Software Requirements Specification

for

HomeGoWhere

Version 1.3 approved

Prepared by Lim Jing Rong, Loo Ping Wee, Goh Shuen Wei, Ng Jing En, Babu Sankar Nithin Sankar, Lau Zhan You

SC2006 SDAD Team 6

03/11/2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Lim Jing Rong	03/09/2024	Add Lab 1 Deliverables	1.0
Lim Jing Rong	17/09/2024	Add Lab 2 Deliverables	1.1
Lau Zhan You	22/10/2024	Add Lab 3 Deliverables and updated diagrams	1.2
Goh Shuen Wei	03/11/2024	Add Lab 4 Deliverables and updated diagrams	1.3

1. Introduction

1.1 Purpose

This document specifies the software requirements for HomeGoWhere, a mobile application developed to simplify the renting process in Singapore. This Software Requirements Specification (SRS) document details the requirements for HomeGoWhere, covering functional and non-functional specifications for the core components of the application. This SRS encompasses the entire system, describing both frontend and backend functionalities and integration with various external data APIs.

1.2 Document Conventions

This SRS follows the following documentation conventions to ensure clarity and consistency:

- Font Styles: ‘Arial’ for body test and ‘Times New Roman’ for headers.
- Date Format: Dates are to be written in the format of DD/MM/YYYY (E.g. 01/01/2024)
- Requirement Identification: Each requirement is uniquely identified for easy reference.

1.3 Intended Audience and Reading Suggestions

This Software Requirements Specification (SRS) document is intended for various stakeholders involved in the development and deployment of HomeGoWhere:

- **Developers**: To gain a comprehensive understanding of the application's functionality and technical requirements for efficient implementation, developers should pay close attention to the System Features and External Interface Requirements to ensure seamless integration and proper implementation of features.

Readers are advised to start with the overview sections to get an overall understanding of the product's purpose and scope before diving into sections pertinent to their roles.

1.4 Product Scope

The current renting process in Singapore is disorganised and inefficient, presenting significant challenges for both property owners and tenants. Property owners face a lack of streamlined options to advertise their properties, relying primarily on word of mouth, real estate agents, or multiple property listing portals. Additionally, managing tenants is cumbersome, with communication often conducted through informal messaging apps like WhatsApp and rent payments requiring manual tracking. On the tenant side, finding a suitable flat is tedious and time-consuming. Extensive research is needed to evaluate the desirability of different locations, and tenants often have to sift through numerous property listing portals or engage real estate agents which can be costly. Furthermore, interactions with property owners are frequently unstructured, typically carried out through informal messaging platforms. These pain points underscore the need for an all-in-one solution, which HomeGoWhere aims to provide.

HomeGoWhere is a mobile application that seeks to simplify the renting process for both property owners and tenants. It offers an all-in-one platform for owners to efficiently manage their properties and tenants, allowing owners to advertise their property and track rental payments. For tenants, HomeGoWhere simplifies the property search by consolidating all relevant details into a single, comprehensive page, enabling more informed decision-making. Additionally, the app includes an integrated chat feature to facilitate seamless and direct communication between tenants and owners.

1.5 References

Spring Boot: https://docs.spring.io/spring-boot/reference/index.html
React Native: https://reactnative.dev/docs/getting-started
Expo: https://docs.expo.dev/
Postgresql: https://www.postgresql.org/docs/

2. Overall Description

2.1 Product Perspective

2.1.1 System Overview Diagram

A system overview diagram (Figure 2.1.1) and a system architecture diagram (Figure 2.1.2) is provided to illustrate the interconnections and external interfaces.

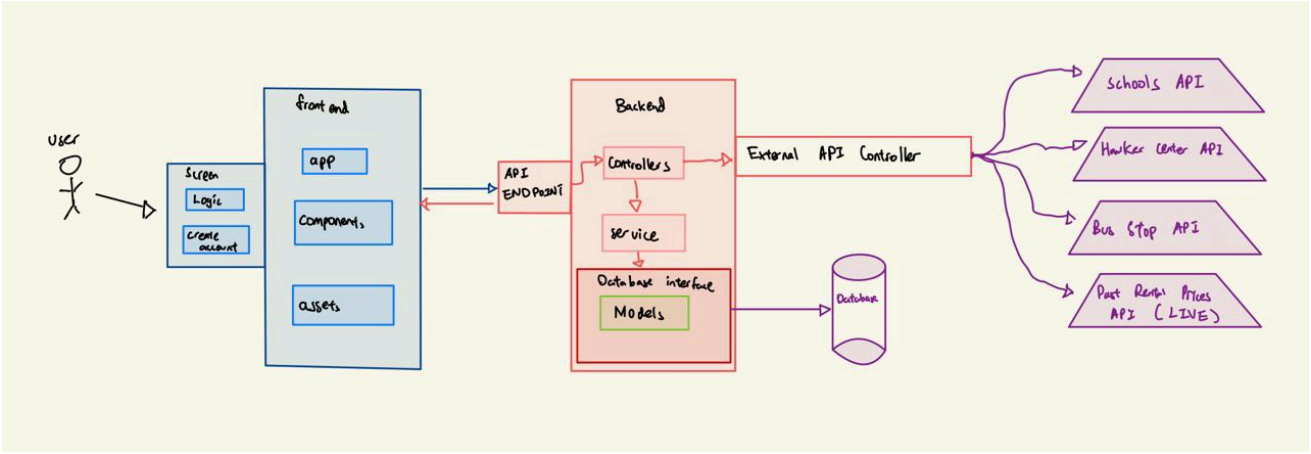


Figure 2.1.1: System Overview Diagram

2.1.2 System Architecture Diagram

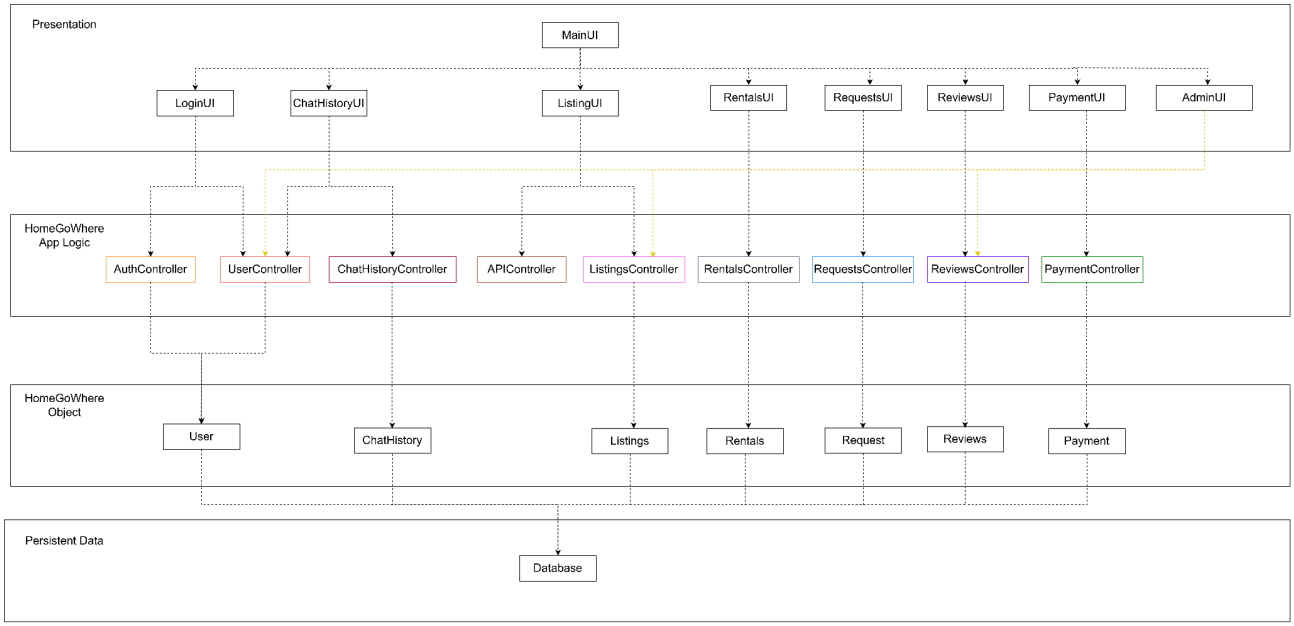


Figure 2.1.2: System Architecture Diagram

Presentation Layer

This layer is largely responsible for Users to interact with HomeGoWhere. The different User Interfaces (UI) will then call the corresponding controllers to run the Application logic.

This layer consists of:

- **MainUI**
  Serves as the main user interface, where users select specific functions. MainUI then calls the appropriate UI — such as LoginUI, ListingUI, ChatHistoryUI, RentalsUI, RequestsUI, ReviewsUI, PaymentUI or AdminUI — to execute the chosen functionality.

- **LoginUI**
  Allows users to create an account and log in.

- **ChatHistoryUI**
  Manages chat functionalities and interfaces through ChatHistoryController.

- **ListingUI**
  Provides property listing views through ListingsController.

- **RentalsUI**
  Handles rental management and interacts through RentalsController.

- **RequestsUI**
  Manages termination requests through RequestsController.

- **ReviewsUI**
  Enables users to leave and manage reviews through ReviewsController.

- **PaymentUI**
  Handles payment-related features through PaymentController.

- **AdminUI**
  Used by admins for administrative tasks through ListingsController, ReviewsController and UserController

App Logic Layer

This layer contains all the controller classes that will provide the presentation layer with its corresponding services. The controller classes will request for entities from the Object Layer when required in order to run its logic.

This layer consists of:

- **AuthController**
  AuthController is called by LoginUI for Users to log in to the application. This controller includes the log in and sign up methods enclosed within its logic.

- **UserController**
  UserController is called by LoginUI, ChatHistoryUI and AdminUI and contains the logic for user management services.

- **ChatHistoryController**
  ChatHistoryController is called by ChatHistoryUI and contains the logic for sending and receiving messages between users.

- **APIController**
  APIController is called by ListingUI and contains the logic to handle external and government data API calls, including data processing.

- **ListingsController**
  ListingsController is called by ListingUI and contains the logic for creating property listings and search functions to return listing results and its corresponding information.

- **RentalsController**
  RentalsController is called by RentalsUI and handles rental management, including the sending of rental offers.

- **RequestsController**
  RequestsController is called by RequestsUI and contains the logic for creating and handling termination requests.

- **ReviewsController**
  ReviewsController is called by ReviewsUI and AdminUI, and contains the logic for submitting reviews, editing reviews, flagging of reviews and processing of reviews.

- **PaymentController**
  PaymentController is called by PaymentUI and contains the logic for managing payment processing and history.

App Object Layer

This layer contains the entity classes that will be called by the App Logic Layer to implement its logic. The entity classes are stored in the database of the persistent layer.

This layer consists of:

- **User**
  User contains information about a user, including userId, name, email, contract, photourl, flagged and password.

- **ChatHistory**
  ChatHistory contains the information about messages, rental offer and termination requests sent between Users in a chat, including messageid, recieverid, senderid, message, date, rentalid and requestid.

- **Listings**
  ChatHistory contains the information about listings posted by owners, including listingid, tenantuserid, rentalprice, depositprice, rentaldate, leaseexpiry, paymenthistory and status.

- **Rentals**
  Rentals contains the information about a rental contract between an owner and a tenant, including rentalid, listingid, tenantuserid, rentalprice, depositprice, rentaldate, leaseexpiry, paymenthistory and status.

- **Request**
  Request contains the information about a termination request, including requestid, rentalid, status and refundamount

- **Reviews**
  Reviews contains the information of a review of a user by another user, including reviewid, userid, reviewerid, rating, title, text and flagged

- **Payment**
  Payment contains the information of a payment made by a tenant, including paymentid, rentalid, amount and date

Persistent Data Layer

This layer contains the database that will store all of the entities.

2.1.3 Class Diagrams

The Class Diagrams (Figure 2.1.3 and 2.1.4) illustrates the interactions between various classes and highlights the essential functionalities that HomeGoWhere must perform

