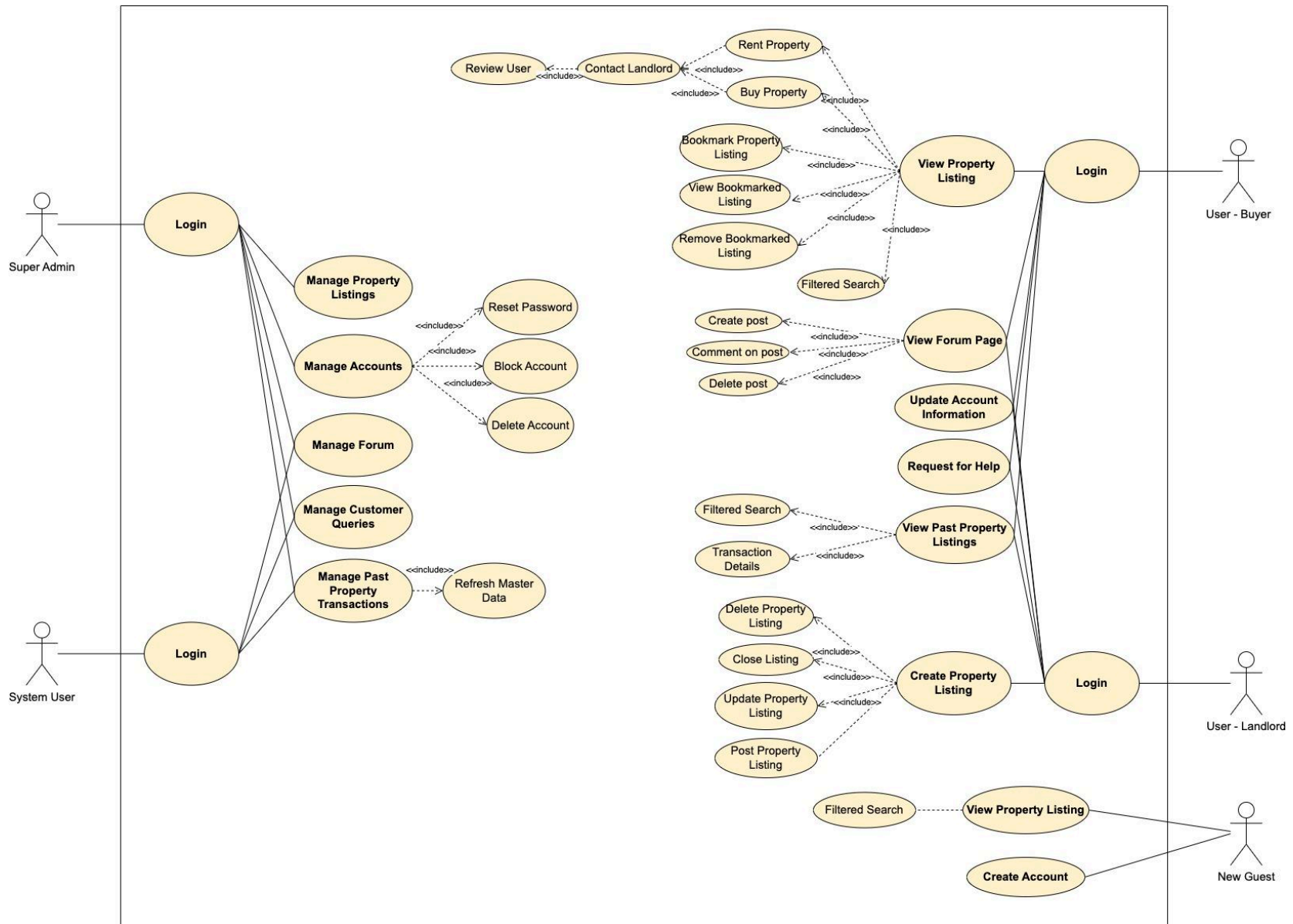
Term	Definition
Register	Process which allows users to make a new account on the app. This can be done by creating a username and providing other important details such as an email account.
Account	The data of the person in the system which consists of the personal particulars e.g. name, username, email.
User	A person who has created an account and will be using the app.
APIs	Application Programming Interface is a software intermediary that allows two applications to talk to each other.
Login	Feature which allows users to login to an existing account in the system. This can only be done after the user has registered their account.
Password	String of characters that the user needs to create during registration to ensure security and allows them to login into the system. Password creation may have certain requirements to fulfill.
For Sale/For Rental	A property being made available for purchase or for lease/rent, respectively. The property owner offers the property for sale or rental and interested parties can make an inquiry or offer.
Property Type	<p>The classification of a real estate property based on its use, design, and construction. Some common types of properties include:</p> <ol style="list-style-type: none">1. HDB (Housing and Development Board) flats - publicly subsidized housing units2. Private apartments/condominiums - privately owned units3. Landed properties - houses with land and private outdoor space4. Executive Condominiums - hybrid of

	public and private housing
Property Age	The age of a real estate property, measured from the date of completion and/or occupation.
PSF (Per Square Foot)	Unit measurement to express the price of real estate properties.
Tenure	<p>The type of ownership or use rights that a Landlord has over a property. The main types of tenure are:</p> <ol style="list-style-type: none"> 1. Freehold - The owner has permanent and absolute ownership of the property, with no restrictions on the length of time they can hold onto it. 2. Leasehold - The owner has the right to use the property for a specified period of time, typically 99 years or less. 3. Government lease - A leasehold property owned by the government, typically for public housing units.
Location	The general region of the property is limited to Singapore geography. Eg. Boon Lay, Ang Mo Kio, Tampines etc
Amenities	<p>The amenities is a collection of the following buildings/facilities:</p> <ul style="list-style-type: none"> ● MRT Stations ● Schools ● Bus-stops ● Shopping Malls ● Parks ● WiFi etc
Filter	Search criteria which allows users to set on the UI while searching for an apartment.

Bookmark	Feature that allows users to save and quickly access the contents marked as important and can be categorized into folders
Rating	A classification or ranking of something based on the comparative assessment of their performance. In this case, it determines how well the landlord or tenant worked during the period of housing rent.
Review	Allows the user to give a short write-up of their experience and any feedback.
Master Data	Collection of data for the sales made through the use of the website and data retrieved from public API.

4.0 Initial Use Case Model

4.1 Use case diagram



4.2 Use Case Description

Use Case ID:	A0001		
Use Case Name:	Create Account		
Created By:		Last Update By:	Hemanth
Date Created:	28/01/2023	Date Last Updated:	06/02/2023

Actor:	User
Description:	The user creates an account in our system database. To create an account, users must provide a username, valid email address, password and personal details such as their name and phone number.
Precondition:	<ol style="list-style-type: none">1. User's device should have access to the internet.2. User's email should be valid and not have been used before.3. Username specified by user must not be taken.4. Password must adhere to requirements; at least 1 uppercase letter, 1 lowercase letter, 1 number, 1 special character and must be at least 8 characters long
Postcondition:	<ol style="list-style-type: none">1. User successfully signs up for an account2. System sends confirmation link to user's email account3. System displays main page
Priority:	Low
Frequency of use:	Once per lifetime
Flow of events:	<ol style="list-style-type: none">1. User navigates to the "Sign Up" page2. User selects between options of landlord, tenant or Landlord.3. User inputs the required information: Username, Password and Email Address4. System validates the information.5. Upon validation, a notification detailing "New account is successfully created. Please sign in" will be displayed.6. System updates account information in database.
Alternative Flows:	AF-S1: User account already exists: Phone number/ Email taken. <ol style="list-style-type: none">1. System displays "User account already exists"2. System prompts users to create an account or log in.3. If you create an account, then continue from main flow step 2.4. If log in, then system directs to log in page to continue from

	<p>Login (A2007)</p> <p>AF-S2: Username already taken</p> <ol style="list-style-type: none"> 1. System displays “Username already taken” 2. System prompts user to input a different username. 3. Continue from main flow step 2. <p>AF-S3: Password does not meet requirements.</p> <ol style="list-style-type: none"> 1. System displays “Please input a password of at least 1 special character, mixed case and at least 8 characters long” message. 2. Continue from main flow step 2.
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	A0002		
Use Case Name:	User Login (Landlord/Tenant)		
Created By:		Last Update By:	Hemanth
Date Created:	24/01/2023	Date Last Updated:	07/02/2023

Actor:	User
Description:	User logs into account. To login into the account, users must input their username and the corresponding password. There will be a maximum of 3 attempts, after which, the user will have to wait for 5 minutes before another attempt.
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. User should have an existing account in the application database
Postcondition:	The user successfully logs in and the system displays the main page.
Priority:	Low
Frequency of use:	Depends on usage
Flow of events:	<ol style="list-style-type: none"> 1. User enters username, password on the login page and submits. 2. System checks for input validity 3. If the username and password are verified, the user will be able to log in successfully and is directed to the main page.
Alternative Flows:	<p>AF-S1: Login credentials are wrong</p> <ol style="list-style-type: none"> 1. System displays an error message "Invalid username or password". 2. System goes back to step 1 in the 'Flow of events'. <p>AF-S2: Maximum 3 attempts exceeded</p> <ol style="list-style-type: none"> 1. System displays "Maximum attempts exceeded, try again in 5 minutes" message 2. System locks the user account for 5 minutes
Exceptions:	
Includes:	Verifying that account details are correct.
Special Requirements:	
Assumptions:	

Notes and Issues:	
-------------------	--

Use Case ID:	A0003		
Use Case Name:	Forget Password		
Created By:	Hemanth	Last Update By:	
Date Created:	07/02/2023	Date Last Updated:	

Actor:	User
Description:	The user forgets their password to an existing account
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. User must have an existing account in the database
Postcondition:	<ol style="list-style-type: none"> 1. User successfully changes his/her password
Priority:	Low
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user clicks on the "Forgot Password" button 2. The user will be prompted to enter his/her existing email address linked to his/her account 3. The system will send a reset password link to the user's registered email 4. The user will be prompted to enter a new password of required specifications 5. Following that, a "Password changed successfully" notification will be shown 6. User will be redirected to log in
Alternative Flows:	<p>AF-S1: The email address does not exist</p> <ol style="list-style-type: none"> 1. The system will display an error message <ol style="list-style-type: none"> a. The message is "The address added is not a valid address! Please try again" 2. The system will go back to step 2 in the Flow of events <p>AF-S2: New password is not valid</p> <ol style="list-style-type: none"> 1. If the new password does not meet the minimum criteria, the website will display an error message <ol style="list-style-type: none"> a. The message is "The password does not meet the requirements. Please try again" 2. The system will go back to step 4 in the Flow of events
Exceptions:	

Includes:	
Special Requirements:	
Assumptions:	The users have an existing email account from any email domain.
Notes and Issues:	

Use Case ID:	A0004		
Use Case Name:	Update Account Information		
Created By:		Last Update By:	Hemanth
Date Created:	28/01/2023	Date Last Updated:	07/02/2023

Actor:	User
Description:	User requests to update account information such as personal details and profile picture
Precondition:	<ol style="list-style-type: none"> 1. Users must have an existing account and must be logged into account to update it. 2. User's device should have access to the internet and GPS should be turned on.
Postcondition:	<ol style="list-style-type: none"> 1. User has successfully updated his/her profile
Priority:	Low
Frequency of use:	3
Flow of events:	<ol style="list-style-type: none"> 1. The user navigates to the "Profile" page. 2. The user clicks on Edit Profile 3. He will then be prompted with a new interface with current profile data 4. The user will be able to edit current profile data (profile photo, name, email, bio, password) within the new interface provided 5. The user selects "Update". 6. The profile is updated with the latest information
Alternative Flows:	AF-S1: Profile Page is exited without saving <ol style="list-style-type: none"> 1. No changes will be made to current profile data
Exceptions:	
Includes:	
Special Requirements:	The user must have an existing account.
Assumptions:	
Notes and Issues:	

Use Case ID:	A0005		
Use Case Name:	Create Property Listing		
Created By:		Last Update By:	Hemanth
Date Created:	28/01/2023	Date Last Updated:	07/02/2023

Actor:	User (Landlord)
Description:	When a user of type Landlord logs on to the system, he or she is able to create a Property Listing. To do so, the user needs to fill up the required details pertaining to the property itself and relevant pictures. Once the information is inputted, then the user can successfully list the property.
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. User must be logged in as a Landlord to create a Property Listing 3. User's account must have been verified [Email + Phone Number] 4. The property listing should not already exist in the database
Postcondition:	<ol style="list-style-type: none"> 1. The property listing is created 2. System creates a chat box for that Property Listing for Landlords and Tenants or Landlords to communicate
Priority:	Low
Frequency of use:	5
Flow of events:	<ol style="list-style-type: none"> 1. The user clicks on "Create New Listing" 2. The users enter details of the property: <ol style="list-style-type: none"> a. Sale / Rental b. Address c. Unit number d. Price e. Property Type f. Property age g. Floor Size h. Furnishing i. Number of Bedrooms j. Price per square foot (if Sale) k. Tenure 3. User uploads all property photographs and relevant documents 4. The user clicks on the "Complete Listing" button

	5. The System checks the details that the users has written for errors
Alternative Flows:	ERR1: The address is invalid <ol style="list-style-type: none"> 1. The system will display an error message <ol style="list-style-type: none"> a. The message is “The address added is not a valid address! Please try again” 2. The system will go back to step 2 in the Flow of events
Exceptions:	
Includes:	<ol style="list-style-type: none"> 1. Check Status 2. Post Property Listing 3. Delete Property Listing 4. Update Property Listing 5. Close Listing
Special Requirements:	
Assumptions:	The user must have an existing account and of type landlord.
Notes and Issues:	

Use Case ID:	A0006		
Use Case Name:	View Property Listing		
Created By:		Last Update By:	Hemanth
Date Created:	25/01/2023	Date Last Updated:	07/02/2023

Actor:	User (Tenant / Landlord)
Description:	When a user of type Tenant or Landlord logs on to the system, he or she is able to view several Property Listings. From there they can also look at specific properties based on the filtered search. Should they indicate interest in a particular property the user can bookmark it or proceed to chat with the Landlord
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. Property Listing should be in the database system 3. User must be logged in as a Tenant or Landlord to view a Property Listing 4. User's account must have been verified [Email + Phone Number]
Postcondition:	<ol style="list-style-type: none"> 1. Users can view the Property Listings 2. Users can utilize the filtered search
Priority:	
Frequency of use:	5
Flow of events:	<ol style="list-style-type: none"> 1. Users land on the main page of the system 2. Following which, users can start viewing property listings. 3. Users have many options to choose from such as bookmark property listings, filtered search and contact landlord 4. System carries out what the user clicks.
Alternative Flows:	
Exceptions:	
Includes:	<ol style="list-style-type: none"> 1. Bookmark Listing 2. Delete Bookmarked Listing 3. View Bookmarked Listing 4. Rent Property 5. Buy Property 6. Filtered Search