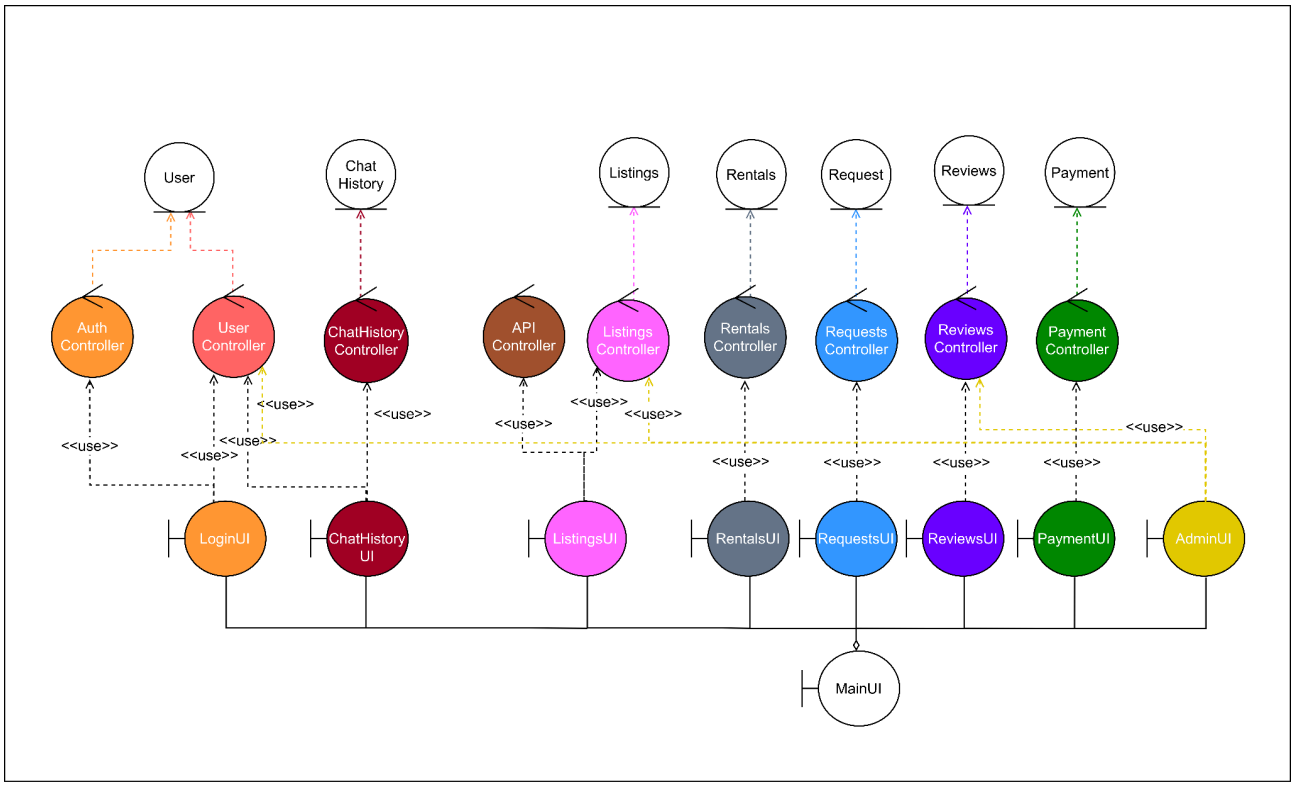


Figure 2.1.3: Class Diagram

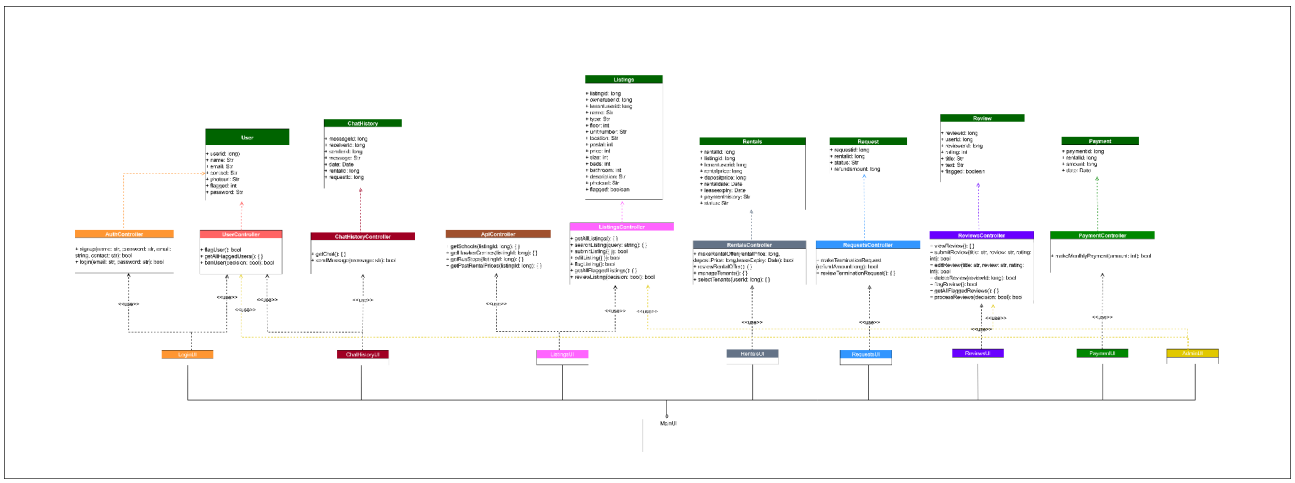


Figure 2.1.4: Key Classes Diagram

2.2 Product Functions

2.2.1 Use Case Diagram

The Use Case Diagram (Figure 2.2.1) outlines the key functionalities offered by HomeGoWhere.

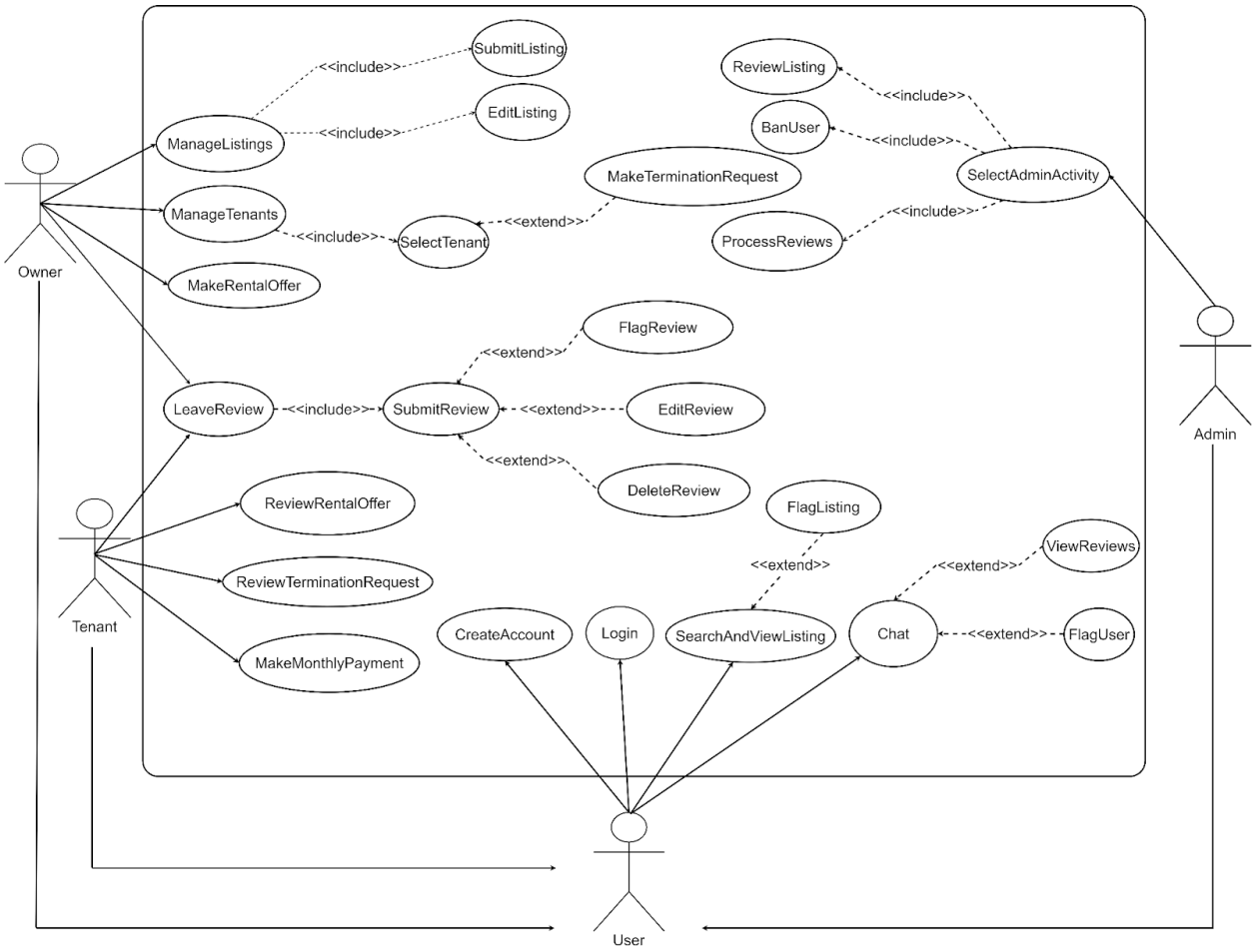


Figure 2.2.1: Use Case Diagram

2.2.2 Key Product Functions

- **For Users:**
    - Login and Account Creation
    - Edit profile
    - Search and View Listings
    - Chat
    - Leave and Manage Reviews
    - Flag Listings, Users or Reviews
- **For Owners:**
    - Submit and Manage Listings
    - Manage Tenants
        - Make Rental Offers
        - Make Termination Requests
        - Track Rent Payments
- **For Tenants:**
    - Review Rental Offer
    - Make Monthly Payments
    - Review Termination Request
- **For Admins:**
    - View flagged Listings, Reviews and Users
    - Take follow-up actions
        - Ban Users
        - Delete Reviews
        - Delete Listings

2.2.3 Dialog Map

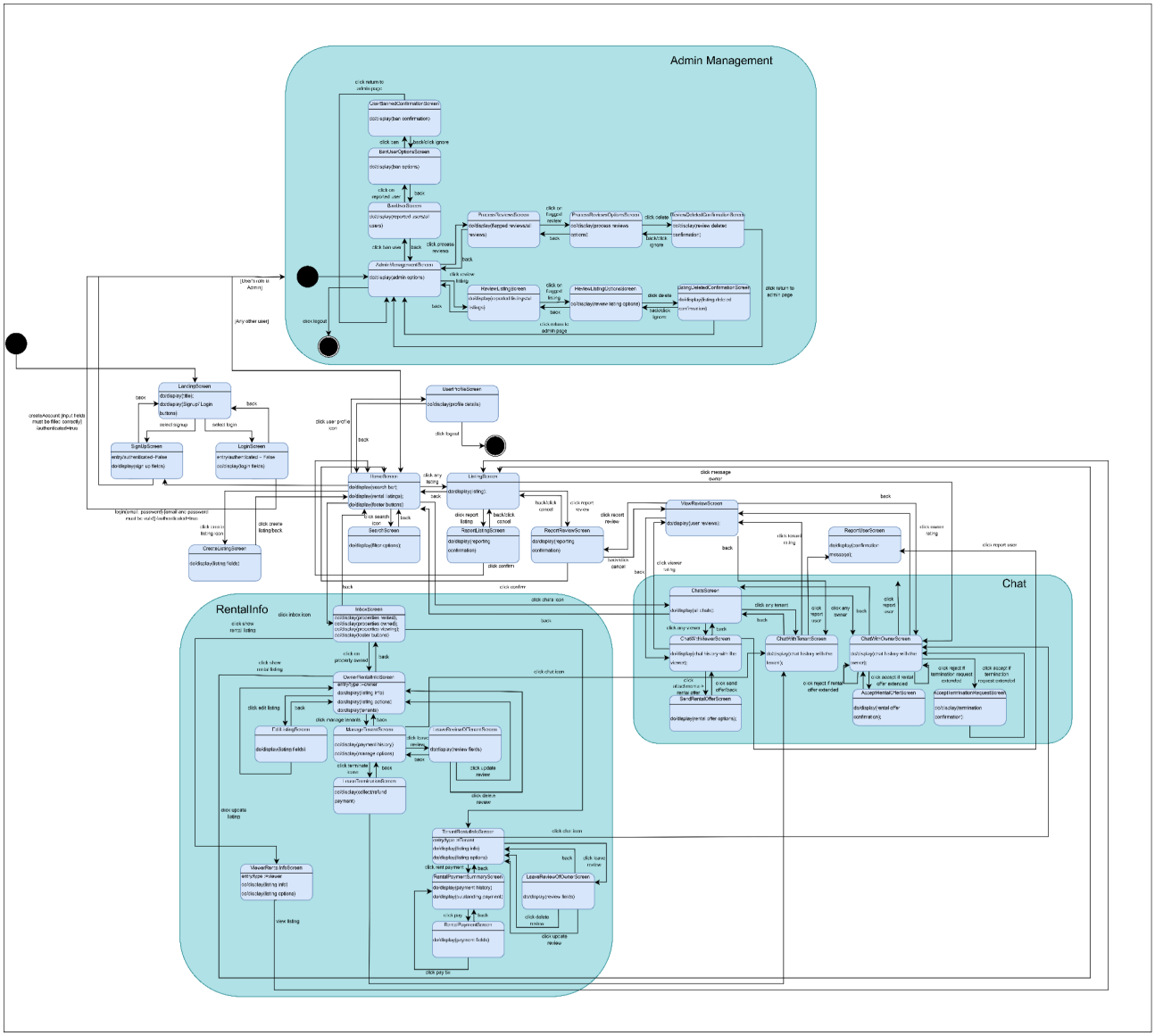


Figure 2.2.2: Dialog Map

2.3 User Classes and Characteristics

There are 3 primary users of HomeGoWhere — Tenant, Owner and Admin. Each type of user interacts with HomeGoWhere to use functions specific to their role as depicted in the Use Case Diagram (Figure 2.5).

- **Tenants**
  Tenants comprise two groups: potential tenants seeking suitable properties to rent, and active tenants with ongoing rental contracts. They use HomeGoWhere to explore property listings, manage rental contracts, and communicate with property owners.

- **Owners**
  Owners are property owners who list and manage their rental properties on the app. They utilise HomeGoWhere to advertise properties, manage tenant interactions, and track rental payments.

- **Admin**
  Admins are staff members responsible for maintaining the safety and integrity of the HomeGoWhere platform. They monitor activity, address user reports, and ensure a smooth and secure user experience.

2.4 Operating Environment

HomeGoWhere is a mobile application designed for iOS and Android devices.

2.5 Design and Implementation Constraints

Overview of Tech Stack Used:

- Frontend
  - React Native
  - Expo
  - JavaScript
- Backend
  - Spring Boot (Java)
- Database
  - Postgresql

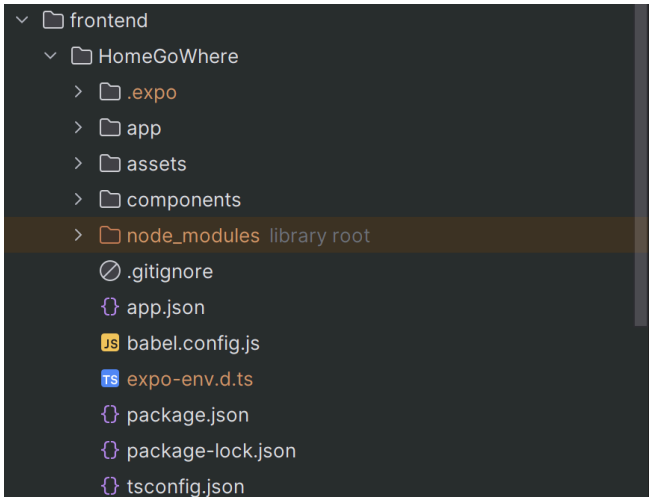
2.5.1 Frontend

Built with **React Native** framework.

Frontend (React Native) mainly consists of the different User Interfaces (app), which are structured and categorised into different UI.

Assets is a folder which contains all of the images used in the app.

Components is a folder where we have our more commonly used items such as buttons.



2.5.2 Backend

Built with **Spring Boot** framework

Models: Contains the Business Objects.

Service: Contains the methods to access/modify the Business Objects.

Repository: Contains the specific Create, Read, Update, Delete (CRUD) by inheriting from JPA repository.

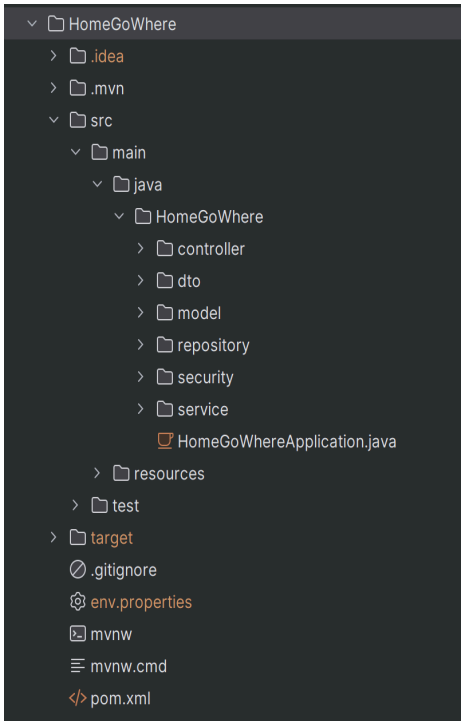
Controllers: Contains all the controllers that provide the detailed implementation of the various services. It also contains REST API endpoints for data communication between the frontend and backend.

Security: Contains all the security files in charge of ensuring logging in is secure with the use of JWT Authentication.

DTO: Contains all the Data Transferable objects of each model in order to allow communication between the API and our database server without exposing too much information

Target: Contains all classes built

Resources: Contains the file with all used live API keys



2.5.3 Database

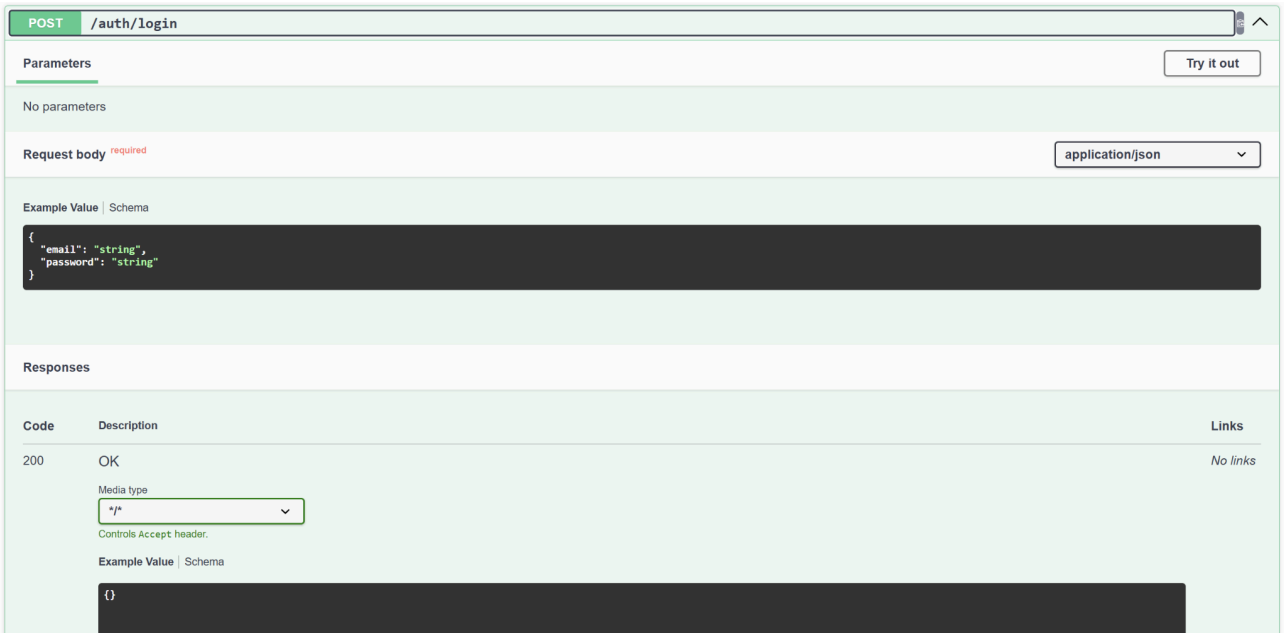
The database for HomeGoWhere is built using **PostgreSQL**, which efficiently handles structured and relational data.

2.6 User Documentation

HomeGoWhere includes a comprehensive set of documentation to guide developers. The documentation ensures a clear understanding of the setup, usage, and integration of the different components of the application.

- **README File**
  The main repository features a README file that provides an overview of HomeGoWhere, including setup instructions, architecture and repository structure.

- **API Documentation**
  This documentation is located in http://localhost:8080/swagger-ui/index.html#/ after running the backend.



2.7 Assumptions and Dependencies

HomeGoWhere is dependent on the following government data APIs:

- General information of schools [MOE (Ministry of Education)]
    - To display list of schools near property in “View Listing” screen
    - https://data.gov.sg/datasets/d_688b934f82c1059ed0a6993d2a829089/view

- List of Government Markets Hawker Centres [NEA (National Environment Agency)]
    - To display list of Hawker Centres near property in “View Listing” screen
    - https://data.gov.sg/datasets/d_68a42f09f350881996d83f9cd73ab02f/view

- Bus Stops [LTA DataMall]
    - To display list of Bus Stops near property in “View Listing” screen
    - https://datamall2.mytransport.sg/ltaodataservice/BusStops

- Private Residential Properties Rental Contract [URA]
    - To display list of past rental contract prices in the estate of a private property in “View Listing” screen
    - https://www.ura.gov.sg/maps/api/#private-residential-properties-rental-contract

- Renting Out of Flats [HDB (Housing and Development Board)]
    - To display list of past rental contract prices in the estate of a HDB property in “View Listing” screen
    - https://data.gov.sg/datasets/d_c9f57187485a850908655db0e8cfe651/view

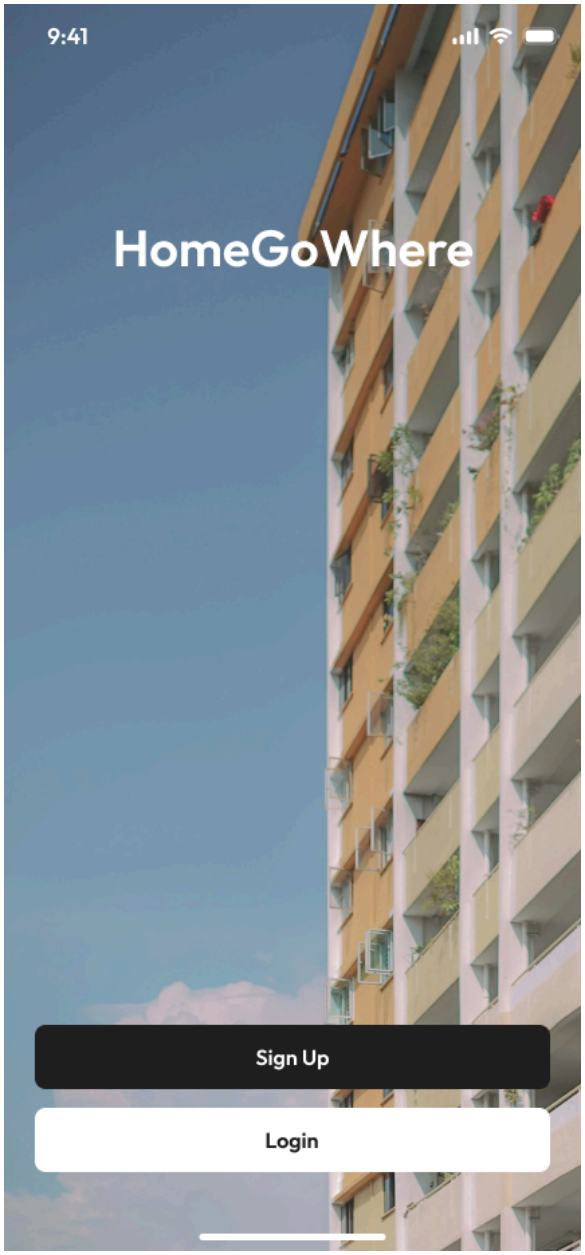
HomeGoWhere is also dependent on the following external APIs:

- OneMap
    - To process and filter data obtained from government data APIs
    - To obtain coordinates of postal codes to be used in interactive map found in “View Listing” screen
    - https://www.onemap.gov.sg/apidocs/apidocs

3. External Interface Requirements

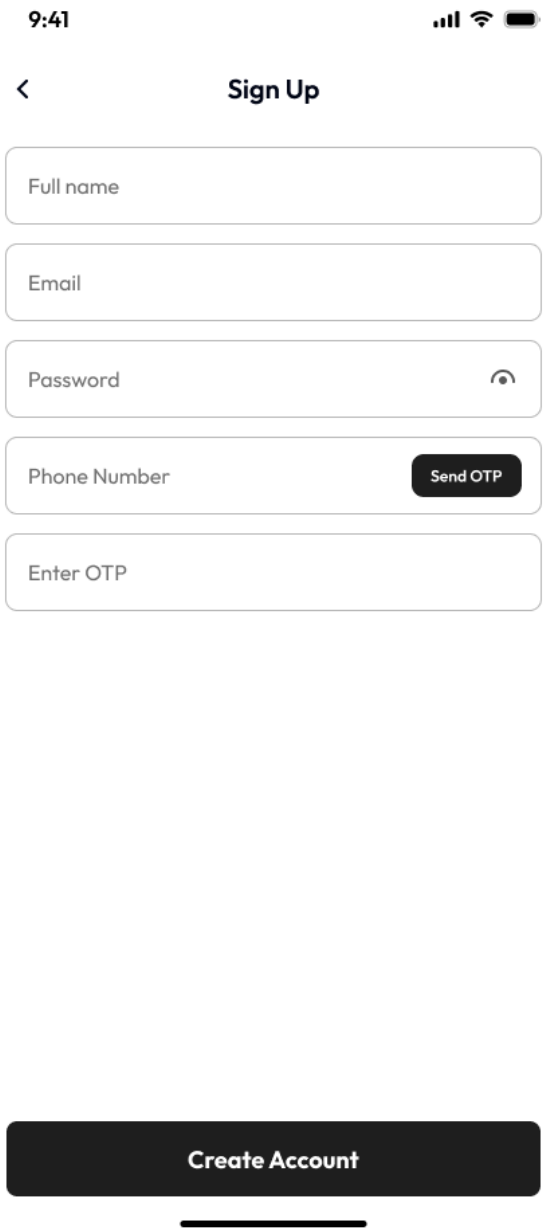
3.1 User Interfaces

3.1.1 Landing



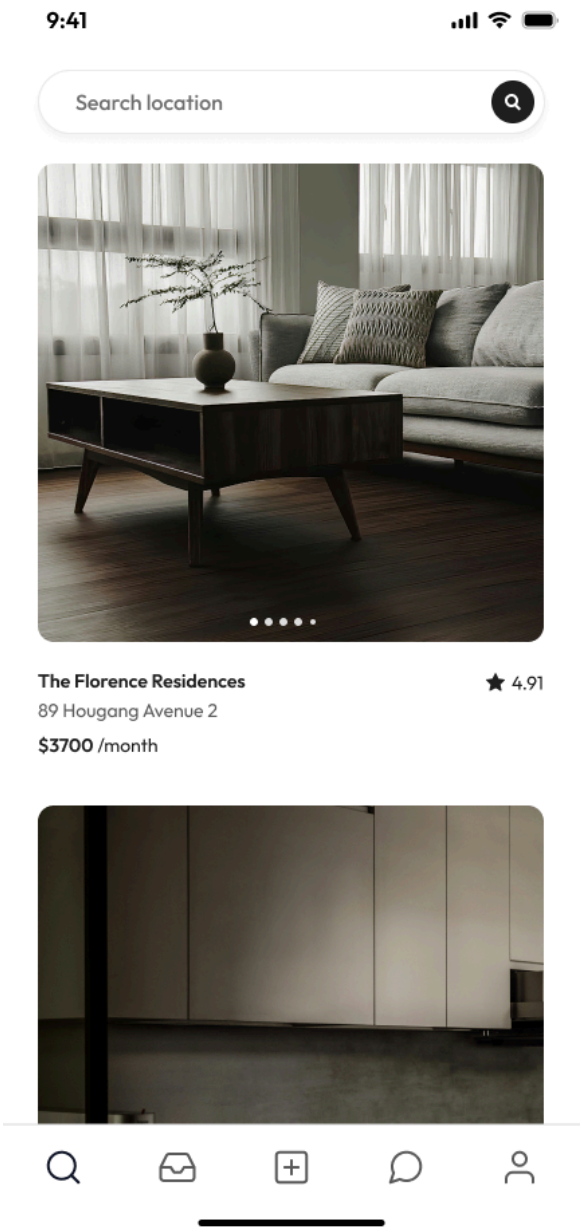
3.1.2 Login

3.1.3 Sign Up

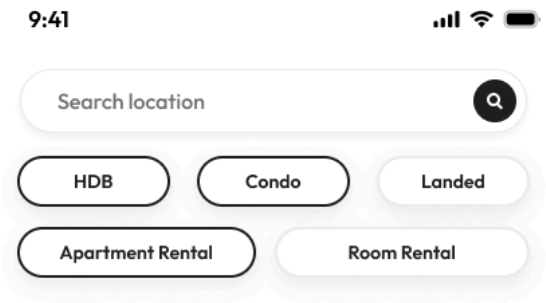

9:41
Sign Up
Full name
Email
Password
Phone Number
Send OTP
Enter OTP
Create Account

3.1.4 Home Screen

The app features 5 tabs in the navigation bar. Search, Inbox, Create New Listing, Chats, Profile (From left to right)



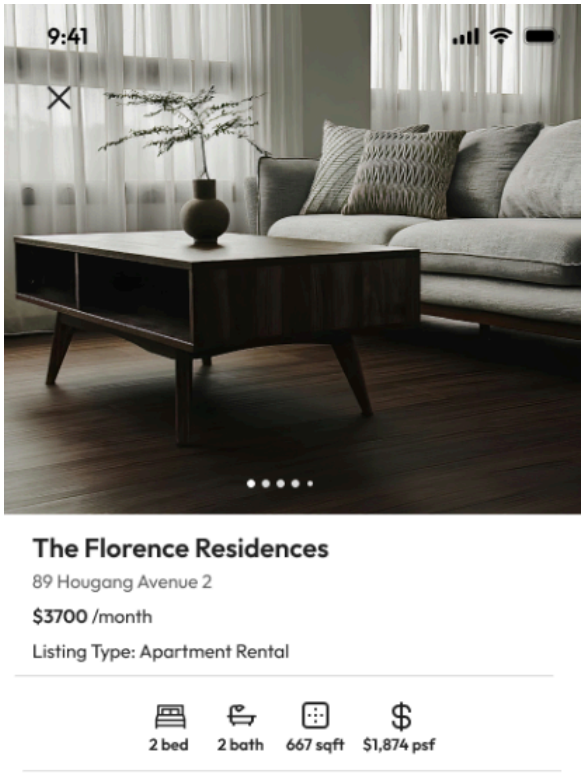
3.1.5 Search Listing



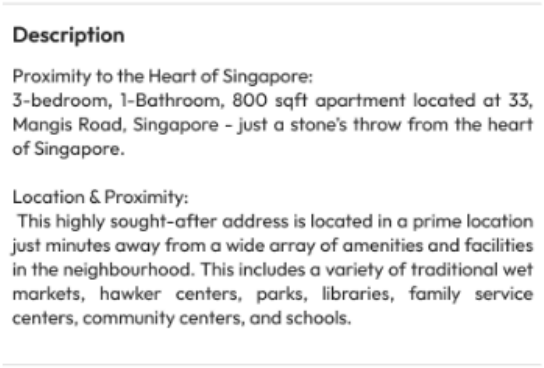
3.1.6 View Listing

"View Home Listing" is one continuous page containing multiple sections.