Special Requirements:	
Assumptions:	The user must have an existing account and of type Landlord / tenant.
Notes and Issues:	

Use Case ID:	A0007		
Use Case Name:	Post Property Listing		
Created By:		Last Update By:	Hemanth
Date Created:	28/01/2023	Date Last Updated:	07/02/2023

Actor:	User (Landlord)
Description:	Once the user has created a property listing, it is in the “draft” status. Once the user has filled the property listing with all the relevant details and is satisfied, the user requests for the property listing to be posted.
Precondition:	The property listing is not posted and it is in “draft” status. View Property listings cannot display to the user the listing.
Postcondition:	The property listing is posted. View Property listings can display to the user the listing. Status of the listing is changed to “Posted”.
Priority:	Low
Frequency of use:	5
Flow of events:	<ol style="list-style-type: none"> 1. The user clicks on the post listing button 2. The system displays “Listing has been posted”
Alternative Flows:	
Exceptions:	<p>AF-S1: The database is down and thus the posting cannot be done</p> <ol style="list-style-type: none"> 1. The system will display an error message “error unable to post please try again later” 2. The system will return to step 1 of the Flow of events <p>AF-S2: The posting has been posted already</p> <ol style="list-style-type: none"> 1. The system will display an error message “the post has already been posted
Includes:	
Special Requirements:	
Assumptions:	<ol style="list-style-type: none"> 1. The Listing has been created 2. The Listing has not been posted
Notes and Issues:	

Use Case ID:	A0008		
Use Case Name:	Update Property Listing		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	07/02/2023

Actor:	User (Landlord)
Description:	User wants to update the information about the Property Listing he or she has already created.
Precondition:	<ol style="list-style-type: none"> 1. Users must have an existing account and must be logged into account to update their property listing. 2. User's device should have access to the internet and GPS should be turned on. 3. Users must already create a property listing saved in the database.
Postcondition:	<ol style="list-style-type: none"> 1. The user is logged into their account 2. The listing still exists 3. The listing posted status remains unchanged from before 4. The listing details are changed
Priority:	Low
Frequency of use:	5
Flow of events:	<ol style="list-style-type: none"> 1. User heads to the listing(s) he or she created 2. User clicks on the "Update Property Listing" button 3. The system displays the details of the property for the user to make changes 4. The user enters the changes that they wish to make 5. The user clicks on the "Save Changes" button.
Alternative Flows:	
Exceptions:	<p>AF-S1: The address is invalid</p> <ol style="list-style-type: none"> 1. The system will display an error message <ol style="list-style-type: none"> a. The message is "The address added is not a valid address! Please try again" 2. The system will go back to step 2 in the Flow of events
Includes:	
Special Requirements:	

Assumptions:	
Notes and Issues:	

Use Case ID:	A0009		
Use Case Name:	Delete Property Listing		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	07/02/2023

Actor:	User (Landlord)
Description:	Users are able to removes any property listing they have created disregarding the status of the listing
Precondition:	<ol style="list-style-type: none"> 1. Users must have an existing account and must be logged into account to update their property listing. 2. User's device should have access to the internet and GPS should be turned on. 3. Users must already create a property listing saved in the database.
Postcondition:	<ol style="list-style-type: none"> 1. The listing no longer exists in the database
Priority:	
Frequency of use:	Whenever the user intends to remove their listing
Flow of events:	<ol style="list-style-type: none"> 1. The user clicks on the listing they want to delete from the listing(s) they have already created 2. The user than clicks on "Delete Listing" 3. System will prompt users with a confirmation notification to make sure they are sure of their choice. 4. User selects "Yes" to deleting. 5. The system than displays to the user "Listing successfully deleted" 6. The system than returns to the main menu
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	User is connected to the database
Notes and Issues:	

Use Case ID:	A0010		
Use Case Name:	Close Property Listing		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	08/02/2023

Actor:	User (Landlord)
Description:	After the landlord user has successfully managed to sell or rent his or her property, the user can close the property listing. What this will do is remove the listing from existing property listings but not from the master database of all property listings.
Precondition:	<ol style="list-style-type: none"> 1. Users must have an existing account and must be logged into account to update their property listing. 2. User's device should have access to the internet and GPS should be turned on. 3. Users must already create a property listing saved in the database. 4. User's Property Listing is posted and status is "Sold".
Postcondition:	<ol style="list-style-type: none"> 1. The Property Listing is removed from view of potential tenants and Landlords, removed from the available Property Listings 2. The Property Listing is closed for sale / rent. 3. The Property Listing can be viewed only by the User who created it
Priority:	
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user heads to the page of their property listing. 2. The user clicks on the "Close Property Listing" 3. The system will display "Confirm closure of the listing?" 4. If the the user clicks on Yes <ol style="list-style-type: none"> a. The system will label the listing as closed 5. The user clicks on No <ol style="list-style-type: none"> a. The system will return to the property listing
Alternative Flows:	<p>AF-S1: The listing is already closed</p> <ol style="list-style-type: none"> 1. The system will display an error message "the listing is already closed. Please try again" 2. The user clicks on "okay" 3. The system than returns to the property listing menu
Exceptions:	

Includes:	
Special Requirements:	
Assumptions:	User is logged into their account and the user is a landlord
Notes and Issues:	

Use Case ID:	A0011		
Use Case Name:	Bookmark Property Listing		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	08/02/2023

Actor:	User (Landlord / Tenant)
Description:	Users can place a digital bookmark on the Property Listings that they are interested in and want to have a look at again quickly.
Precondition:	<ol style="list-style-type: none"> 1. Users must have an existing account and must be logged into account. 2. User's device should have access to the internet and GPS should be turned on. 3. Users must be on the page of available Property Listings page.
Postcondition:	<ol style="list-style-type: none"> 1. The Property Listing is bookmarked by the User 2. User can view this Listing under his / her bookmarked Property List
Priority:	Low
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. User is on the available Property Listings page or on a specific Property Listing page. 2. The user clicks on the "Bookmark Property Listing" button 3. The system bookmarks the Property Listing
Alternative Flows:	AF-S1: The property listing is not in the database <ol style="list-style-type: none"> 1. The system displays an error message: "the property does not exist! Please refresh the webpage!"
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	User is logged in as a Landlord or tenant
Notes and Issues:	

Use Case ID:	A0012		
Use Case Name:	Remove Bookmarked Listing		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	08/02/2023

Actor:	User (Landlord / Tenant)
Description:	Users can remove a digital bookmark on the Property Listings that they have already bookmarked.
Precondition:	<ol style="list-style-type: none"> 1. Users must have an existing account and must be logged into account. 2. User's device should have access to the internet and GPS should be turned on. 3. Users must have already bookmarked the Property Listing.
Postcondition:	<ol style="list-style-type: none"> 1. The Property Listing is removed from the User's bookmarked property listings
Priority:	Low
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user views the list of bookmarked property listings 2. The user selects the property listing he or she wishes to unlist and clicks on the "Remove Bookmark" button 3. System will display "Property Listing removed from Bookmarks successfully".
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	A0013		
Use Case Name:	View Bookmarked Listings		
Created By:		Last Update By:	Hemanth
Date Created:	19/1/23	Date Last Updated:	08/02/2023

Actor:	User (Landlord / Tenant)
Description:	Allows the user to view all of their bookmarked listings on a page
Precondition:	<ol style="list-style-type: none"> 1. Users must have an existing account and must be logged into account. 2. User's device should have access to the internet and GPS should be turned on. 3. Users must have already bookmarked Property Listing(s).
Postcondition:	<ol style="list-style-type: none"> 1. The system will display only the entries that the user has bookmarked
Priority:	Low
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user clicks on "View all Bookmarked Listings" 2. The system displays only the entries that the user has bookmarked
Alternative Flows:	<p>AF-S1: The user has not bookmarked any listings at all</p> <ol style="list-style-type: none"> 1. The system will display an error message "There are no existing bookmarks!" 2. The user clicks on "Okay" as an acknowledgement 3. The system carries on to displays all available entries <p>AF-S2: The user has not logged in</p> <ol style="list-style-type: none"> 1. The system will display an error message "You must log in to use this feature!" 2. The user clicks "Okay" 3. The system will redirect to the login page
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	

Notes and Issues:	
-------------------	--

Use Case ID:	A0014		
Use Case Name:	Property Listing Filtered Search		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	22/01/2023

Actor:	User (Landlord / Tenant)
Description:	Allows the user to view the entire list of available Property Listing entries based on their specific filters chosen
Precondition:	<ol style="list-style-type: none"> 1. Users must have an existing account and must be logged into account. 2. User's device should have access to the internet and GPS should be turned on.
Postcondition:	<ol style="list-style-type: none"> 1. The system will display all the entries that match the user's chosen filters.
Priority:	
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user goes to the page "View all Listings" 2. The user clicks on the "Filter properties" 3. The system displays several options that the user can choose from 4. The user selects his or her filters and presses "Confirm selection" 5. The system only displays the entries that match the criteria
Alternative Flows:	<p>AF-S1: The criteria specified returned no entries</p> <ol style="list-style-type: none"> 1. The system displays an error message "Nothing was found! Please try again" 2. The user clicks the okay button 3. The system displays all entries 4. The flow of events repeats from step 2 <p>AF-S2: The user isn't logged in</p> <ol style="list-style-type: none"> 1. The system will display an error message "You must log in to use this feature!" 2. The user clicks "Okay" 3. The system will display the login page
Exceptions:	
Includes:	

Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	A0015		
Use Case Name:	Rent Property		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	11/02/2023

Actor:	User
Description:	User of type tenant goes into a property listing and selects to rent the property. This will notify the landlord creating a chat between the user and the landlord where they can negotiate an agreement and the terms.
Precondition:	<ol style="list-style-type: none"> 1. Users must have an existing account and must be logged into account. 2. User's device should have access to the internet and GPS should be turned on. 3. User account must be verified
Postcondition:	<ol style="list-style-type: none"> 1. Landlord of the Property Listing is notified about the offer made by the user 2. The user is in negotiation with the landlord via the chat function
Priority:	High
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user clicks on the "Rent Property" button on the Property Listing 2. The system displays "Request made successfully. Landlord will be notified" 3. The system creates a chat between the user and the landlord. 4. User will wait for Landlord's response
Alternative Flows:	<p>AF-S1: The user is not logged in</p> <ol style="list-style-type: none"> 1. The system will display an error message "You must log in to use this feature!" 2. The user clicks "okay" 3. The system will display the login page <p>AF-S2: The user is not verified</p> <ol style="list-style-type: none"> 1. The system will display an error message "You must be verified to use this feature!" 2. The user clicks "Okay" 3. The system will display the profile page where user can update account information

Exceptions:	
Includes:	<ol style="list-style-type: none"> 1. Review Property 2. Landlord notified
Special Requirements:	
Assumptions:	Status of Property Listing is Open
Notes and Issues:	

Use Case ID:	A0016		
Use Case Name:	Buy Property		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	11/02/2023

Actor:	User (Landlord)
Description:	Users can move forward on the Property Listing he or she is interested in and make an offer. This will notify the landlord creating a chat between the user and the landlord where they can negotiate an agreement and the terms.
Precondition:	<ol style="list-style-type: none"> 1. Users must have an existing account and must be logged into account. 2. User's device should have access to the internet and GPS should be turned on. 3. User account must be verified
Postcondition:	<ol style="list-style-type: none"> 1. Landlord of the Property Listing is notified about the offer made by the user 2. The user is in negotiation with the landlord via the chat function
Priority:	High
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user clicks on the "Buy Property" button on the Property Listing 2. The system displays "Request made successfully. Landlord will be notified" 3. The system creates a chat between the user and the landlord. 4. User will wait for Landlord's response
Alternative Flows:	<p>AF-S1: The user is not logged in</p> <ol style="list-style-type: none"> 1. The system will display an error message "You must log in to use this feature!" 2. The user clicks "okay" 3. The system will display the login page <p>AF-S2: The user is not verified</p> <ol style="list-style-type: none"> 1. The system will display an error message "You must be verified to use this feature!"

	<ol style="list-style-type: none"> 2. The user clicks “Okay” 3. The system will display the profile page where user can update account information
Exceptions:	
Includes:	<ol style="list-style-type: none"> 1. Contact Landlord 2. Review property
Special Requirements:	
Assumptions:	Status of Property Listing is Open
Notes and Issues:	

Use Case ID:	A0017		
Use Case Name:	Review User		
Created By:		Last Update By:	Hemanth
Date Created:	28/01/2023	Date Last Updated:	11/02/2023

Actor:	User
Description:	Our Property Marketplace enables transactions between tenants and landlords. Following a successful transaction, both parties of a transaction can leave a review to the other user. Users can accumulate reviews and make themselves more appealing for future transactions.
Precondition:	<ol style="list-style-type: none"> 1. Users must have an existing account and must be logged into account. 2. User's device should have access to the internet and GPS should be turned on. 3. User must have completed a transaction either as a Landlord or Tenant
Postcondition:	<ol style="list-style-type: none"> 1. The other party of the transaction is notified about the review 2. The review will appear under their profile page
Priority:	
Frequency of use:	One rating for each use of services
Flow of events:	<ol style="list-style-type: none"> 1. Following the successful transaction made, the chat function will request each party to leave a review for the other party 2. User gives a rating based on a 5 point scale. 3. User also types in additional remarks in a given textbox 4. Once done, user clicks on "Submit Review" button 5. Rating is saved in the database
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	Both tenants/buyers and landlords can rate.
Notes and Issues:	

Use Case ID:	A0018		
Use Case Name:	View Master Data		
Created By:	Hemanth	Last Update By:	
Date Created:	11/02/2023	Date Last Updated:	

Actor:	User
Description:	Users are able to view all the past property transactions on this a page. Users can use this past transactions as reference when creating a new property listing if they are a landlord or when making an offer for a property listing if they are buyer or tenant
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. Users must have logged on to an account
Postcondition:	<ol style="list-style-type: none"> 1. The Master Data showcases all past property transactions and their details
Priority:	Low
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user heads to the "View Past Transactions" page 2. The system shows the full database in a table format 3. The system also displays the options for a filtered search 4. The system also displays the option to view the transaction details
Alternative Flows:	
Exceptions:	
Includes:	<ol style="list-style-type: none"> 1. Filtered Search 2. Transaction details
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	A0019		
Use Case Name:	Refresh Master Data		
Created By:		Last Update By:	Hemanth
Date Created:	19/02/2023	Date Last Updated:	11/02/2023

Actor:	User (Admin)
Description:	The Admin User will refresh the master data list of past property transactions. This will enable the data to be updated via API call.
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. Users must have logged on to an account with administrative access.
Postcondition:	<ol style="list-style-type: none"> 1. The Master Data List of past property transactions is updated to date.
Priority:	Low
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user (admin) heads to the "View Past Transactions" page 2. The system shows the full data in a table format 3. User click on the the button "Refresh Data" 4. Once done, system notifies "Master Data has been refreshed successfully!!!" 5. List of Past Transactions is updated to date
Alternative Flows:	
Exceptions:	
Includes:	<ol style="list-style-type: none"> 1. Filtered Search 2. Transaction details
Special Requirements:	
Assumptions:	User has logged in as an administrator
Notes and Issues:	