3.1.6.1 Listing photos and primary details



3.1.6.2 Listing description

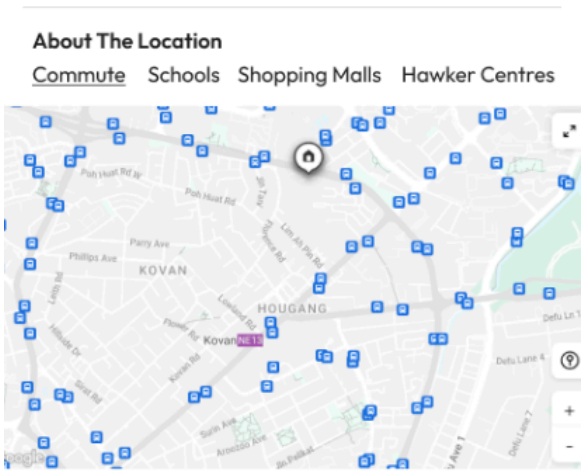


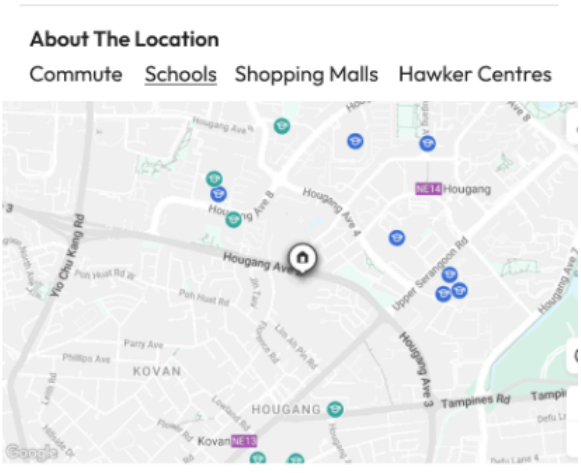
Description

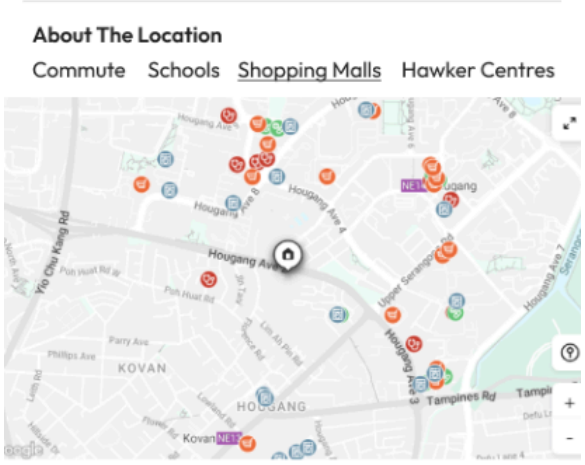
Proximity to the Heart of Singapore:
3-bedroom, 1-Bathroom, 800 sqft apartment located at 33, Mangis Road, Singapore - just a stone's throw from the heart of Singapore.

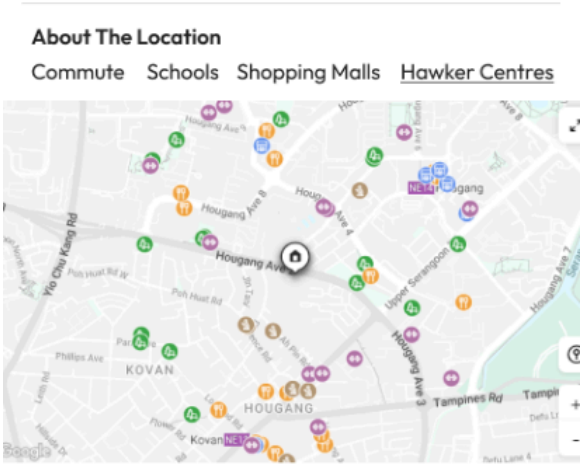
Location & Proximity:
This highly sought-after address is located in a prime location just minutes away from a wide array of amenities and facilities in the neighbourhood. This includes a variety of traditional wet markets, hawker centers, parks, libraries, family service centers, community centers, and schools.

3.1.6.3 Location Details

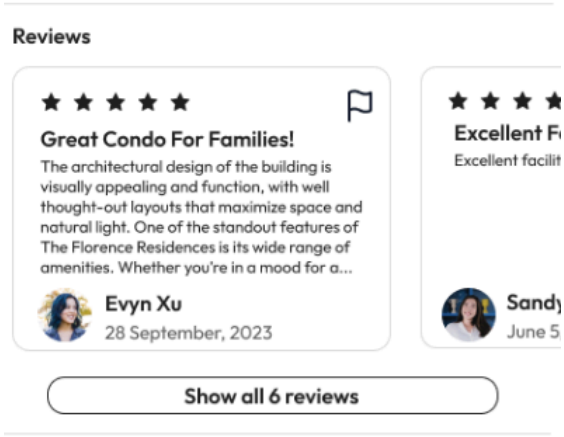


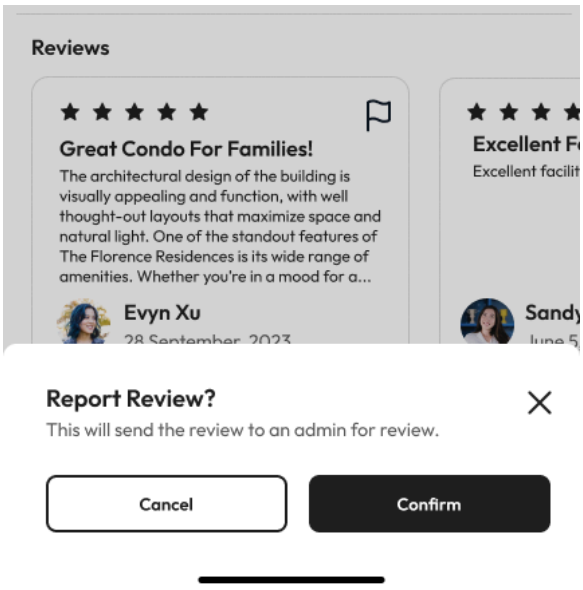


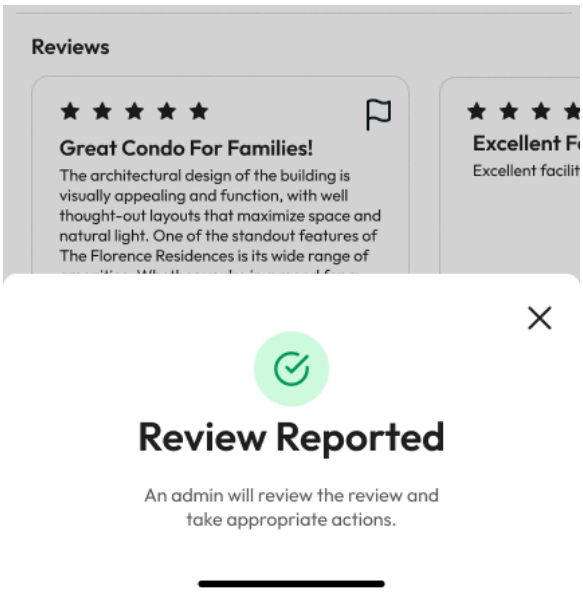




3.1.6.4 Reviews + Report Reviews







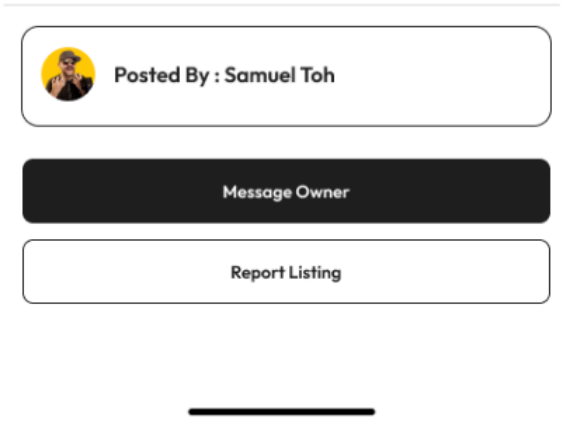
3.1.6.5 Past Price Insights

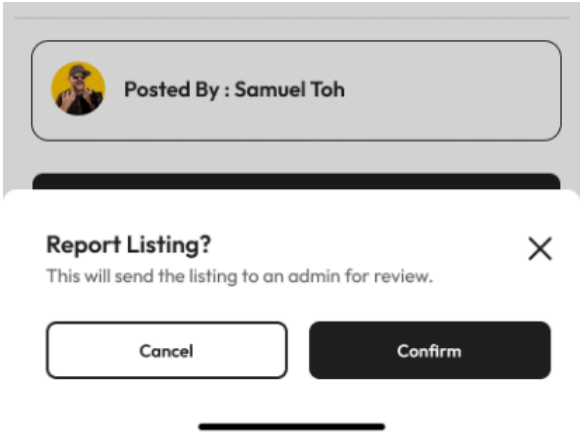
Price Insights

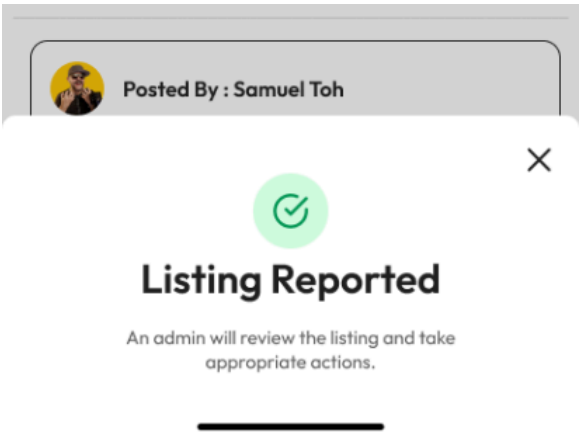
Last Transaction Price
S$3,300
49 Transactions

Date	Unit	Size (sqft)	Price
Jul 2024	#XX-XX	600-700	S$ 3,300
Jul 2024	#XX-XX	600-700	S$ 3,500
Jul 2024	#XX-XX	600-700	S$ 3,400
Jul 2024	#XX-XX	900-1000	S$ 3,600
Jul 2024	#XX-XX	600-700	S$ 3,400

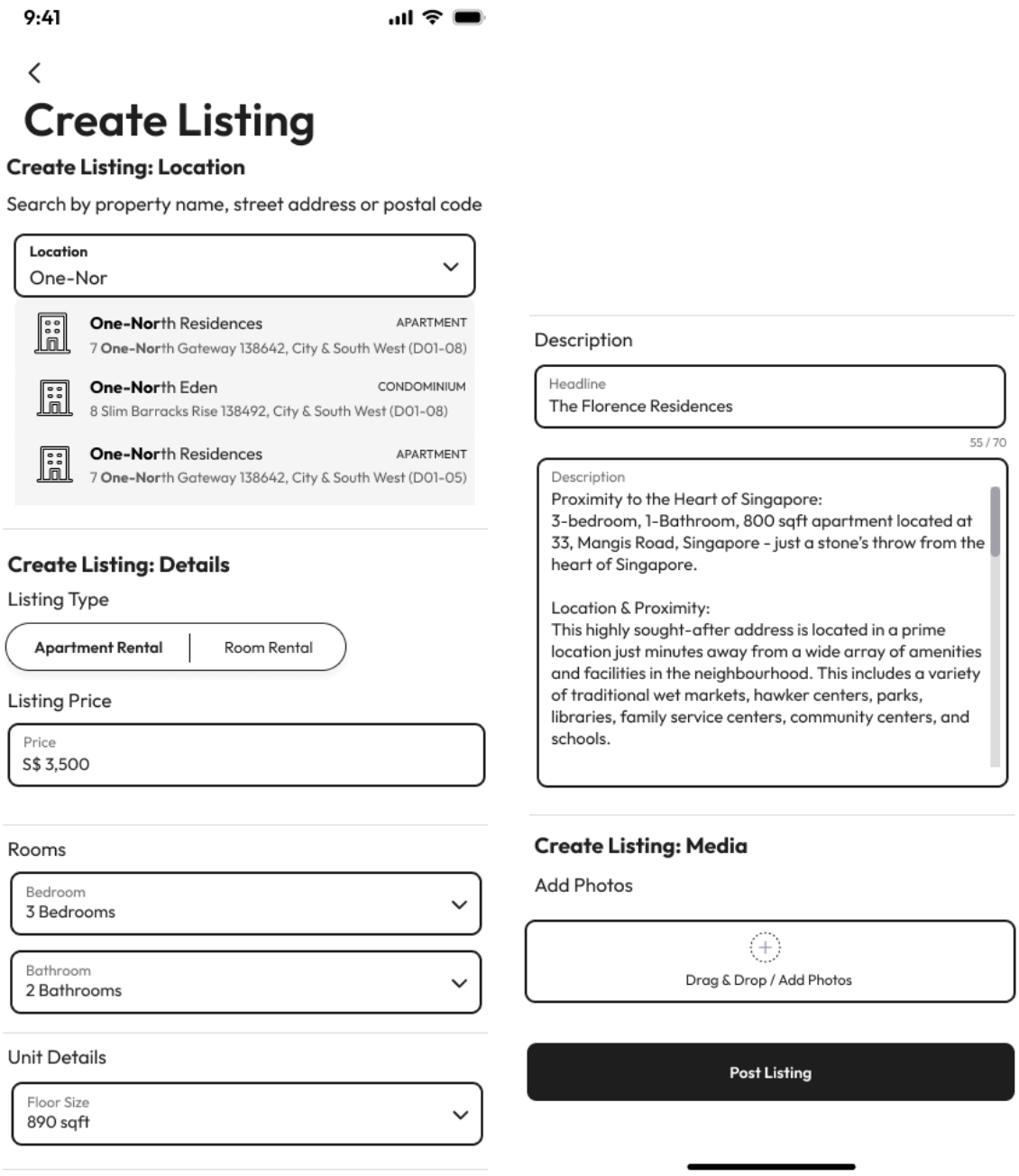
3.1.6.6 Owner Information + Report Listing