Use Case ID:	A0020		
Use Case Name:	Master Data Filtered Search		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	12/02/2023

Actor:	User
Description:	Allows users to conduct a filtered search on the entire list of past property transactions based on a set of criteria they choose.
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. Users must have logged on to an account with administrative access.
Postcondition:	<ol style="list-style-type: none"> 1. The system will display all the entries that match the user's chosen filters.
Priority:	Low
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user goes to the page "View all Past Transactions" 2. The user clicks on the "Filter properties" 3. The system displays several options that the user can choose from 4. The user selects his or her filters and presses "Confirm selection" 5. The system only displays the entries that match the criteria
Alternative Flows:	
Exceptions:	<p>ERR1: The criteria does not return any entries at all!</p> <ol style="list-style-type: none"> 1. The system will display an error message to the user "There are no relevant entries! Please try again!" 2. The user presses the "Okay" button 3. The system will display all entries in the database 4. The flow returns to step 2 of Flow of events
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	A0021		
Use Case Name:	View Past Transaction Details		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	12/02/2023

Actor:	User
Description:	User is able to view the details of each past transaction available in the Past Property Transactions data
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. User must be logged in to view the Past Property Transactions
Postcondition:	<ol style="list-style-type: none"> 1. The system no longer displays the full list 2. The system displays the transaction details of the entry the user has selected
Priority:	Low
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user heads to the "View Past Transactions" page 2. The user clicks on the "View Details" button on a specific entry 3. The system displays only the transaction details of the entry chosen
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	User is logged in as a admin
Notes and Issues:	

Use Case ID:	A0022		
Use Case Name:	View Forum Page		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	12/02/2023

Actor:	User
Description:	Users can view the forum page where other users have published posts for the property market community. In this page, users can also publish their own posts and comment on others
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. User must be logged in to view the Forum Page
Postcondition:	<ol style="list-style-type: none"> 1. User can view the Forum page filled with several posts from multiple users 2. User has the option to create post, report post and comment on posts
Priority:	Low
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user heads "Forum" page 2. The system brings the user to the forum page with all the different posts made with the most recent one at the top
Alternative Flows:	ERR1: The user is not logged in <ol style="list-style-type: none"> 1. The system will display an error message "You have not logged in! You have to log in before entering this page" 2. The user clicks on "sign in" 3. The system goes to the login functionality
Exceptions:	
Includes:	<ol style="list-style-type: none"> 1. Create post 2. Delete post 3. Comment on post
Special Requirements:	
Assumptions:	The forum exists
Notes and Issues:	

Use Case ID:	A0023		
Use Case Name:	Create Forum Post		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	12/02/2023

Actor:	User
Description:	Users are able to create forum posts to discuss any matters at hand with the property market community. Users can comment on a post creating a thread of messages. On top of that, users can delete their posts or report a post if they deem it to be inappropriate.
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. User must be logged in to create a forum post
Postcondition:	<ol style="list-style-type: none"> 1. User creates a forum post with some text in it. 2. The forum post is published on the forum page where all other users can view and comment
Priority:	
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. User heads to the "Forum" page 2. User selects 'Create New Post' button. 3. User will be prompted to input some text. 4. Once completed, User can click on "Publish". 5. Then the system will notify "Post published successfully".
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	A0024		
Use Case Name:	Delete Forum Post		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	12/02/2023

Actor:	User
Description:	Users are able to delete their own forum posts if they wish to do so.
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. User must be logged in to delete a forum post 3. User must have already posted a forum post
Postcondition:	<ol style="list-style-type: none"> 1. The forum post that the user delete will not exist on the forum page anymore.
Priority:	Low
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user heads to the "Forum" page 2. The user clicks on his or her own post and selects "Delete this post" 3. The system displays "Are you sure?" 4. The user clicks "Yes" <ol style="list-style-type: none"> a. The post is deleted 5. The user clicks "no" <ol style="list-style-type: none"> a. The post is not deleted b. System returns to the post itself
Alternative Flows:	<p>AF-S1: The post cannot be deleted</p> <ol style="list-style-type: none"> 1. The system will display an error message "The post was not deleted! Please try again later" 2. The system returns to the post 3. The flow of events return to step 2
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	A0025		
Use Case Name:	Comment on Forum Post		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	12/02/2023

Actor:	User
Description:	Users can make a comment on a post of their choosing
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. User must be logged in to comment on a forum post
Postcondition:	<ol style="list-style-type: none"> 1. The comment will appear on the forum post and will be visible
Priority:	Low
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user clicks on make comment button 2. The system prompts the user to write a comment 3. The user clicks on post comment 4. The system posts the comment
Alternative Flows:	AF-S1: Whe written comment is empty <ol style="list-style-type: none"> 1. The system prints an error message "the comment cannot be empty!" 2. The user clicks on the "okay" button 3. The flow of control returns to step 2 of flow of events
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	A0026		
Use Case Name:	Request for Help		
Created By:		Last Update By:	Hemanth
Date Created:	19/01/2023	Date Last Updated:	12/02/2023

Actor:	User and Customer Service
Description:	Users can communicate with customer service for help on a issue they are facing
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. User must be logged in to request for help 3. The user is at the main account page
Postcondition:	<ol style="list-style-type: none"> 1. The user is no longer seeing the main account page 2. The user is in a chatroom with a customer service representative 3. The user is able to send an instant message to customer service
Priority:	Low
Frequency of use:	
Flow of events:	<ol style="list-style-type: none"> 1. The user clicks on "Contact Customer Service" button. 2. The system pops up a chatroom between user and customer service 3. The system allows the user to send a message to customer service 4. The system also allows the customer service to send messages back to the user 5. The system will display the messages sent between the two parties
Alternative Flows:	ERR1: The message fails to send <ol style="list-style-type: none"> 1. The system will display an error message to the user "the message failed to send. Please try again" 2. The user clicks on "okay" 3. The system returns to step 3 in the flow of events
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	

Notes and Issues:	
-------------------	--

Use Case ID:	A0027		
Use Case Name:	Estimate Property price		
Created By:	Jack	Last Update By:	Hillary
Date Created:	03/03/2023	Date Last Updated:	05/03/2023

Actor:	User (Tenant / Landlord)
Description:	When a user (Tenant) clicks on predict property price on the property listing page, the system will look at past records. Using an algorithm, the system will use the past data to predict a possible price that the property will sell for.
Precondition:	<ol style="list-style-type: none"> 1. User's device should have access to the internet and GPS should be turned on. 2. Property Listing should be in the database system 3. User must be logged in as a Tenant or Landlord to view a Property Listing 4. User's account must have been verified [Email + Phone Number] 5. The user is viewing a property listing
Postcondition:	<ol style="list-style-type: none"> 1. Users can view the estimated price of the property
Priority:	
Frequency of use:	5
Flow of events:	<ol style="list-style-type: none"> 1. The User clicks on "estimate property price" 2. The System will query the Database for the sale price of the other properties with identical attribute tags 3. The System will use an algorithm to predict the price of the property in question. 4. The System will display the predicted price 5. The User clicks "okay" 6. User returns back to property listing page
Alternative Flows:	<p><u>AF1 — No previous property with identical tags exist</u></p> <ol style="list-style-type: none"> 1. The system will display "unknown, no previous records to rely on"

	2. The system returns the user to the property listing page <u>AF2 — There are less than 3 previous records</u> 1. The system will display “not enough previous record to make a reliable prediction” 2. The system returns the user to the property listing page
Exceptions:	
Includes:	1. Refresh Master Data
Special Requirements:	
Assumptions:	The user must have an existing account and of type Landlord / tenant.
Notes and Issues:	