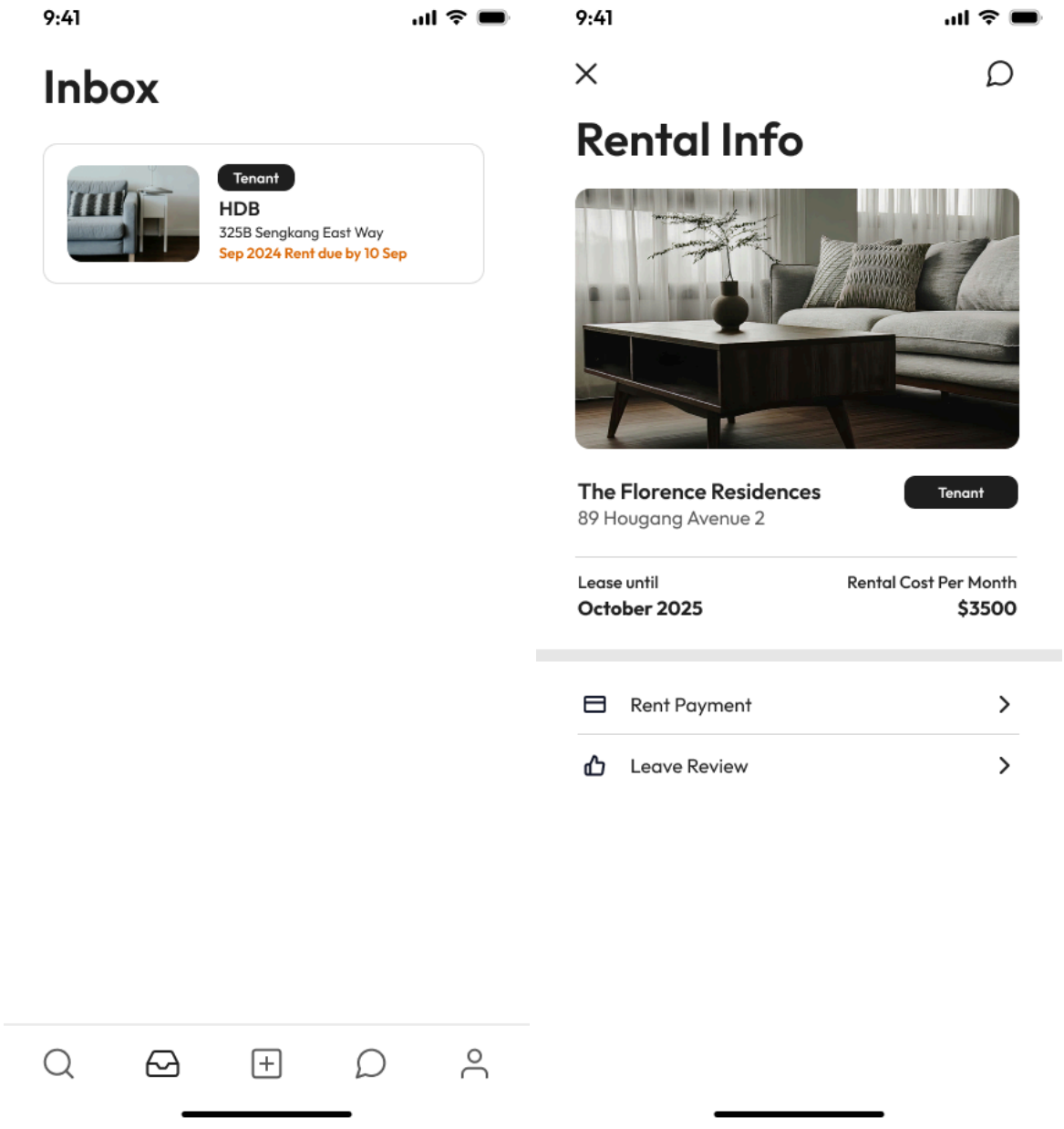
3.1.7 Create New Listing



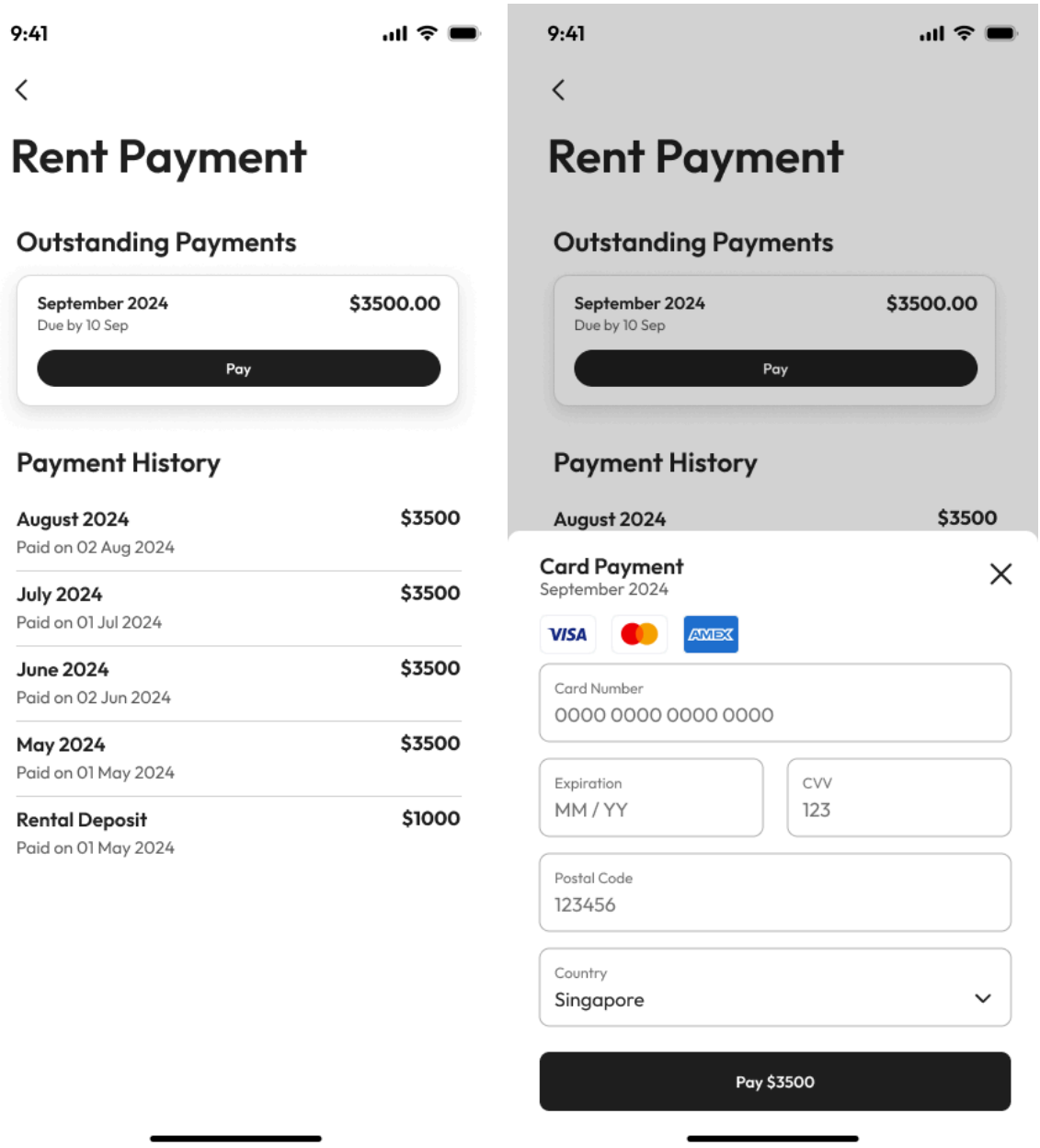
3.1.8 Inbox

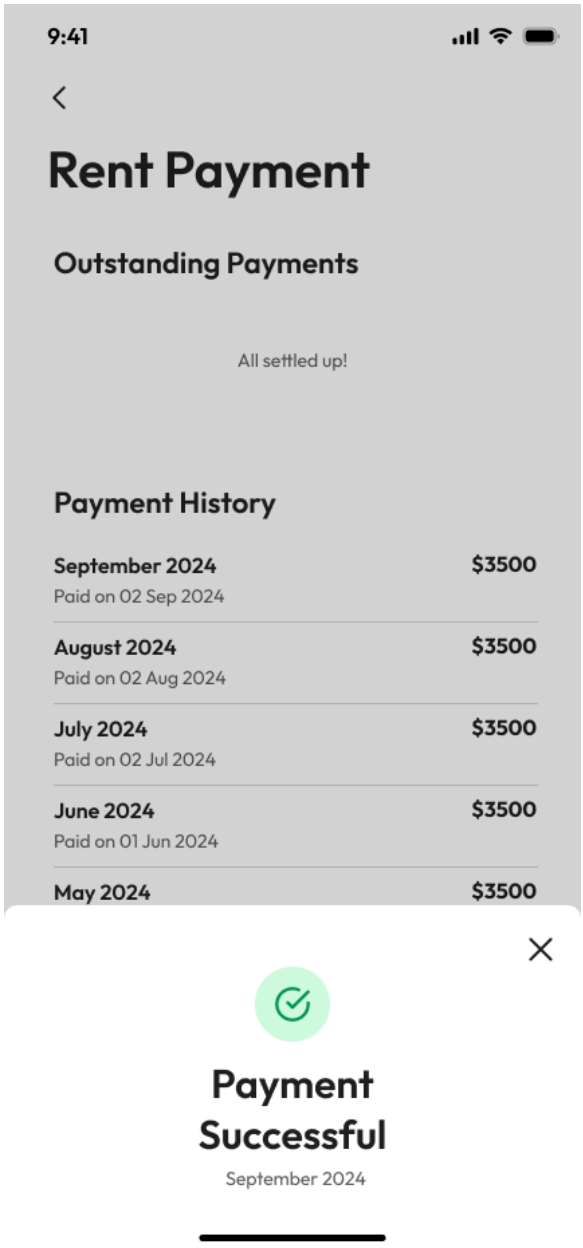
3.1.8.1 Inbox view for a tenant

All active rental units that a tenant is involved with will be displayed in their inbox. They are able to click on each listing card to view additional info and manage their rental.



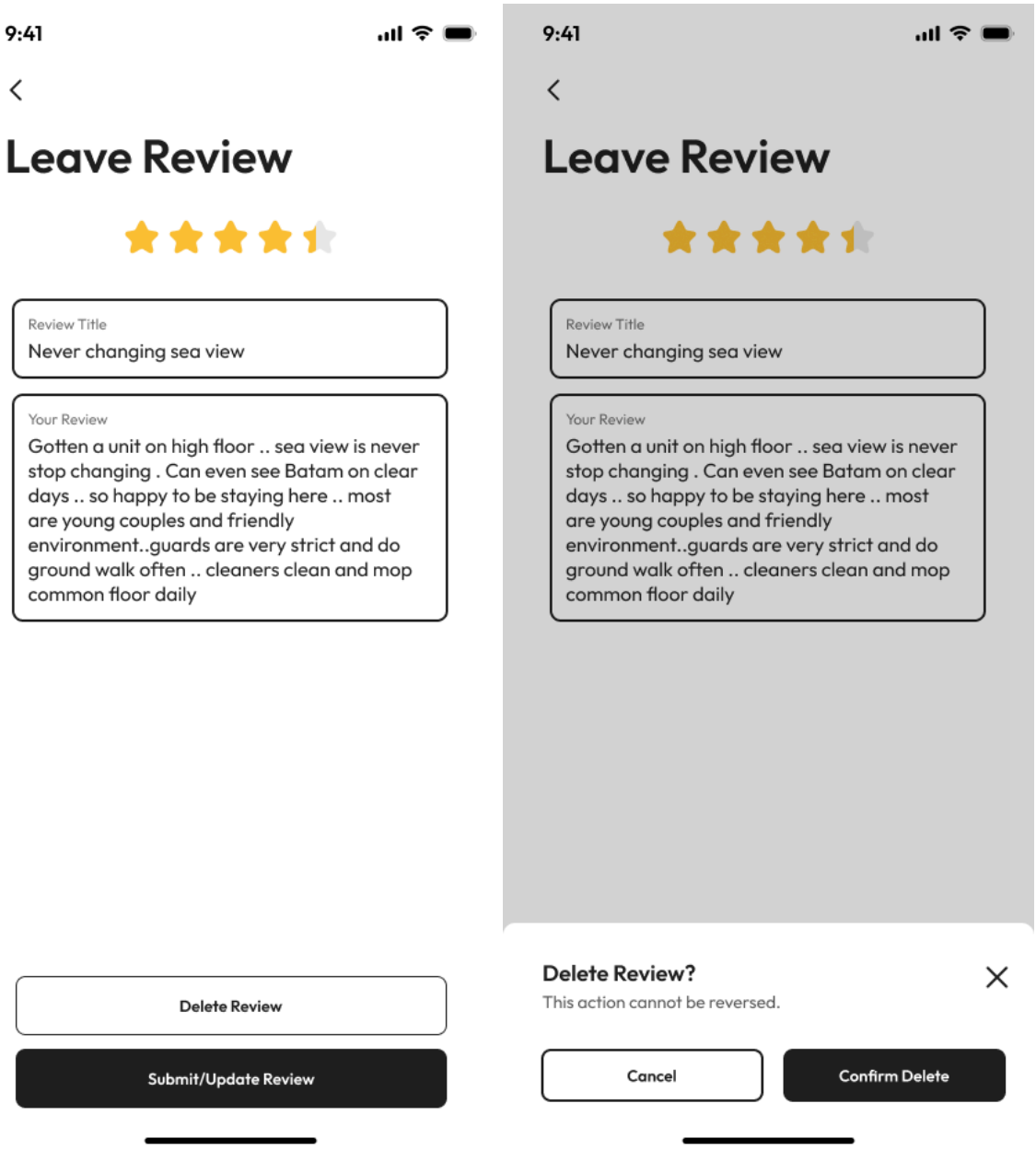
3.1.8.1.1 Rent Payments

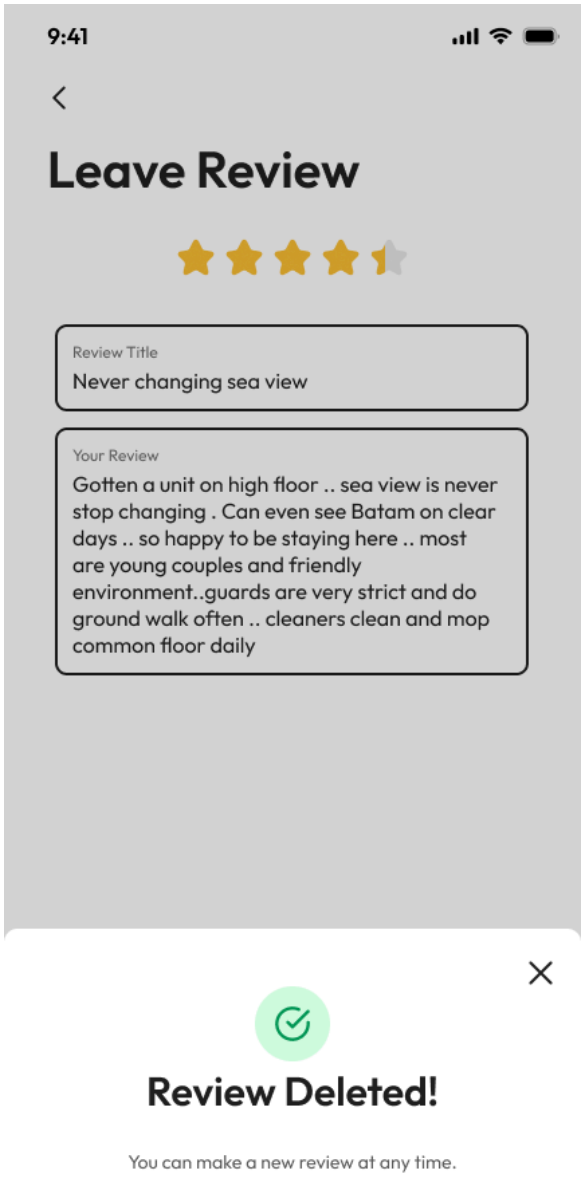



9:41
Rent Payment
Outstanding Payments
All settled up!
Payment History
September 2024
Paid on 02 Sep 2024
$3500
August 2024
Paid on 02 Aug 2024
$3500
July 2024
Paid on 02 Jul 2024
$3500
June 2024
Paid on 01 Jun 2024
$3500
May 2024
$3500

Payment Successful
September 2024

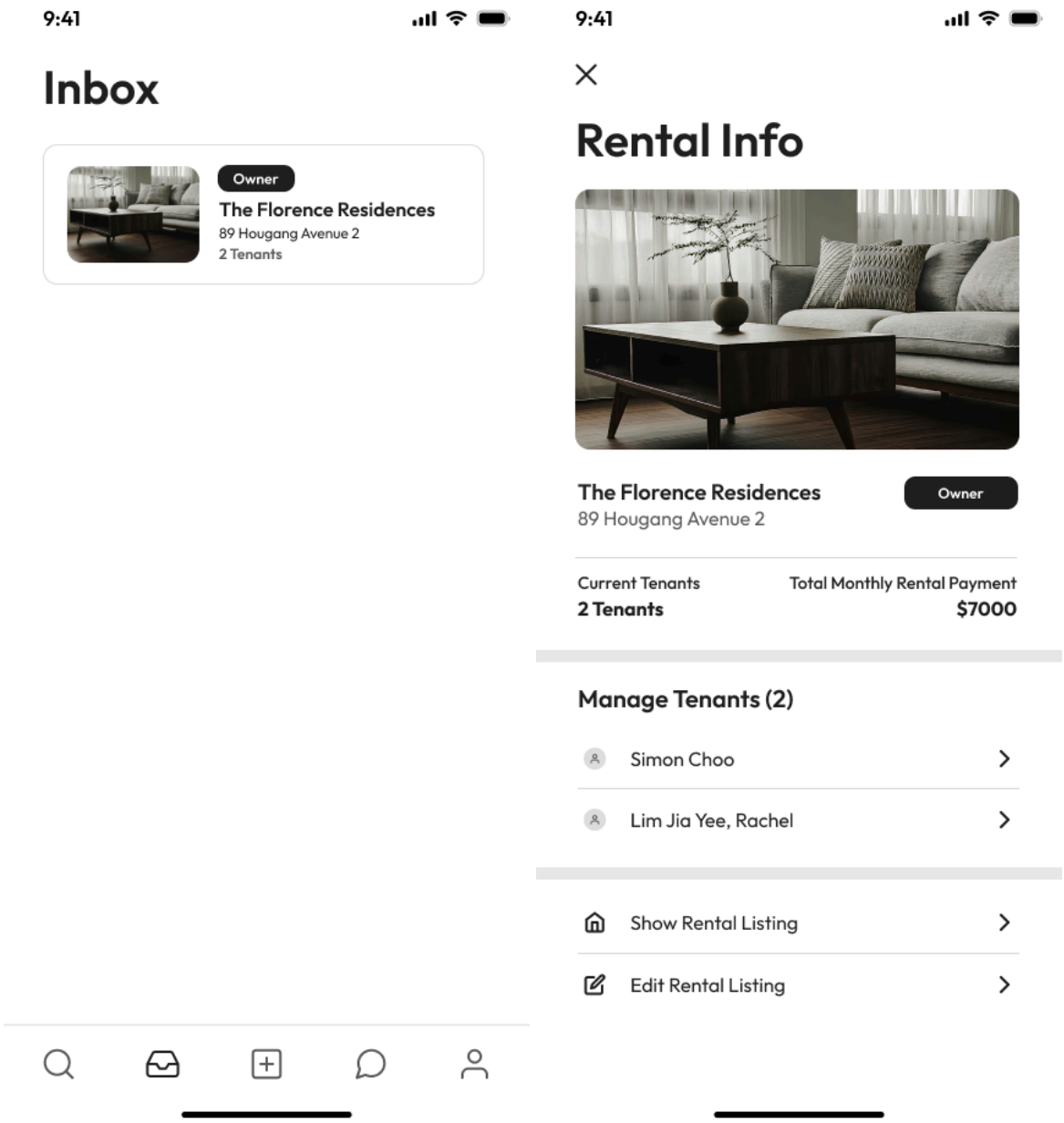
3.1.8.1.2 Leave Review



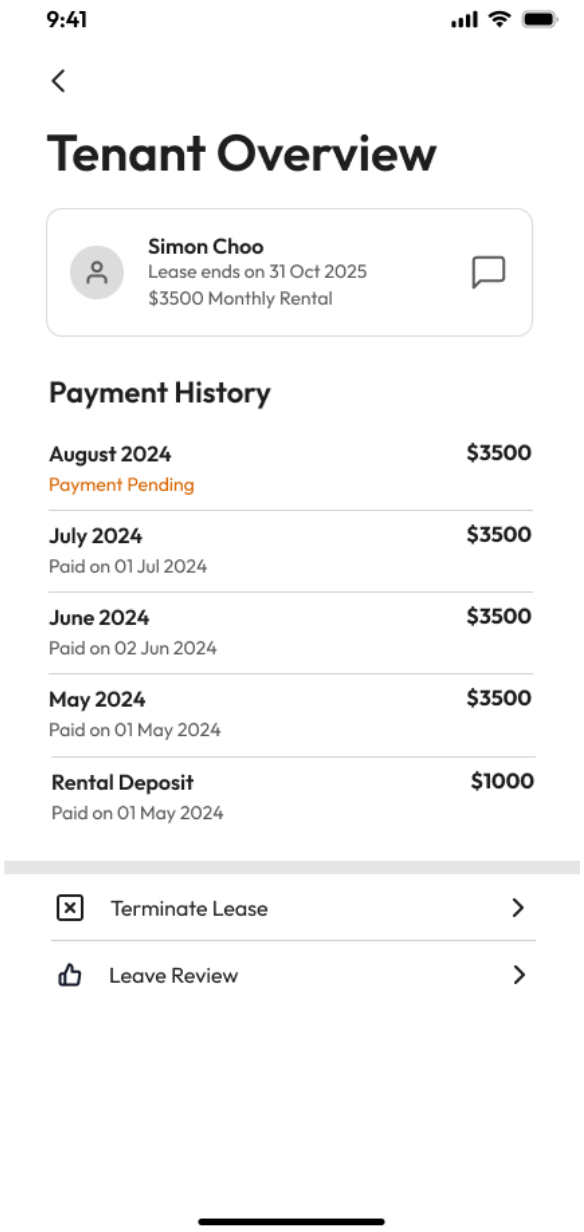

9:41
Leave Review
Review Title
Never changing sea view
Your Review
Gotten a unit on high floor .. sea view is never stop changing . Can even see Batam on clear days .. so happy to be staying here .. most are young couples and friendly environment..guards are very strict and do ground walk often .. cleaners clean and mop common floor daily
Review Deleted!
You can make a new review at any time.

3.1.8.2 Inbox view for an owner

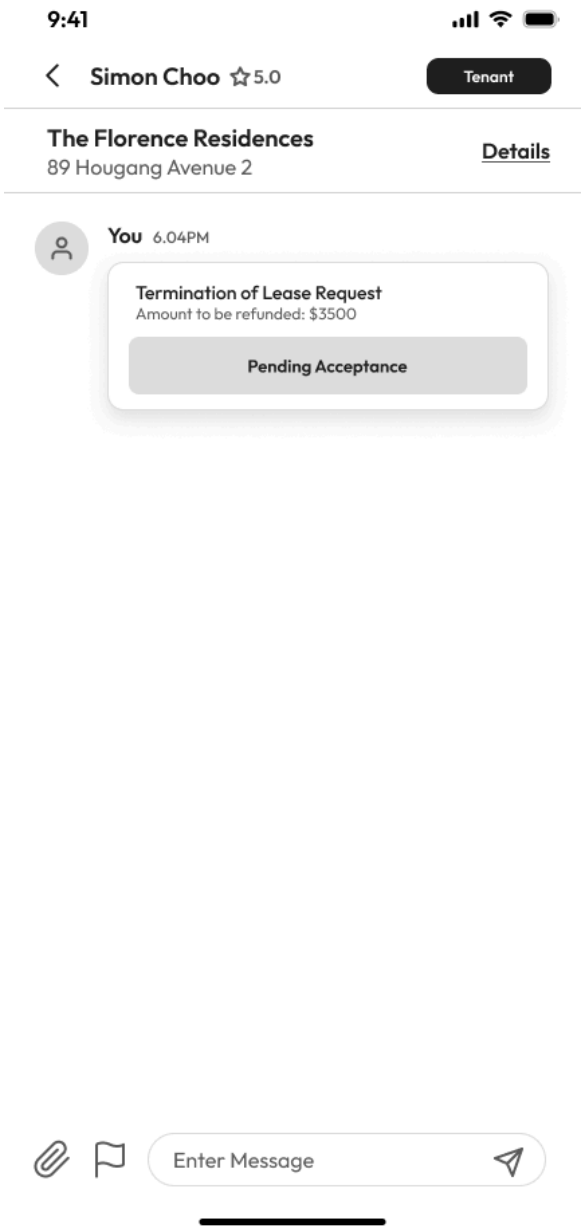
All active rental units that an owner is renting will be displayed in their inbox. They are able to click on each listing card to view additional info and manage their rental.