5.0 UI Mockups

Landing Page

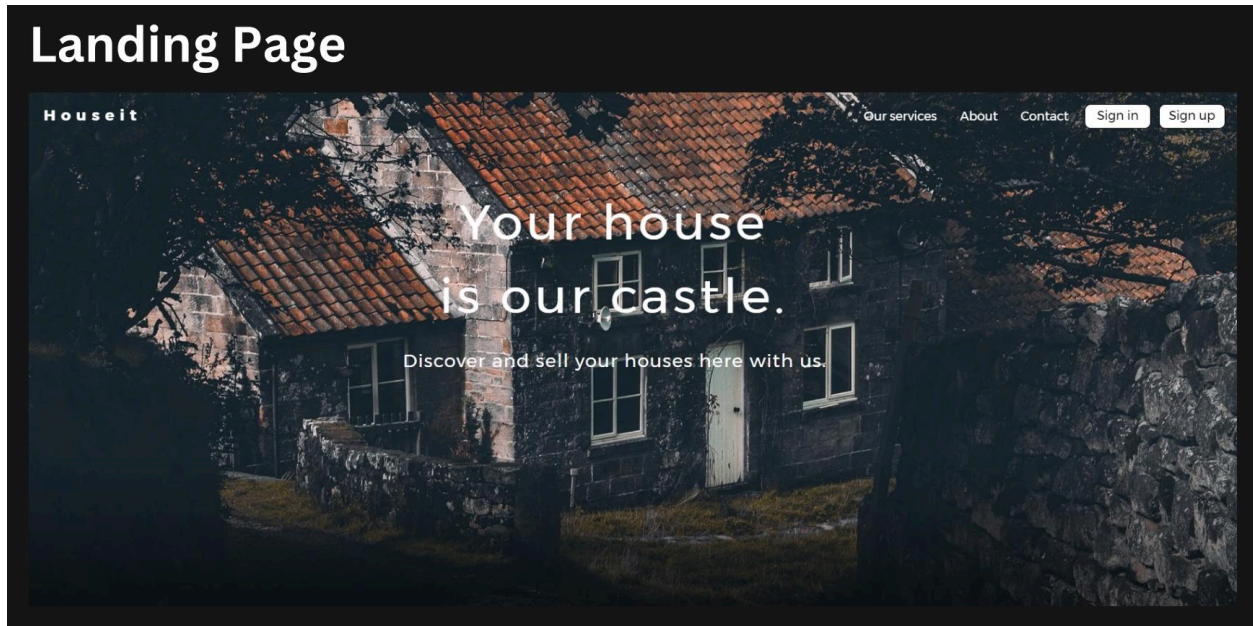


Fig (1). Main landing page of our web application

Sign up Page

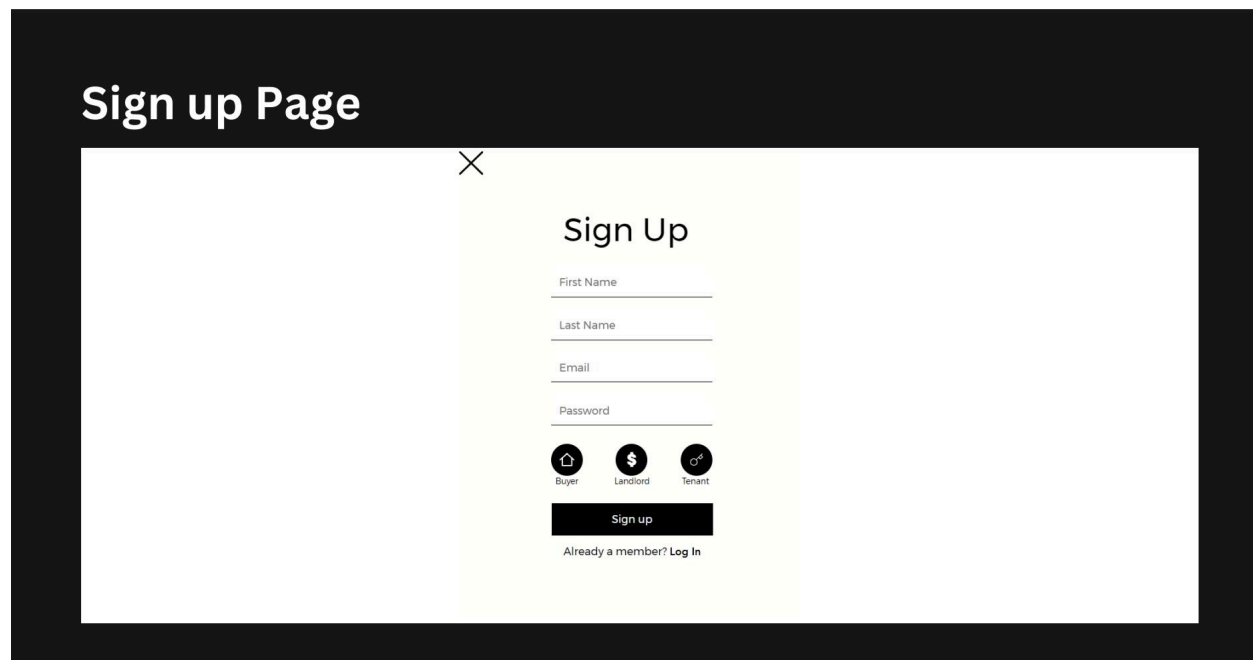


Fig (2) Allows for user to create an account

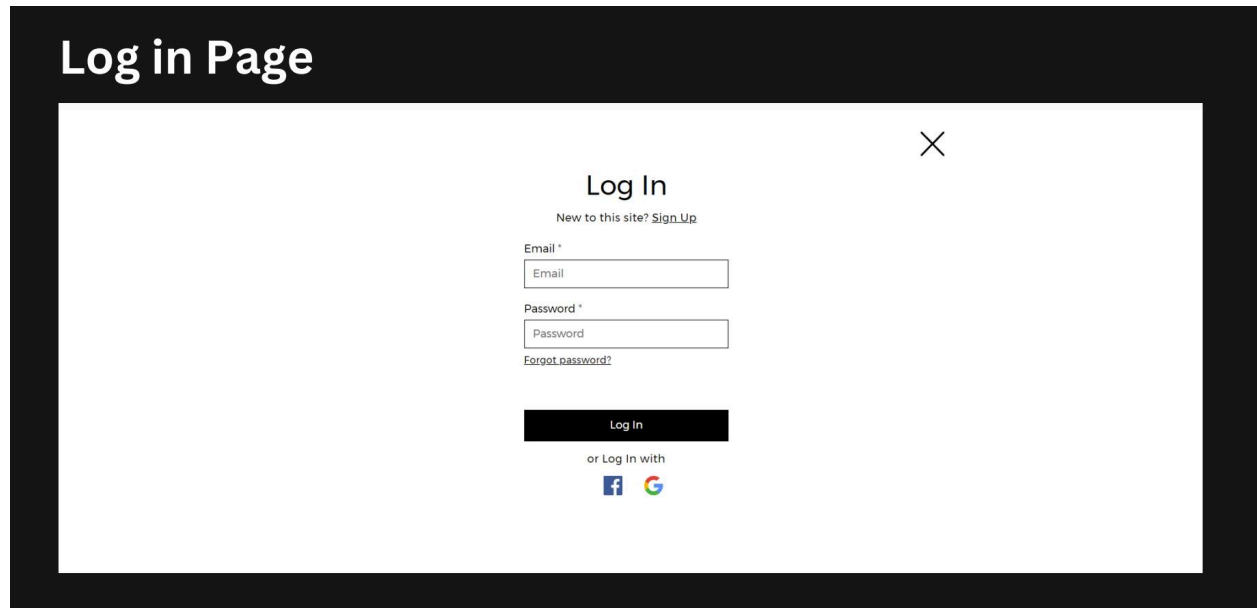


Fig (3) Page for users to log into their account

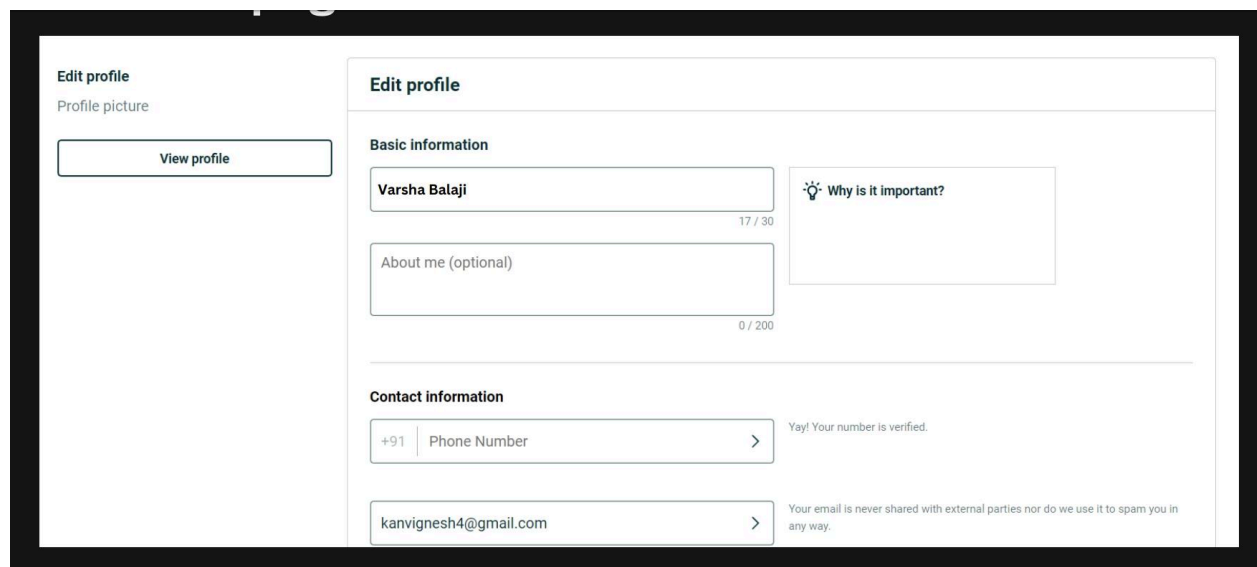


Fig (3) Page for users to view/edit their Profile on the application

Property listing - landlord

List Property

Please complete the form to
post your property

property Name	Property type
<input type="text"/>	<input type="text"/>
property Age	Price *
<input type="text"/>	<input type="text"/>
Bedrooms *	Lising type
<input type="text"/>	<input type="text"/>
Adress *	
<input type="text"/>	
<input type="button" value="Continue"/>	

Fig(4) The create property listing page. The user (landlord) can enter details of their property in order to create a listing that can be posted later

Property listing -landlord

Houseit



Our services

About

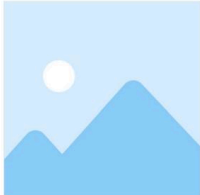
Contact

Properties

Sign in

Sign up

Properties listed



Property 1

Description

[edit](#) [Delete](#)

Fig (5.1) Allows landlord to **view** the property listing that they have already created

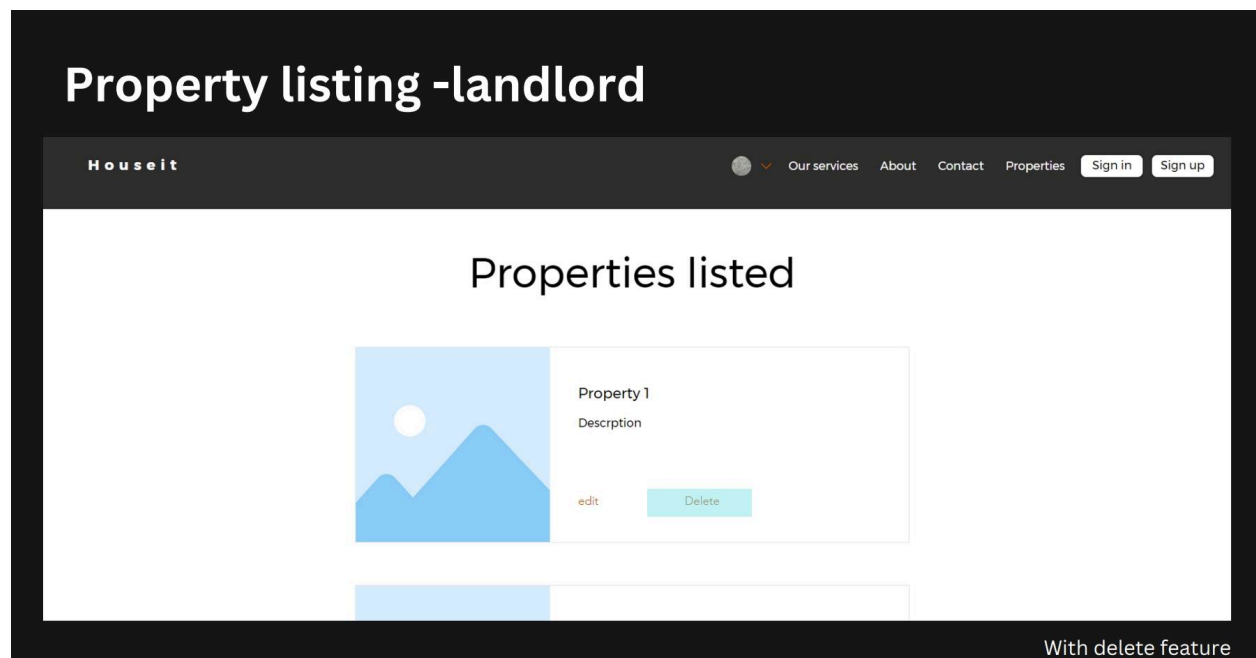


Fig (5.2) Allows landlord to **edit**, and **delete** the property listing that they have already created

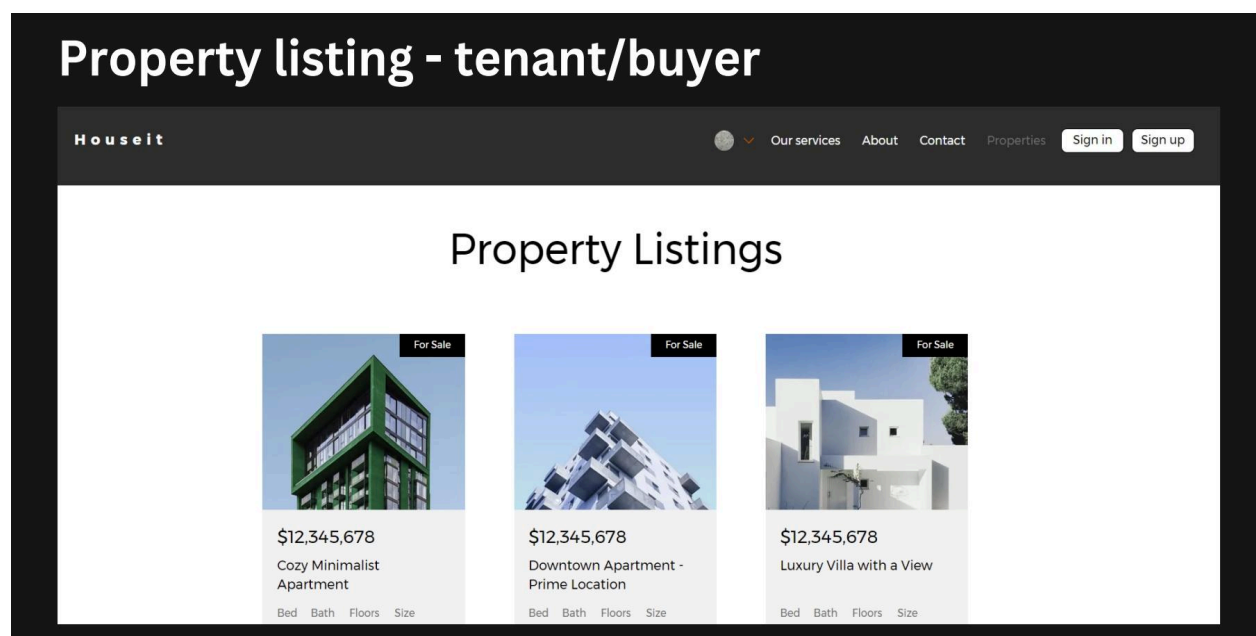


Fig (6) Property listing page for the tenant/Landlord

Property listing - tenant/buyer

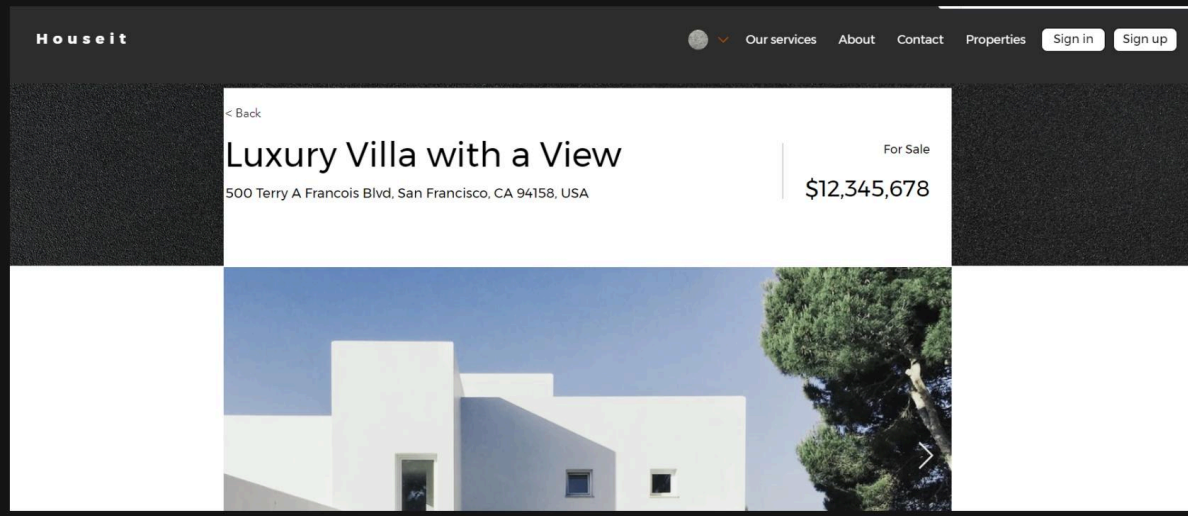


Fig (7) The user (tenant / Landlord) viewing the specific listing that they have clicked on

Property listing - tenant/buyer

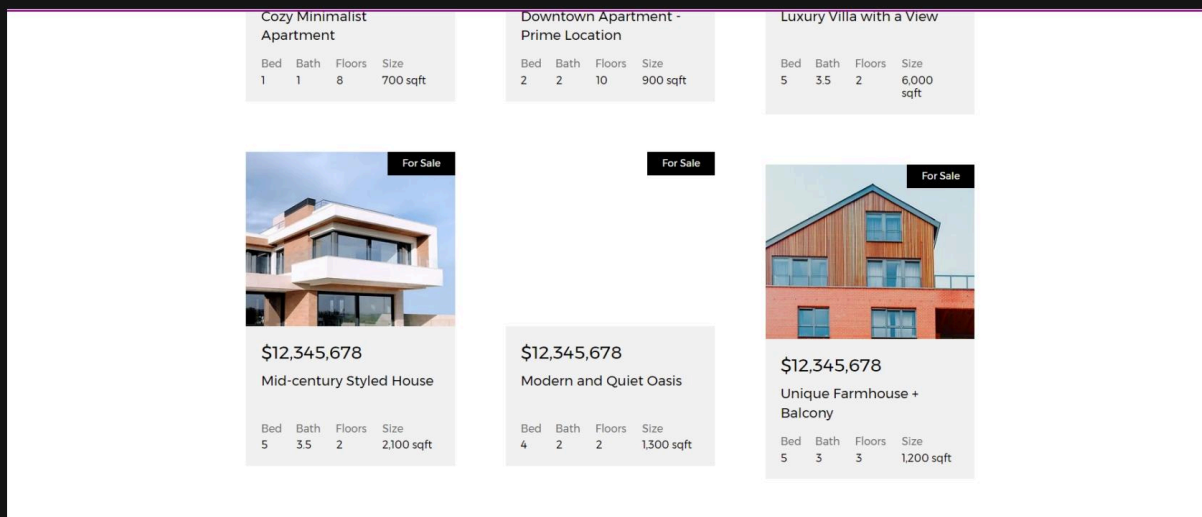



Fig (8) page showing a listing of the property listing that is open and posted

Property listing - tenant/buyer



Property Description

This is placeholder text. To change this content, double-click on the element and click Change Content. To manage all your collections, click on the Content Manager button in the Add panel on the left.

Contact Agent

Kelly Parker
123-456-7890
info@mysite.com

Property Details

Property Type	Bedrooms	Bathrooms
Villa	5	3.5
Size	Floors	Year Built
6,000 sqft	2	1985

Property Location

Fig (9) Details of the property listing for tenants/Landlord


Property listing - tenant/buyer

Property Details

Property Type	Bedrooms	Bathrooms
Villa	5	3.5
Size	Floors	Year Built
6,000 sqft	2	1985

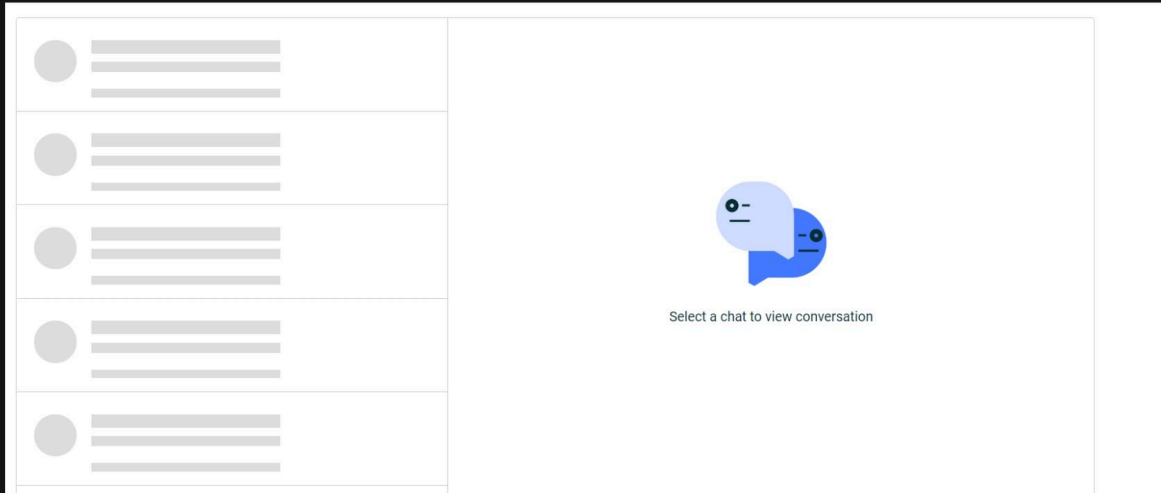
Property Location

500 Terry A Francois Blvd, San Francisco, CA 94158, USA



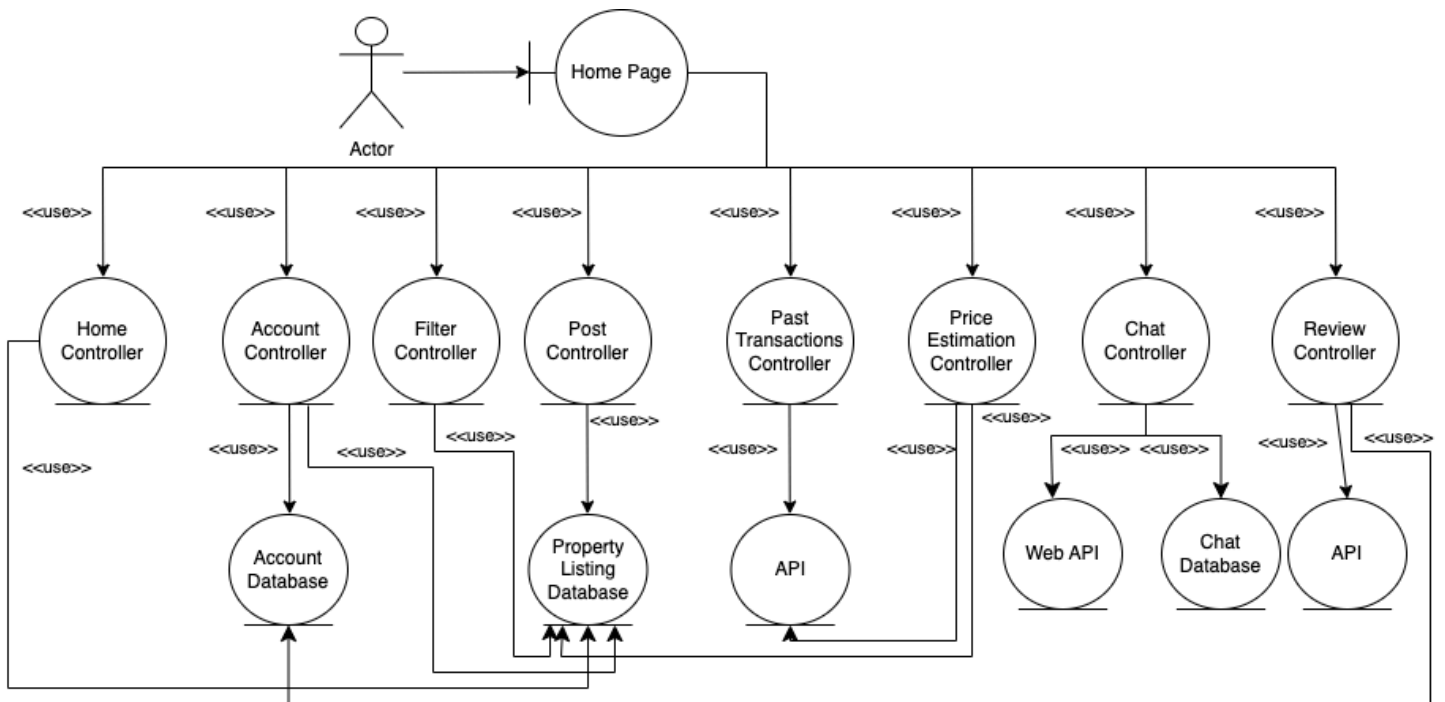
Fig(10) Allows the tenant/Landlord to view the property listing. Users can check the details of the property such as location and property type.

Property listing chat system - tenant/buyer

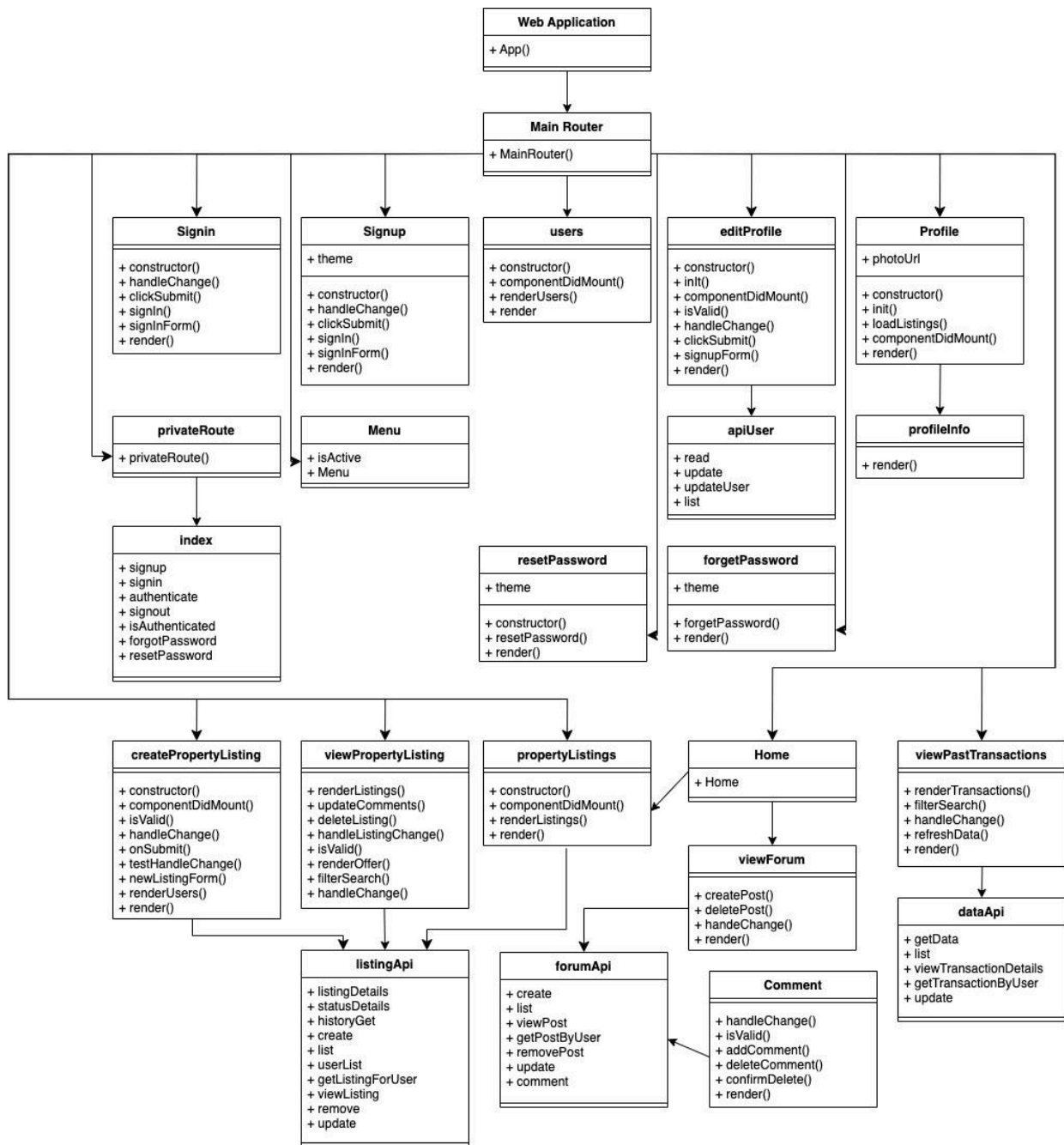


Fig(11) Allows the tenant/Landlord to establish contact with the landlord of interest through a chat forum

6.0 Class Diagrams

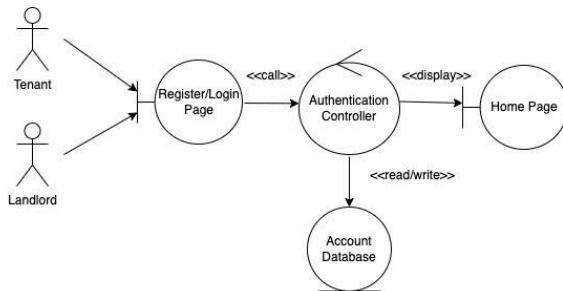


comment

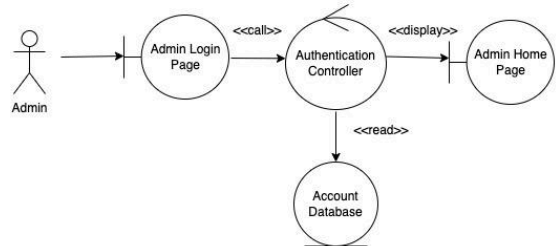


7.0 Conceptual models

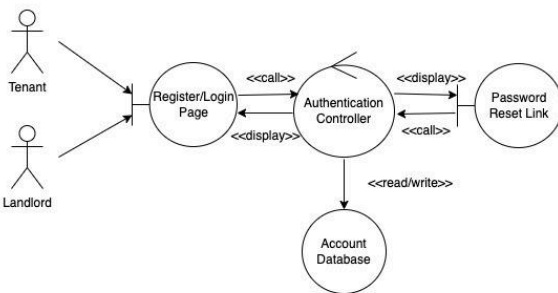
Tenant/Landlord Registration
Tenant/Landlord Log in



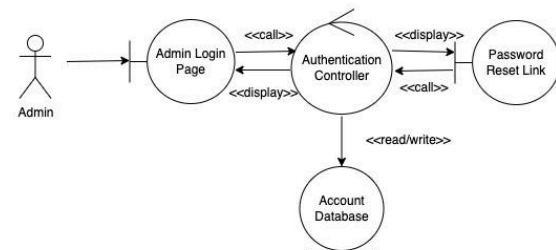
Admin Log in



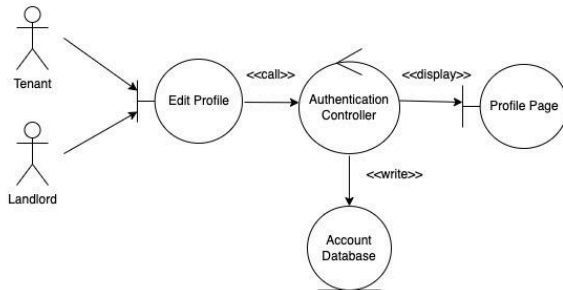
Tenant/Landlord Forget Password



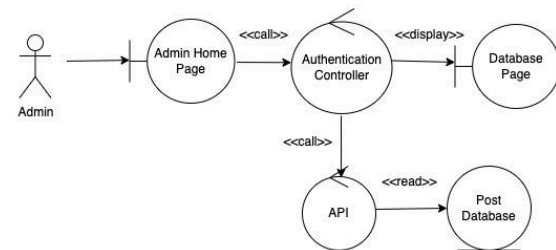
Admin Forget Password



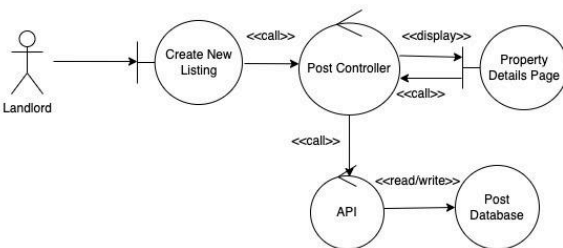
Update Account information



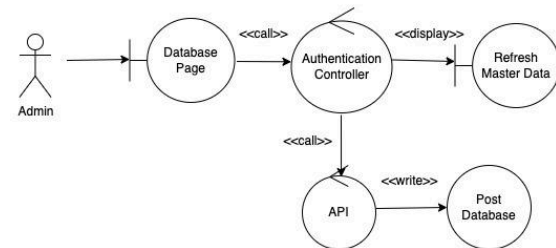
View Database



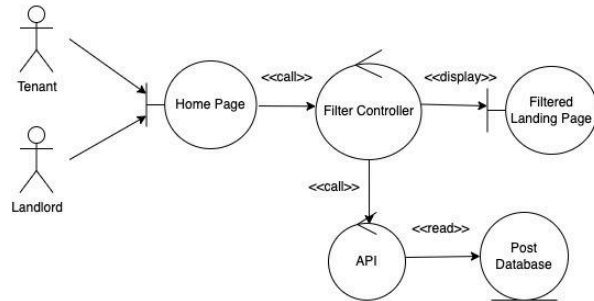
Create/Post/Update/Delete/Close
Property Listing