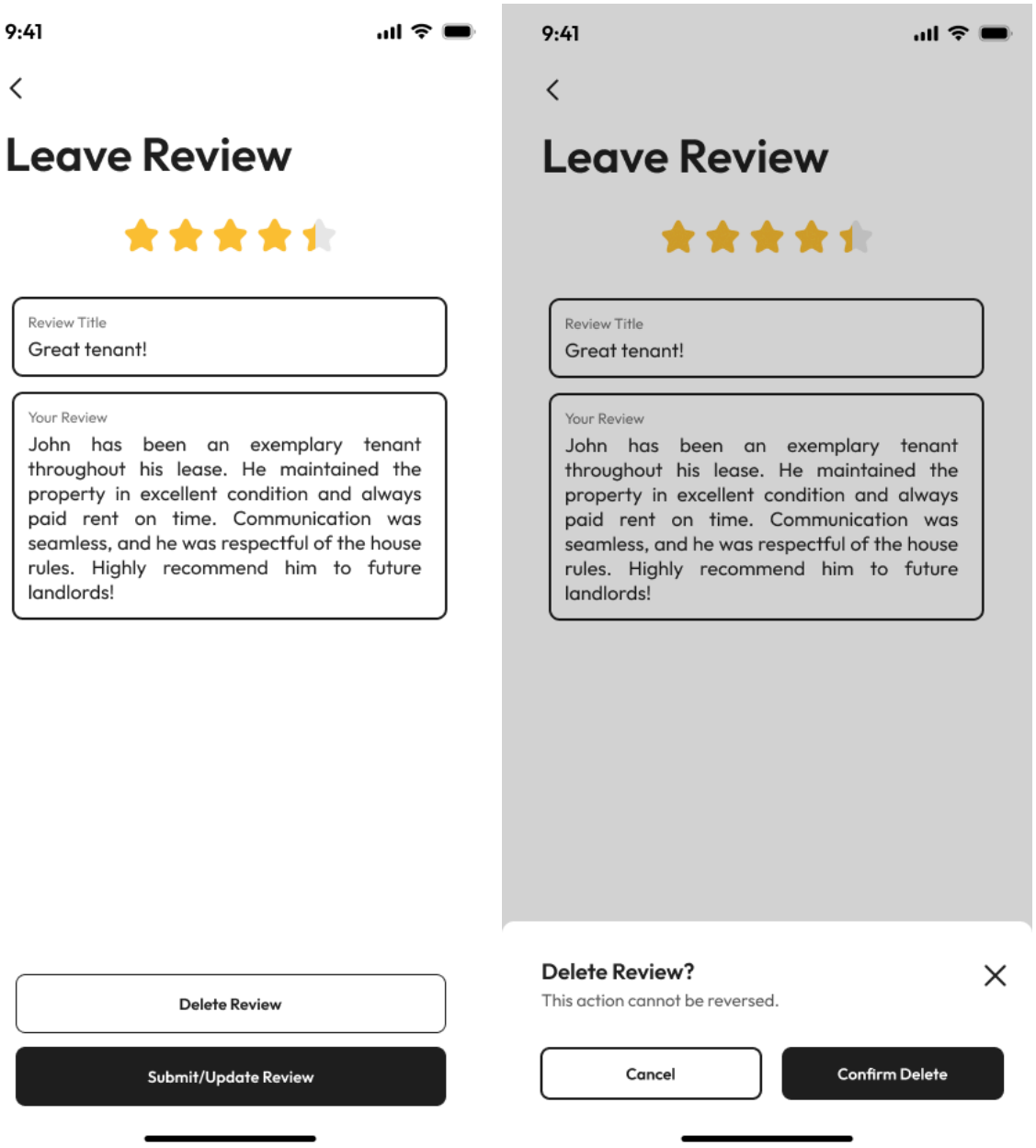
3.1.8.2.1 Manage Tenants

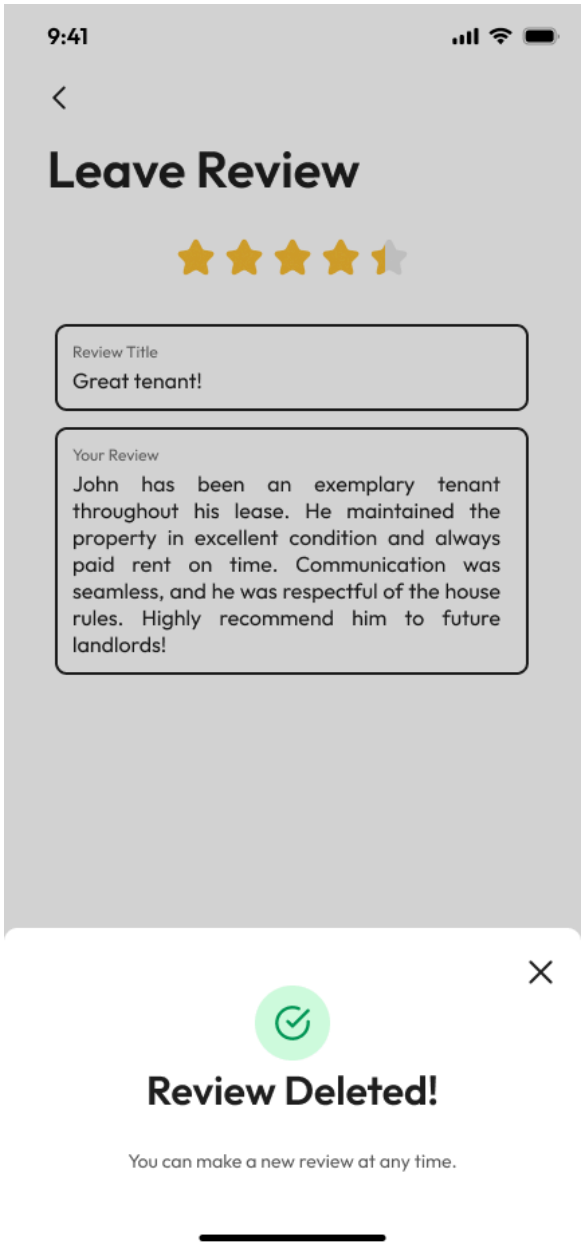


3.1.8.2.2 Terminate Lease

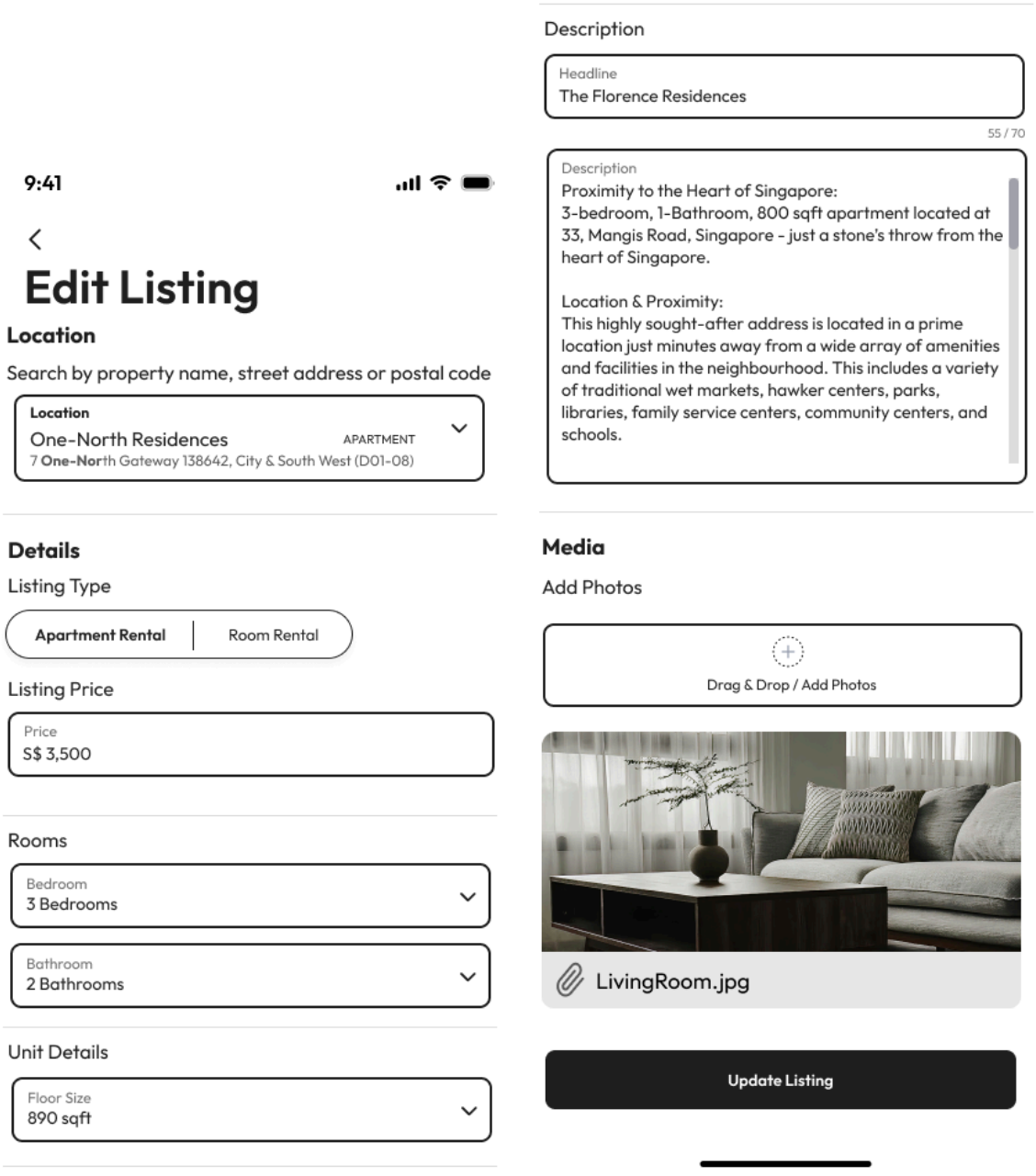


3.1.8.2.3 Leave Review

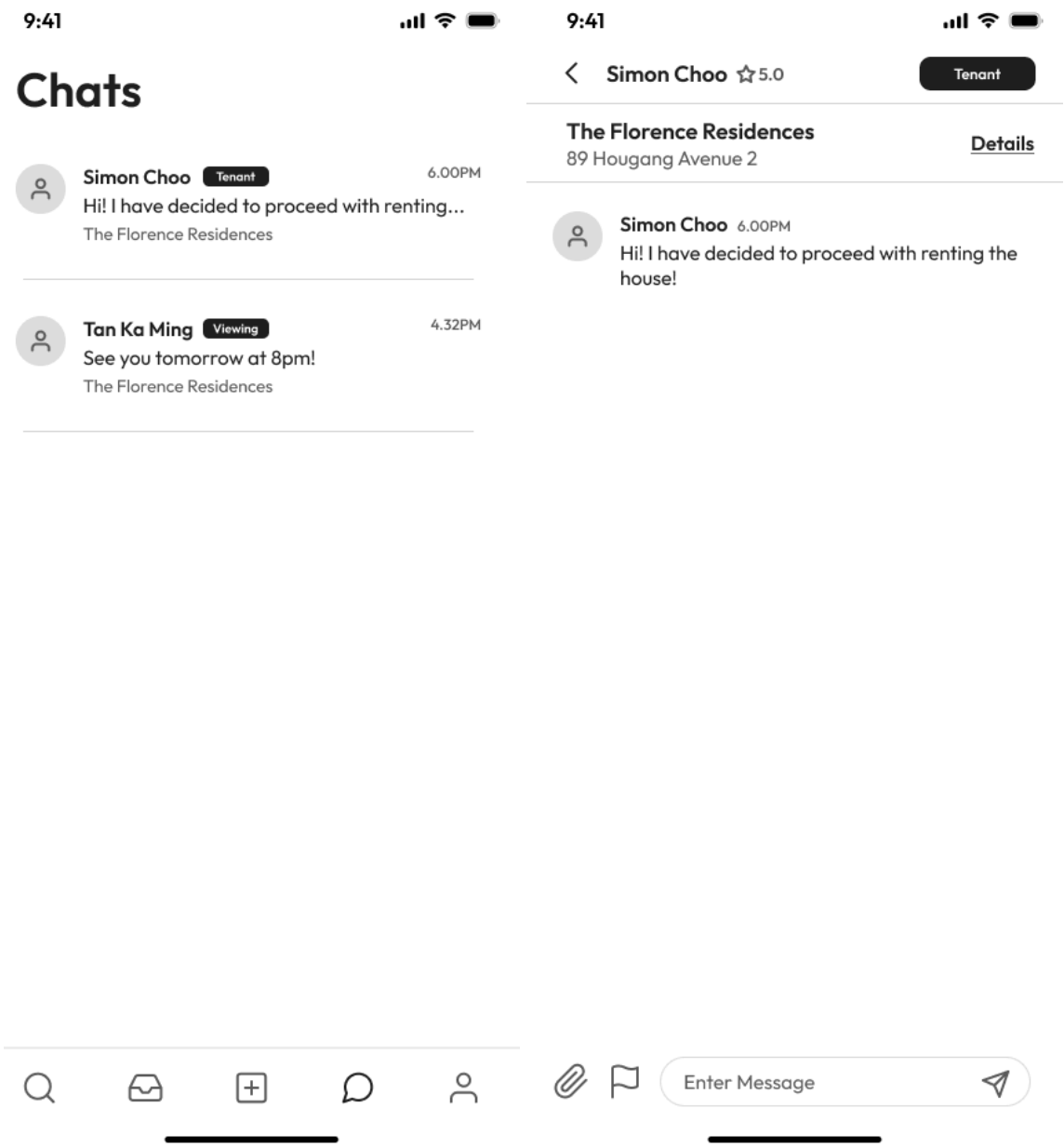



9:41
Leave Review
Review Title
Great tenant!
Your Review
John has been an exemplary tenant throughout his lease. He maintained the property in excellent condition and always paid rent on time. Communication was seamless, and he was respectful of the house rules. Highly recommend him to future landlords!
Review Deleted!
You can make a new review at any time.

3.1.8.2.4 Edit Rental Listing

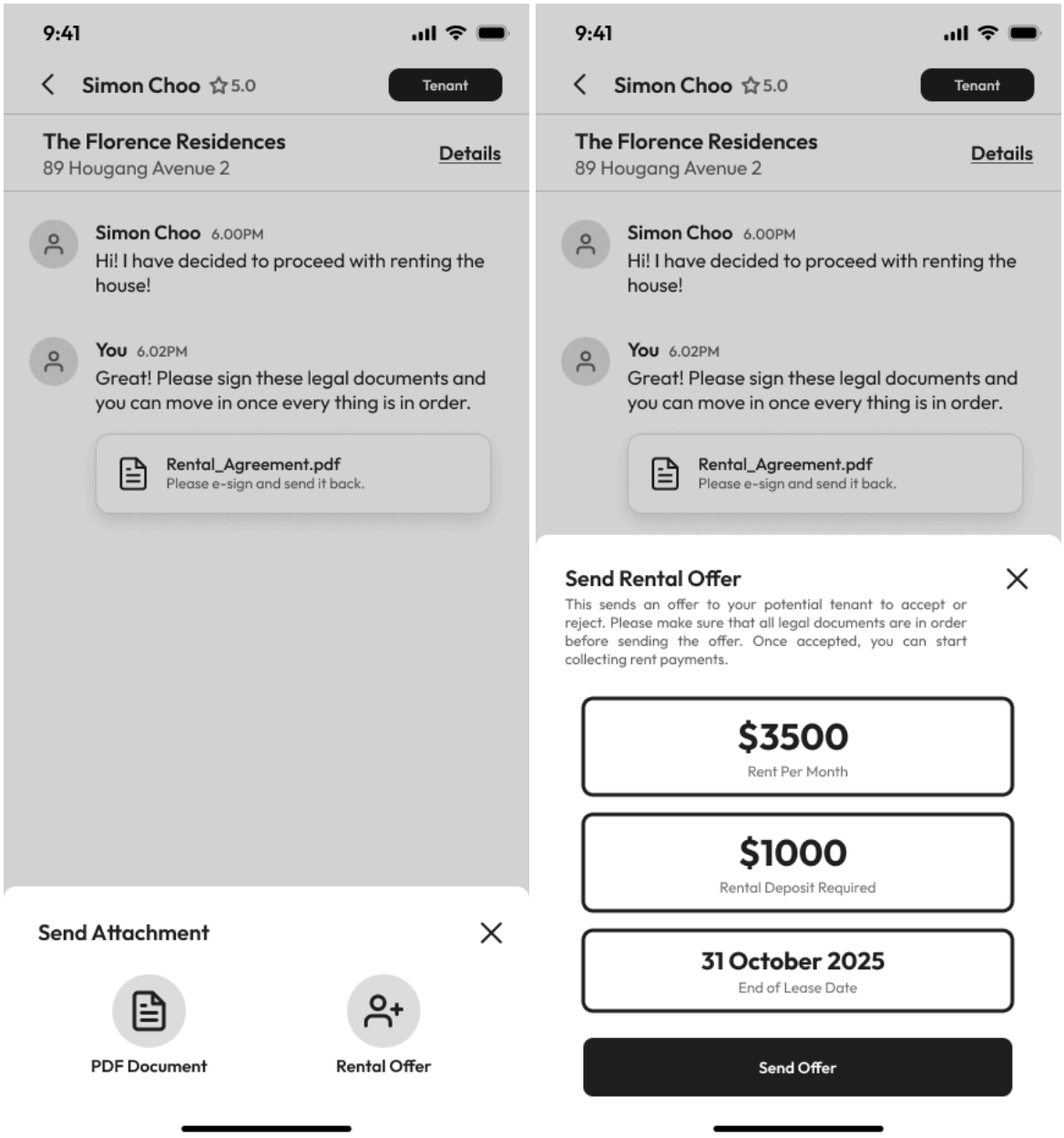


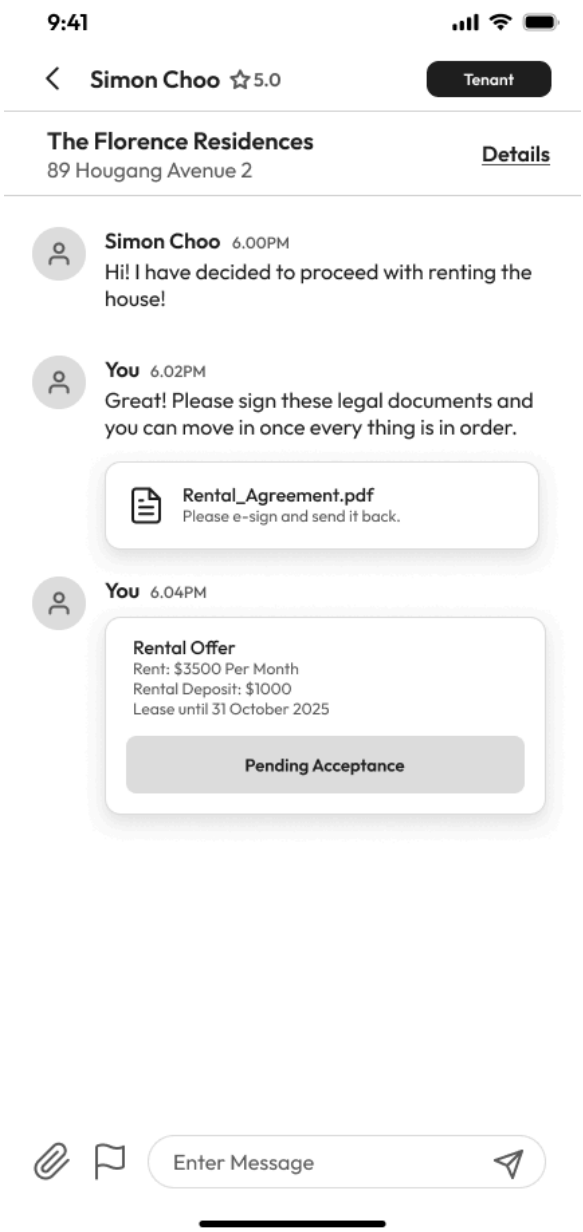
3.1.9 Chats


9:41
Chats
Simon Choo
Tenant
6.00PM
Hi! I have decided to proceed with renting...
The Florence Residences
Tan Ka Ming
Viewing
4.32PM
See you tomorrow at 8pm!
The Florence Residences
9:41
Simon Choo
5.0
Tenant
The Florence Residences
89 Hougang Avenue 2
Details
Simon Choo
6.00PM
Hi! I have decided to proceed with renting the house!
Enter Message