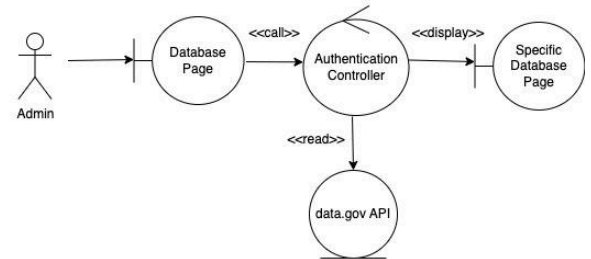
Refresh Master Data



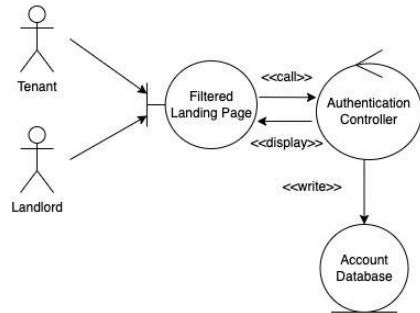
View/Filter Property Listing



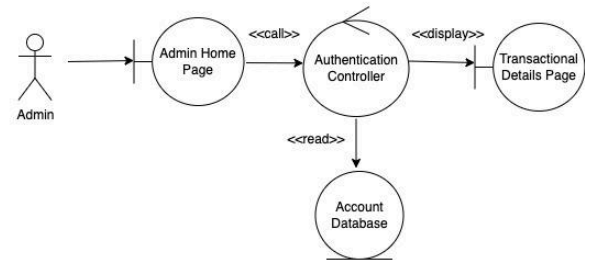
Search Database



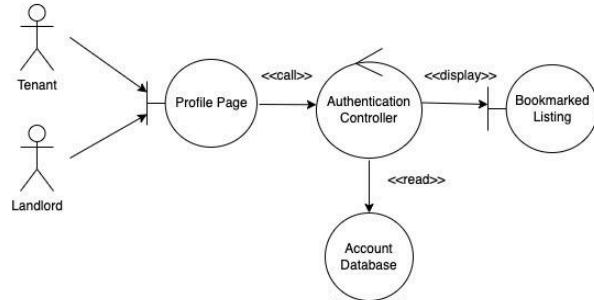
Bookmark/Remove Bookmarked Listing



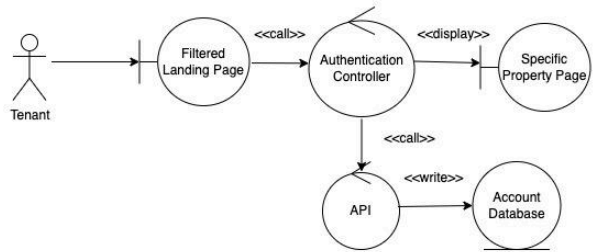
View Transactional Details



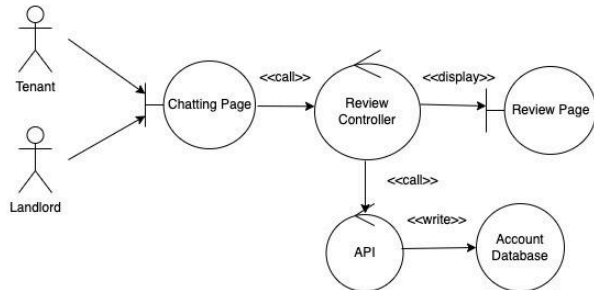
View Bookmarked Listing



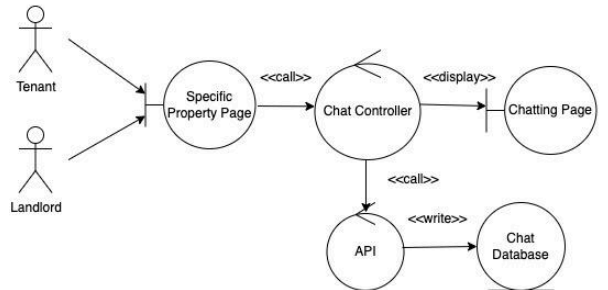
Rent/Buy property



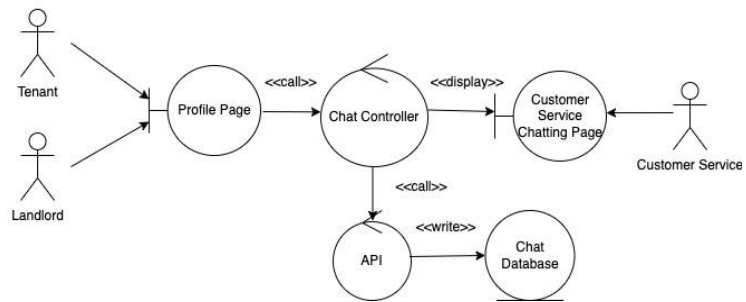
Review User



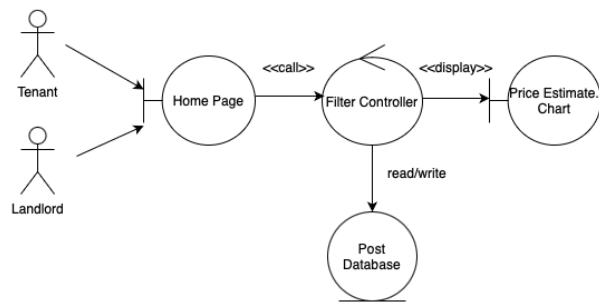
Contact Landlord



Request for help

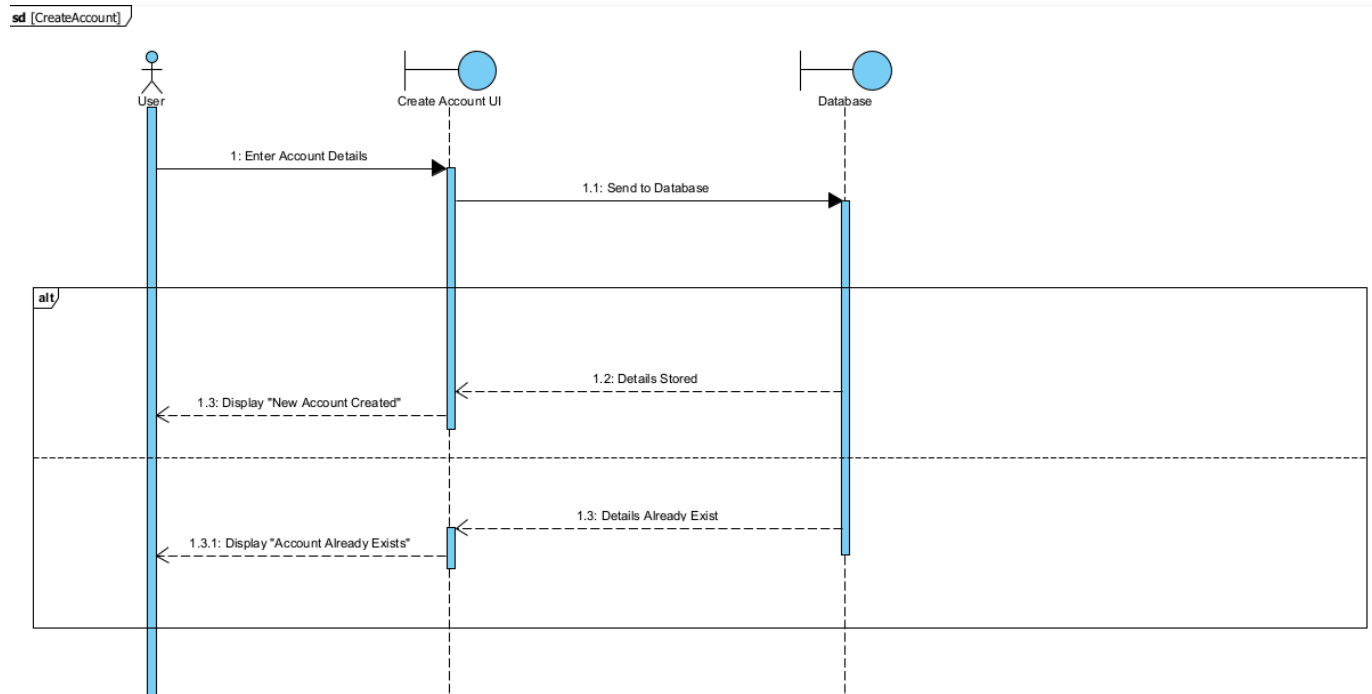


Price Estimate Charting

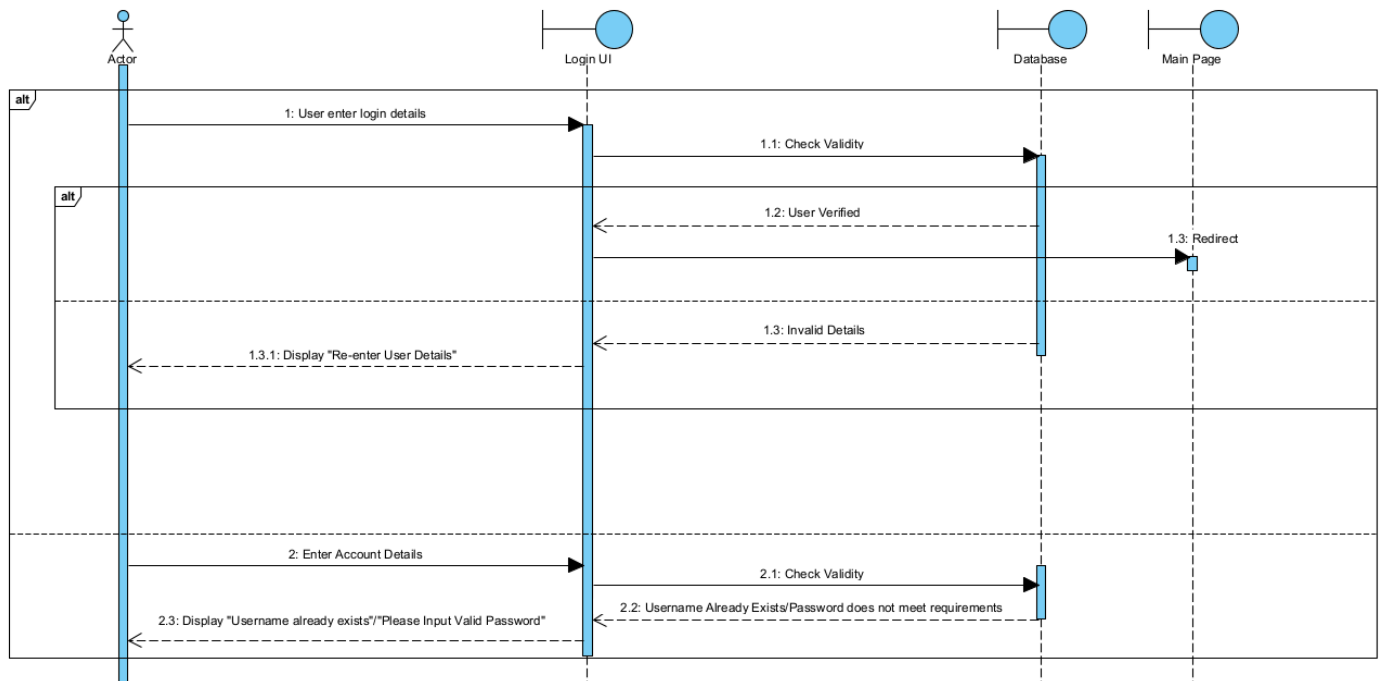


8.0 Sequence Diagrams

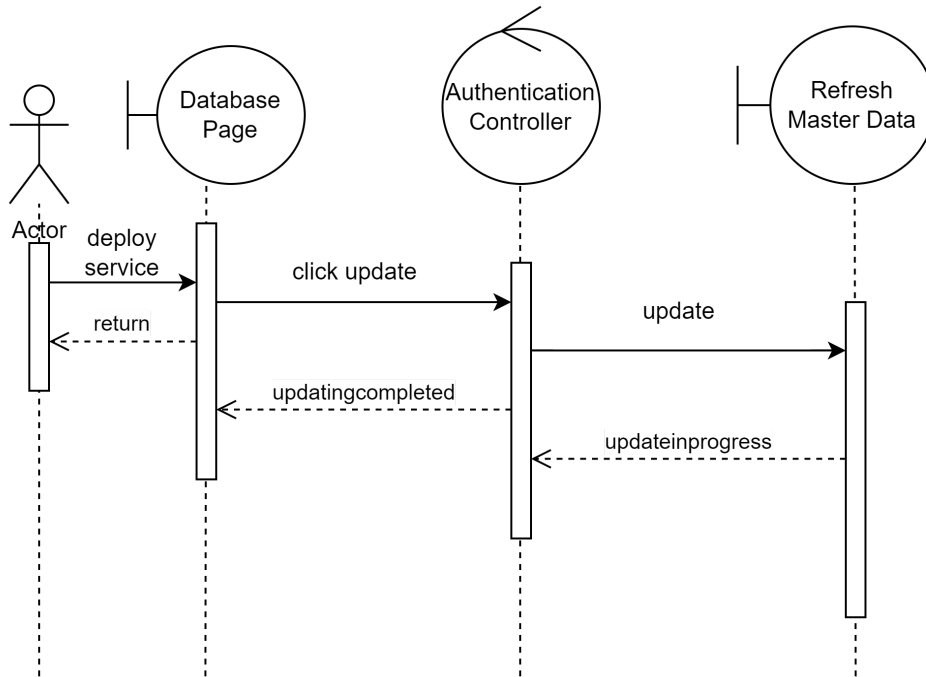
Create Account :



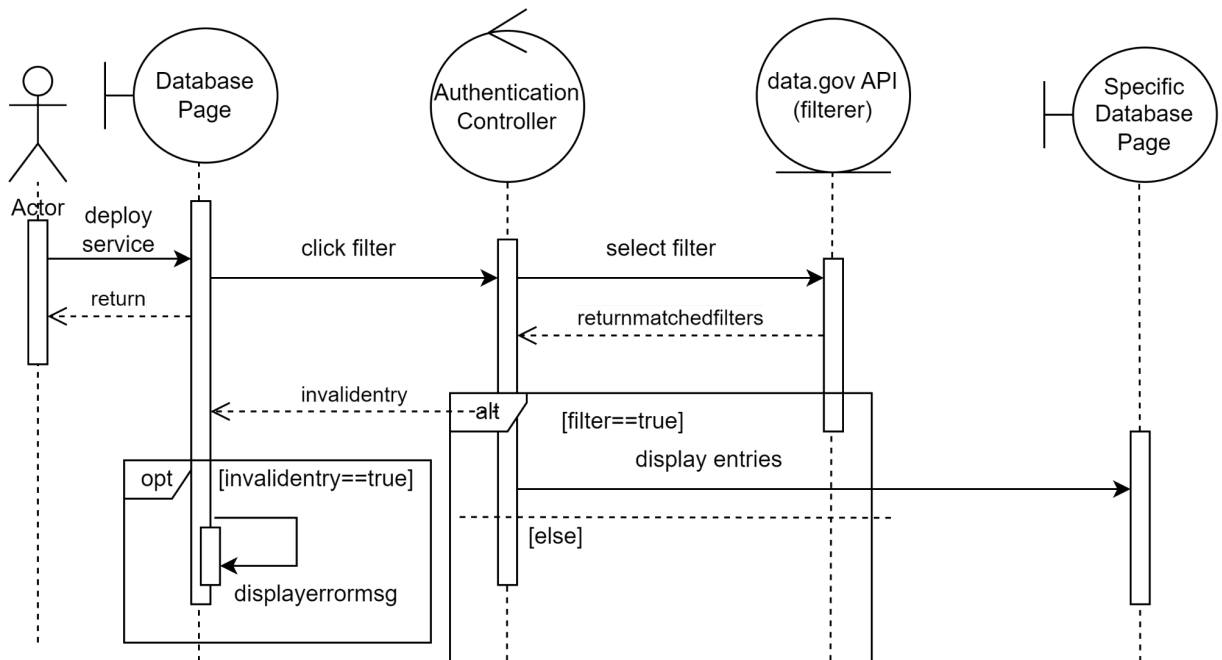
User Login:



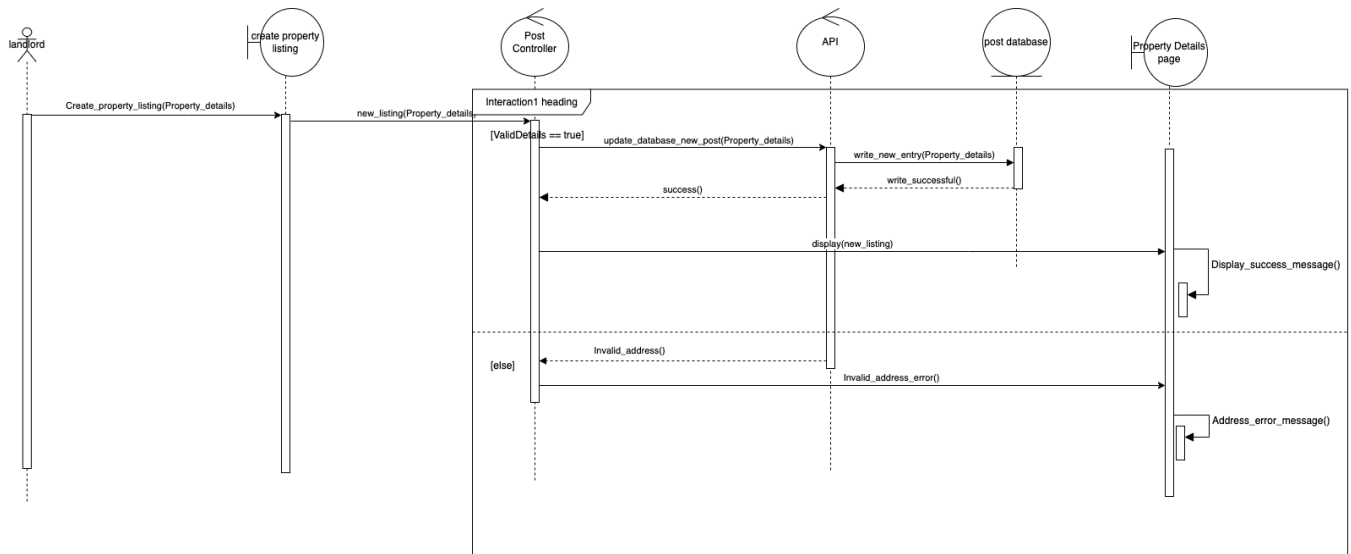
Refresh Master Data



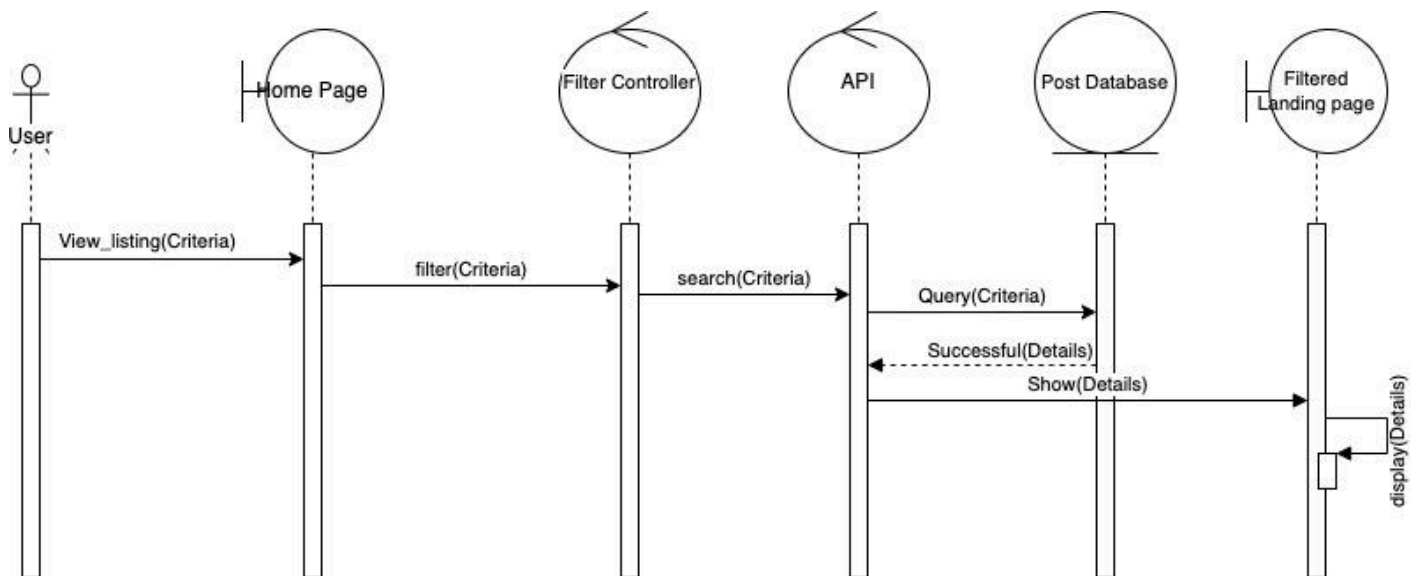
Master data filtered search



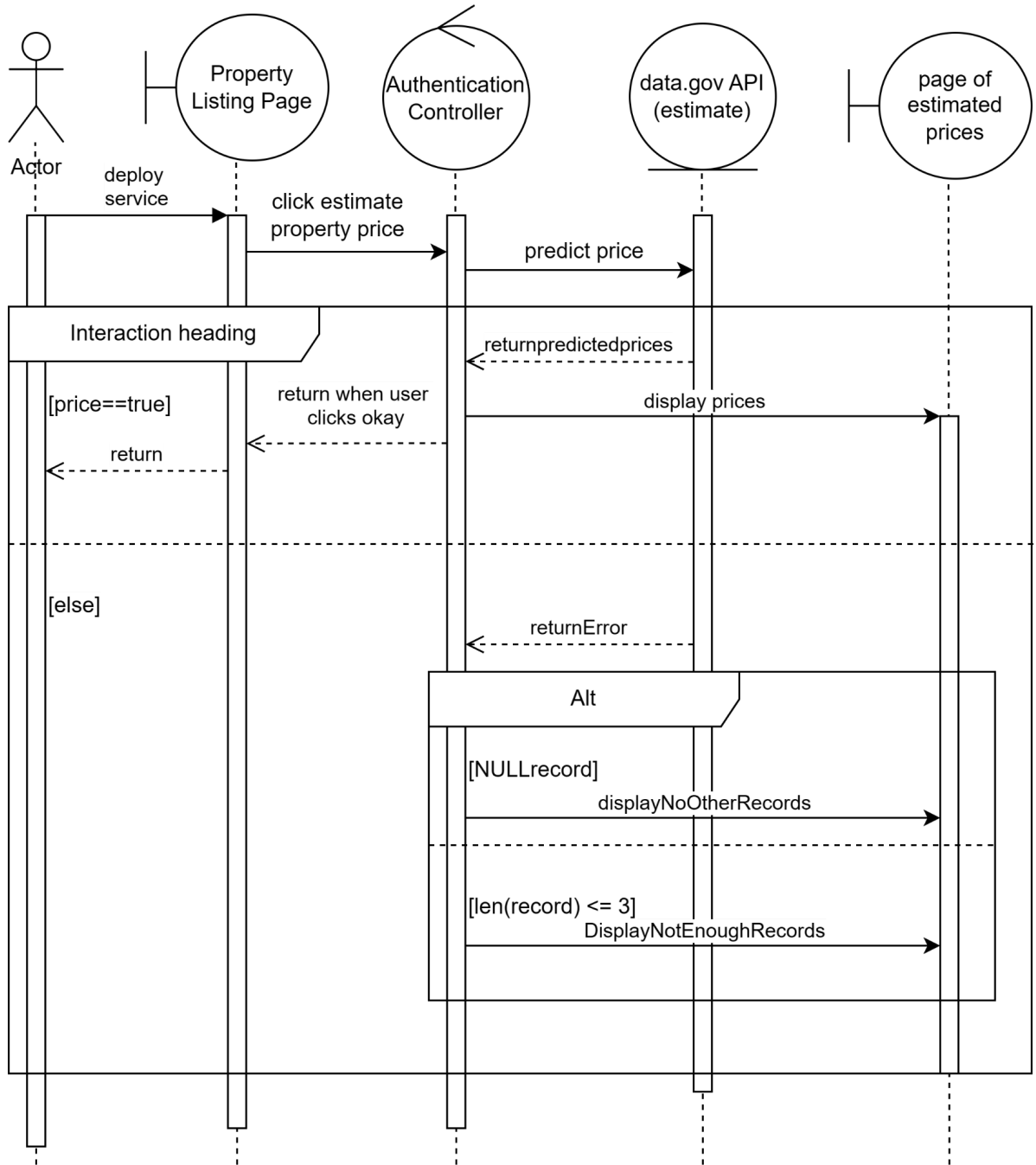
Create Property Listing



View Property Listing

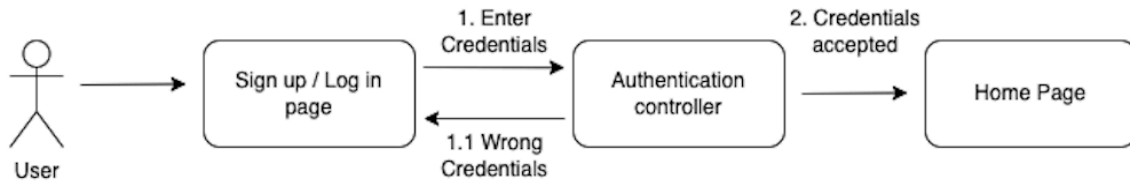


Estimate price

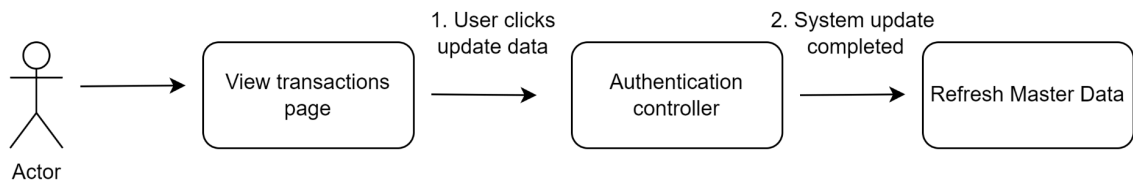


9.0 Dialog Maps

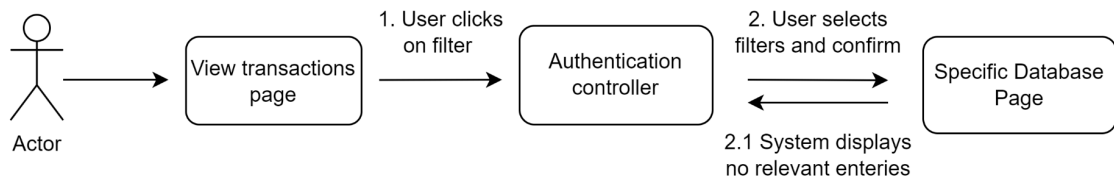
Register new account and log in



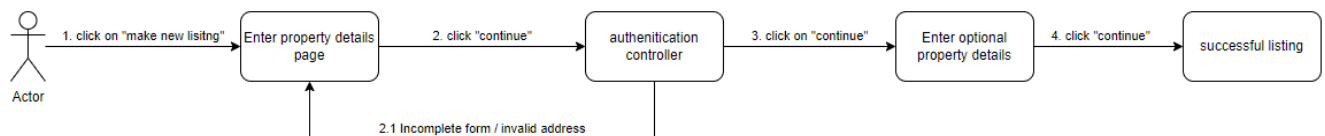
Refresh Master Data

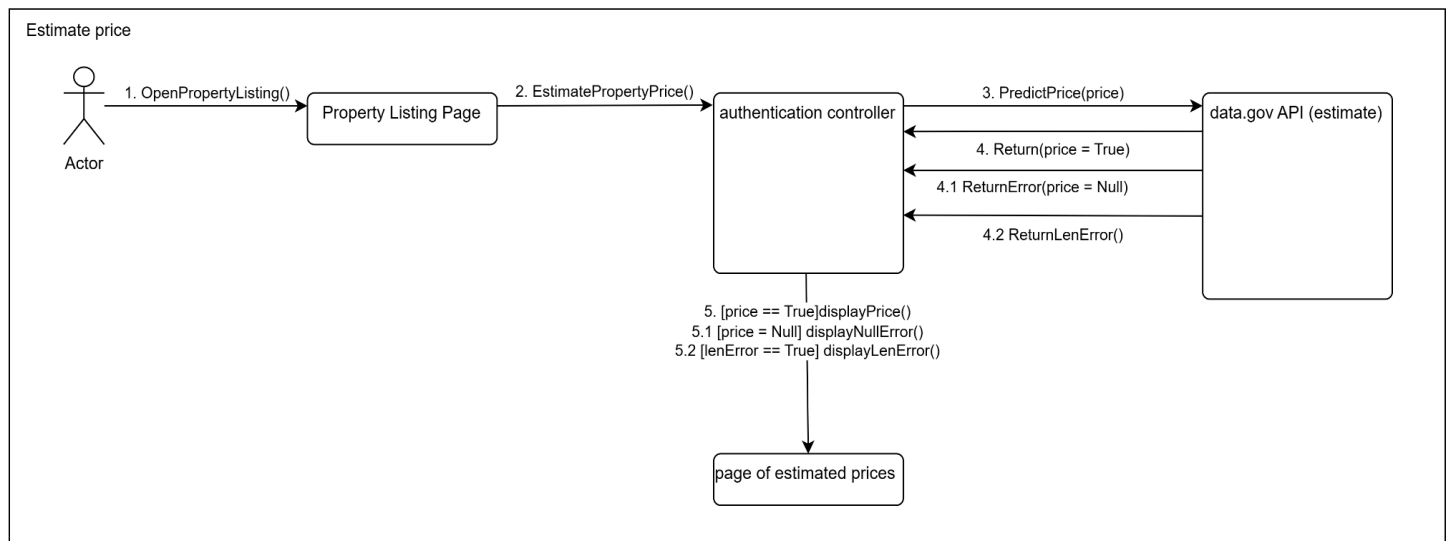
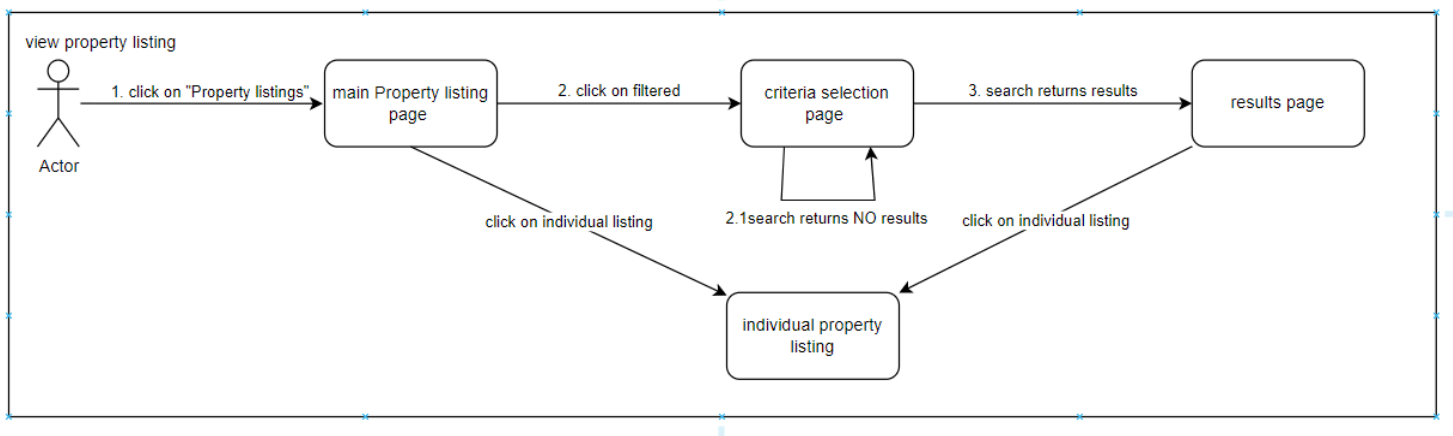


Master Data Filtered Search

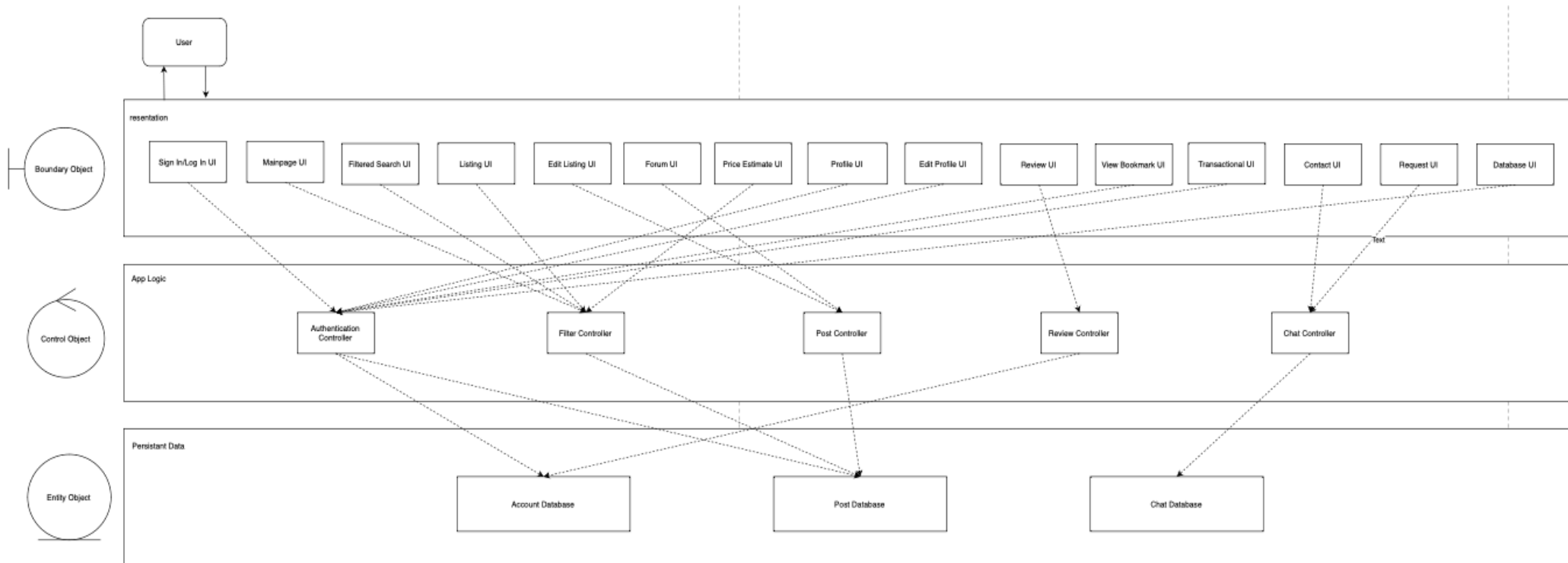


create new listing





10.0 System Architecture



11.0 Application Skeleton

Application Overview

```
project/
├── app/
│   ├── migrations/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── urls.py
│   └── views.py
├── project/
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── manage.py
```

We will be looking into our `models.py` to understand the different classes that encompass our application and its key classes and description.

```

class UserManager(BaseUserManager):
    """Define a model manager for User model with no username field."""

    use_in_migrations = True

    def _create_user(self, email, password, **extra_fields):
        """Create and save a User with the given email and password."""
        if not email:
            raise ValueError('The given email must be set')
        email = self.normalize_email(email)
        user = self.model(email=email, **extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_user(self, email, password=None, **extra_fields):
        """Create and save a regular User with the given email and password."""
        extra_fields.setdefault('is_staff', False)
        extra_fields.setdefault('is_superuser', False)
        return self._create_user(email, password, **extra_fields)

    def create_superuser(self, email, password, **extra_fields):
        """Create and save a SuperUser with the given email and password."""
        extra_fields.setdefault('is_staff', True)
        extra_fields.setdefault('is_superuser', True)

        if extra_fields.get('is_staff') is not True:
            raise ValueError('Superuser must have is_staff=True.')
        if extra_fields.get('is_superuser') is not True:
            raise ValueError('Superuser must have is_superuser=True.')

        return self._create_user(email, password, **extra_fields)

```

1. UserManager(BaseUserManager)

This class is a custom manager for a User model in Django. The User model is the default authentication model in Django that provides features such as user authentication, permission handling, and password hashing.

The methods defined in this class are:

- a) `_create_user`: This method creates and saves a User object with the given email and password. If the email is not provided, a `ValueError` is raised. The email is normalized before creating the user object, and the password is hashed using Django's default password hashing algorithm.
- b) `create_user`: This method creates and saves a regular User object with the given email and password. It calls the `_create_user` method with default values for `is_staff` and `is_superuser`.
- c) `create_superuser`: This method creates and saves a Superuser object with the given email and password. It calls the `_create_user` method with values for `is_staff` and `is_superuser` set to `True`. If either of these values is not `True`, a `ValueError` is raised.

The `use_in_migrations` attribute is set to `True` to indicate that this manager should be used during database migrations.

```

class User(AbstractUser):
    username = models.CharField(max_length=32, unique=True, default=False)
    is_admin = models.BooleanField(default=False)
    email = models.EmailField(('email address'), unique=True)
    first_name = models.CharField(max_length=150, blank=True)
    last_name = models.CharField(max_length=150, blank=True)
    address = models.CharField(max_length=255, blank=True)
    user_type = models.CharField(max_length=10)

    class Meta:
        swappable = 'AUTH_USER_MODEL'

class Masterflatmodeldata(models.Model):
    flatmodelid = models.AutoField(db_column='flatModelID', blank=True, null=False, primary_key=True) # Field name made lowercase.
    flatmodel = models.TextField(db_column='flatModel', blank=True, null=False) # Field name made lowercase.

    class Meta:
        managed = False
        db_table = 'MasterFlatModelData'

class Masterflattypes(models.Model):
    flattypeid = models.AutoField(db_column='flatTypeID', blank=True, null=False, primary_key=True) # Field name made lowercase.
    flattype = models.TextField(db_column='flatType', blank=True, null=False) # Field name made lowercase.

    class Meta:
        managed = False
        db_table = 'MasterFlatTypes'

class Masterstreetdata(models.Model):
    streetid = models.AutoField(db_column='streetID', blank=True, null=False, primary_key=True) # Field name made lowercase.
    streetname = models.TextField(db_column='streetName', blank=True, null=False) # Field name made lowercase.

    class Meta:
        managed = False
        db_table = 'MasterStreetData'

```

2. User(AbstractUser)

This model extends the built-in AbstractUser model provided by Django and adds additional fields to it. The username field is changed to a CharField with a maximum length of 32, and is_admin, email, first_name, last_name, address, and user_type fields are added. is_admin is a BooleanField that is False by default. The email field is an EmailField with a unique constraint. The Meta class specifies that this model is swappable with the default AUTH_USER_MODEL model.

3. Masterflatmodeldata

This model represents the flat model data stored in the MasterFlatModelData table in the database. It has two fields: flatmodelid, which is an AutoField acting as a primary key, and flatmodel, which is a TextField that can store flat model data as a string. The Meta class specifies that this model is not managed by Django.

4. Masterflattypes

This model represents the flat types stored in the MasterFlatTypes table in the database. It has two fields: flattypeid, which is an AutoField acting as a primary key, and flattype,

which is a TextField that can store flat type data as a string. The Meta class specifies that this model is not managed by Django.

5. Masterstreetdata

This model represents the street data stored in the MasterStreetData table in the database. It has two fields: streetid, which is an AutoField acting as a primary key, and streetname, which is a TextField that can store street names as a string. The Meta class specifies that this model is not managed by Django.

```
class Mastertowndata(models.Model):
    townid = models.AutoField(db_column='townID', blank=True, null=False, primary_key=True) # Field name made lowercase.
    townname = models.TextField(db_column='townName', blank=True, null=False) # Field name made lowercase.

    class Meta:
        managed = False
        db_table = 'MasterTownData'

class Masterpropertyresaledata(models.Model):
    resalepropertyid = models.AutoField(db_column='resalePropertyID', primary_key=True, blank=True, null=False) # Field name made lowercase.
    yyyyymm = models.TextField(db_column='YYYYMM', blank=True, null=False) # Field name made lowercase.
    townid = models.IntegerField(db_column='townID', blank=True, null=False) # Field name made lowercase.
    flattypeid = models.IntegerField(db_column='flatTypeID', blank=True, null=False) # Field name made lowercase.
    block = models.TextField(blank=True, null=False)
    streetid = models.IntegerField(db_column='streetID', blank=True, null=False) # Field name made lowercase.
    floorareainsqm = models.TextField(db_column='floorAreaInSqm', blank=True, null=False) # Field name made lowercase.
    flatmodelid = models.IntegerField(db_column='flatModelID', blank=True, null=False) # Field name made lowercase.
    leasecommencedate = models.TextField(db_column='leaseCommenceDate', blank=True, null=False) # Field name made lowercase.
    remainingleaseyears = models.TextField(db_column='remainingLeaseYears', blank=True, null=False) # Field name made lowercase.
    resaleprice = models.DecimalField(db_column='resalePrice', max_digits=10, decimal_places=5, blank=True, null=False) # Field name made lowercase.

    class Meta:
        managed = False
        db_table = 'MasterPropertyResaleData'

class Masterpropertyrentaldata(models.Model):
    rentalpropertyid = models.AutoField(db_column='rentalPropertyID', primary_key=True, blank=True, null=False) # Field name made lowercase.
    yyyyymm = models.TextField(db_column='YYYYMM', blank=True, null=False) # Field name made lowercase.
    townid = models.ForeignKey(Mastertowndata, on_delete=models.CASCADE)
    block = models.TextField(blank=True, null=False)
    streetid = models.IntegerField(db_column='streetID', blank=True, null=False) # Field name made lowercase.
    flattypeid = models.IntegerField(db_column='flatTypeID', blank=True, null=False) # Field name made lowercase.
    monthlyrent = models.IntegerField(db_column='monthlyRent', blank=True, null=False) # Field name made lowercase.

    class Meta:
        managed = False
        db_table = 'MasterPropertyRentalData'
```

6. Mastertowndata

This is a model that represents data about towns. It has two fields: townid and townname. townid is an auto-incrementing integer field that serves as the primary key for the model. townname is a text field that stores the name of the town. The Meta class for this model specifies that it is unmanaged (managed = False) and that its database table name is MasterTownData (db_table = 'MasterTownData').

7. Masterpropertyresaledata

This is a model that represents data about resale properties. It has several fields that store information about a property, including its resalepropertyid, yyyyymm, townid, flattypeid, block, streetid, floorareainsqm, flatmodelid, leasecommencedate, remainingleaseyears,

and resaleprice. resalepropertyid is the primary key for the model. The Meta class for this model specifies that it is unmanaged (managed = False) and that its database table name is MasterPropertyResaleData (db_table = 'MasterPropertyResaleData').

8. Masterpropertyrentaldata

This is a model that represents data about rental properties. It has several fields that store information about a property, including its rentalpropertyid, yyyyymm, townid, block, streetid, flattypeid, and monthlyrent. rentalpropertyid is the primary key for the model. The Meta class for this model specifies that it is unmanaged (managed = False) and that its database table name is MasterPropertyRentalData (db_table = 'MasterPropertyRentalData').

```
class Listedproperty(models.Model):
    propertyid = models.AutoField(db_column='propertyID', primary_key=True, null=False) # Field name made lowercase.
    propertyownerid = models.IntegerField(db_column='propertyOwnerID', blank=True, null=True) # Field name made lowercase.
    # propertyflattypeid = models.ForeignKey('Masterflattypes', models.DO_NOTHING, db_column='propertyFlatTypeID', blank=True, null=True) # Field name made lower
    propertyflattypeid = models.IntegerField(db_column='propertyFlatTypeID', blank=True, null=True) # Field name made lowercase.
    propertyblock = models.TextField(db_column='propertyBlock', blank=True, null=True) # Field name made lowercase.
    propertytownid = models.IntegerField(db_column='propertyTownID', blank=True, null=True) # Field name made lowercase.
    propertyflatmodelid = models.IntegerField(db_column='propertyFlatModelID', blank=True, null=True) # Field name made lowercase.
    propertystreetid = models.IntegerField(db_column='propertyStreetID', blank=True, null=True) # Field name made lowercase.
    propertyage = models.IntegerField(db_column='propertyAge', blank=True, null=True) # Field name made lowercase.
    floorareainsqm = models.IntegerField(db_column='floorAreaInSqm', blank=True, null=True) # Field name made lowercase.
    saleorrentalflag = models.TextField(db_column='saleOrRentalFlag', blank=True, null=True) # Field name made lowercase. This field type is a guess.
    askingmonthlyrent = models.IntegerField(db_column='askingMonthlyRent', blank=True, null=True) # Field name made lowercase.
    leasecommencedate = models.DateField(db_column='leaseCommenceDate', blank=True, null=True) # Field name made lowercase.
    remainingleaseyears = models.IntegerField(db_column='remainingLeaseYears', blank=True, null=True) # Field name made lowercase.
    askingprice = models.IntegerField(db_column='AskingPrice', blank=True, null=True) # Field name made lowercase.
    propertystatus = models.TextField(db_column='propertyStatus', blank=True, null=True) # Field name made lowercase. This field type is a guess.
    propertyimage = models.FileField(db_column='propertyImage', blank=True, null=True)

    class Meta:
        managed = False
        db_table = 'ListedProperty'
```

9. Listedproperty

This is a model class called "Listedproperty", which defines the structure of a database table called "ListedProperty". It contains fields that represent the properties of a listed property. The Meta class for this model specifies that it is unmanaged (managed = False) and that its database table name is ListedProperty

12.0 Testing

12.1 Black Box Testing

12.2 White Box Testing

13.0 Good Software Engineering Practices