3.1.9.1 Send Rental Offer

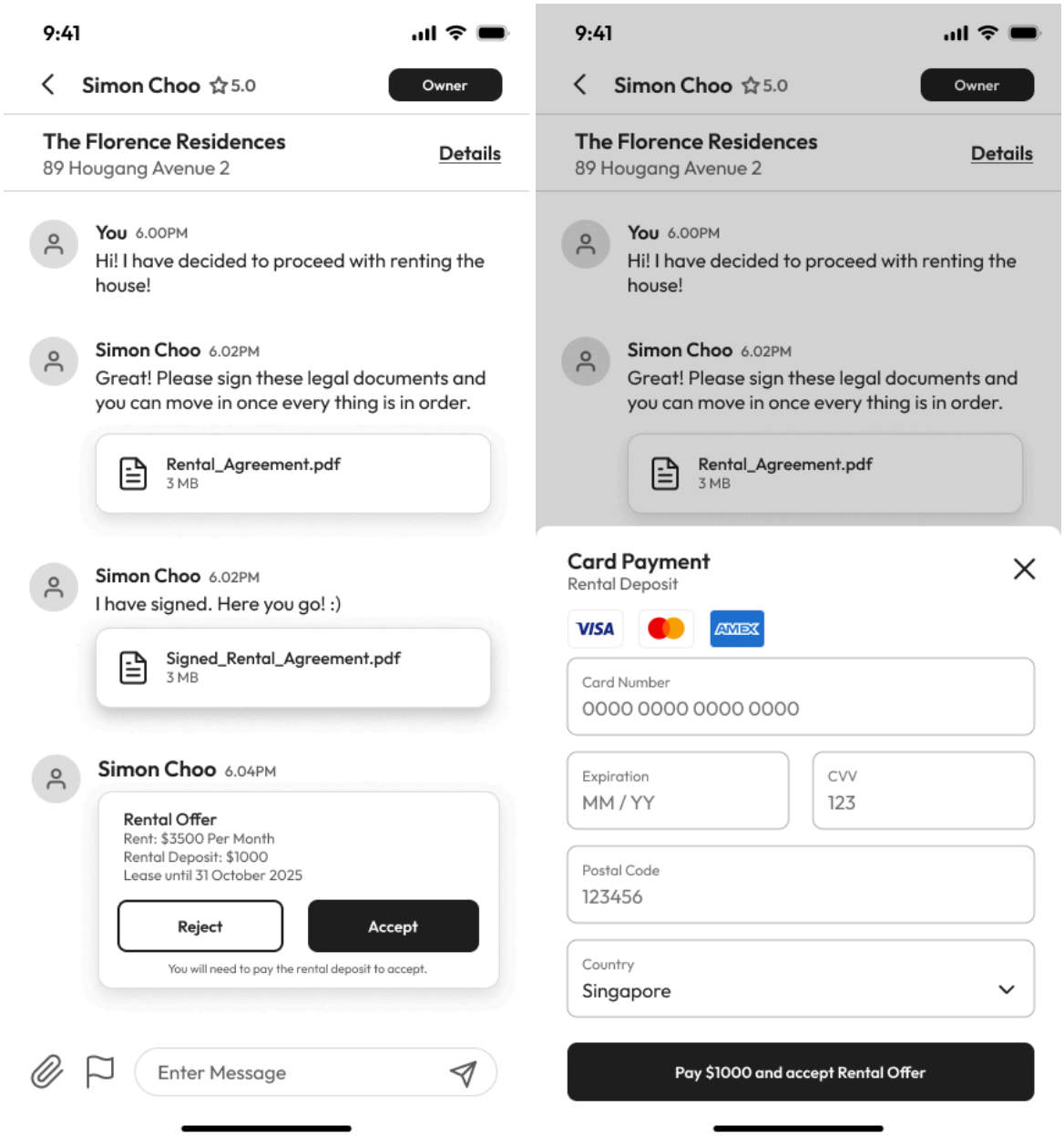
This function is only applicable and accessible by property owners only.

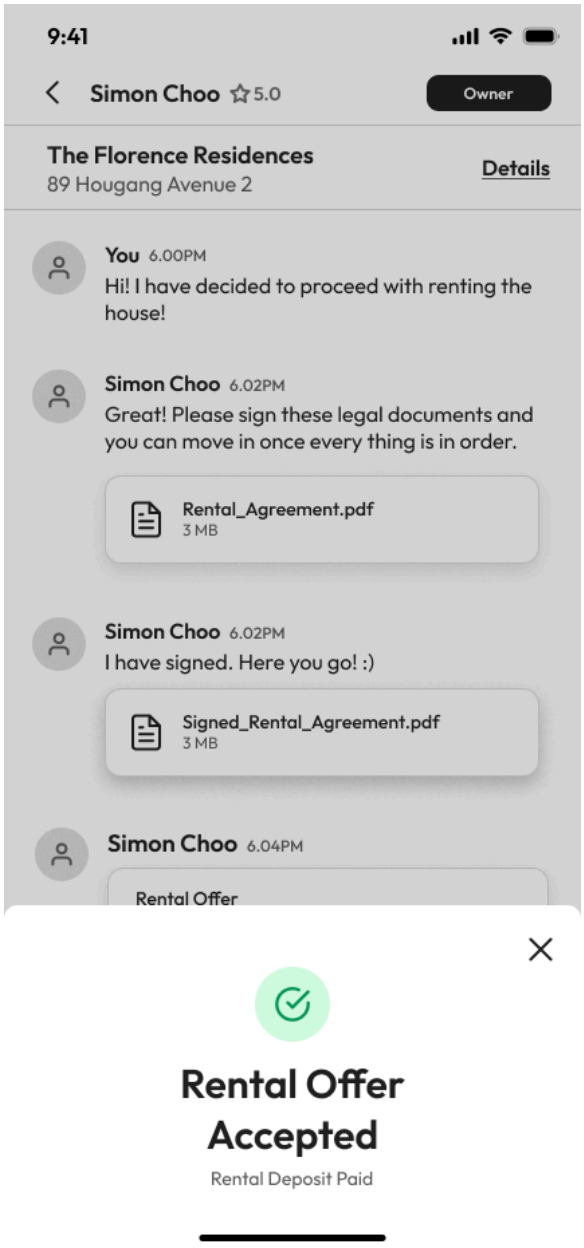



9:41
Simon Choo 5.0
Tenant
The Florence Residences
89 Hougang Avenue 2
Details
Simon Choo 6.00PM
Hi! I have decided to proceed with renting the house!
You 6.02PM
Great! Please sign these legal documents and you can move in once every thing is in order.
Rental_Agreement.pdf
Please e-sign and send it back.
You 6.04PM
Rental Offer
Rent: $3500 Per Month
Rental Deposit: $1000
Lease until 31 October 2025
Pending Acceptance
Enter Message

3.1.9.2 Review rental offer and pay deposit

This function is only applicable and accessible by tenants only.

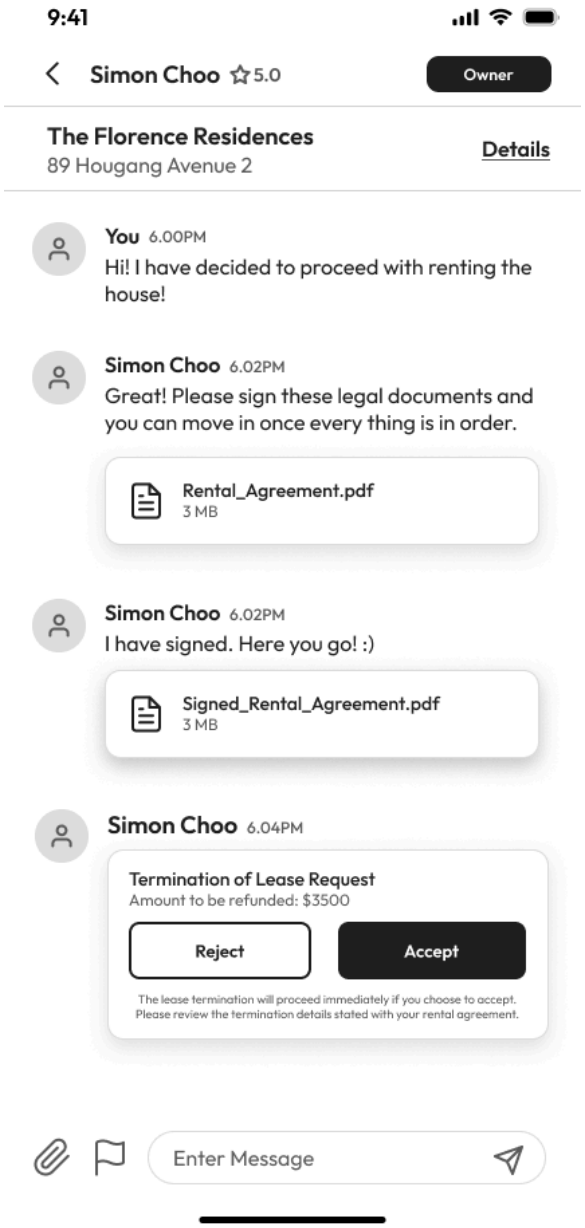



9:41
Simon Choo 5.0
Owner
The Florence Residences
89 Hougang Avenue 2
Details
You 6.00PM
Hi! I have decided to proceed with renting the house!
Simon Choo 6.02PM
Great! Please sign these legal documents and you can move in once every thing is in order.
Rental_Agreement.pdf
3 MB
Simon Choo 6.02PM
I have signed. Here you go! :)
Signed_Rental_Agreement.pdf
3 MB
Simon Choo 6.04PM
Rental Offer

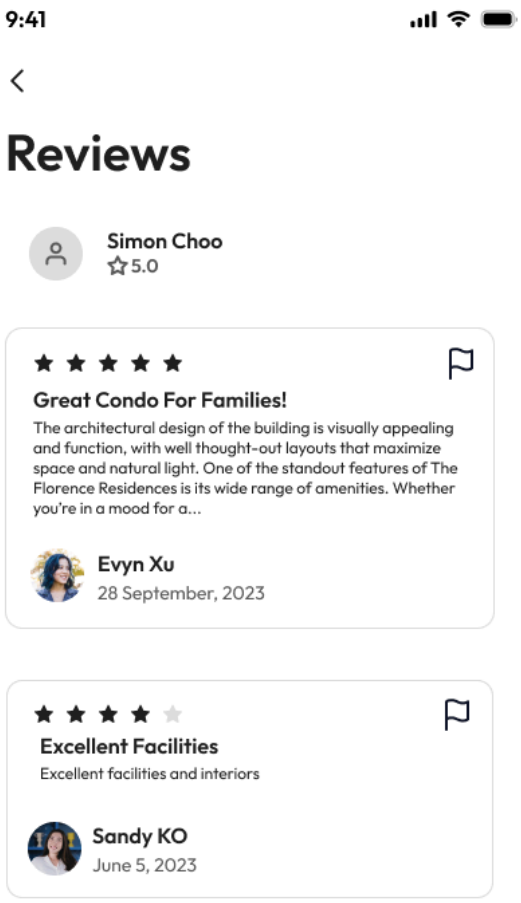
Rental Offer Accepted
Rental Deposit Paid

3.1.9.3 Review termination of lease

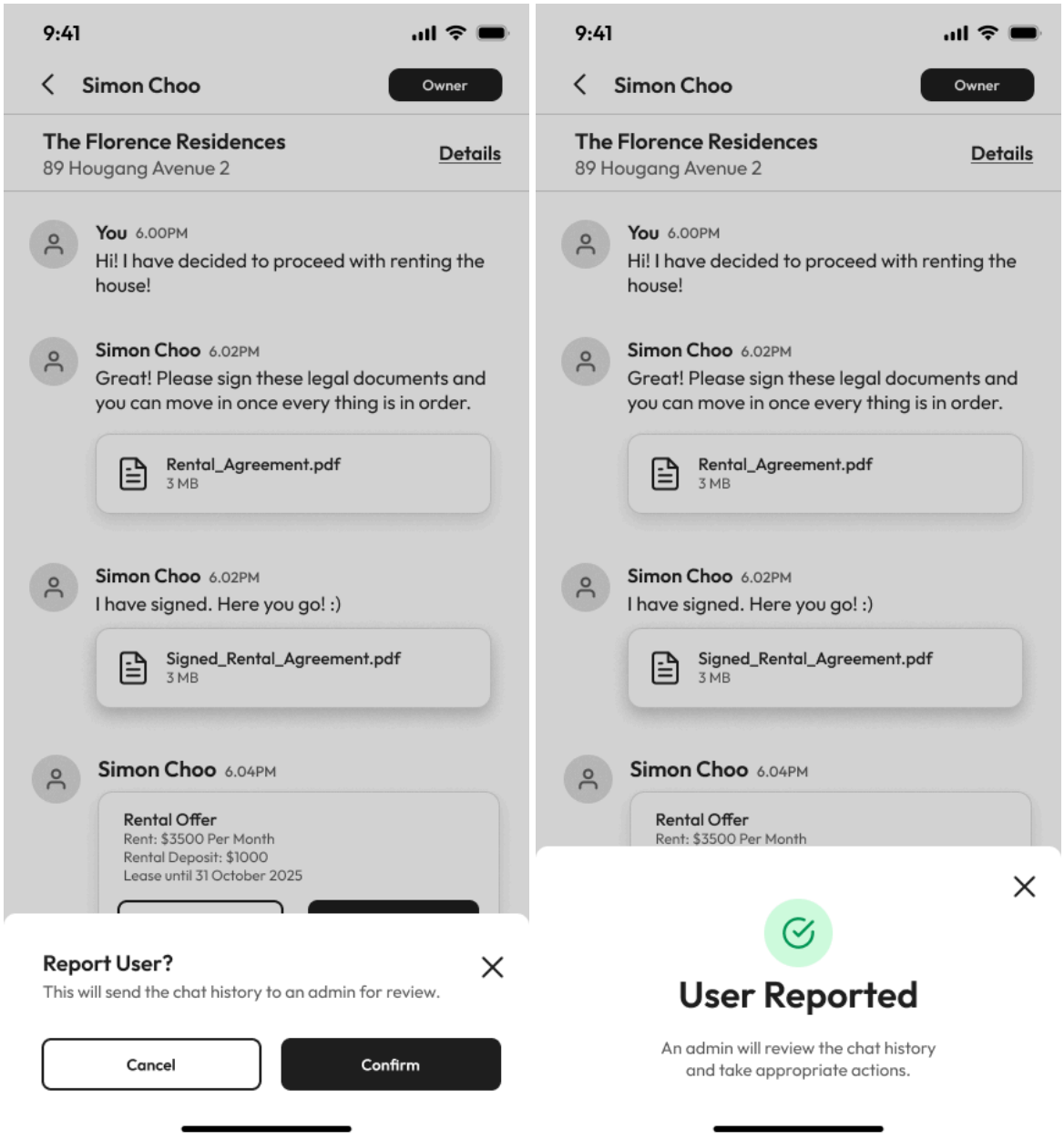
This function is only applicable and accessible by tenants only.



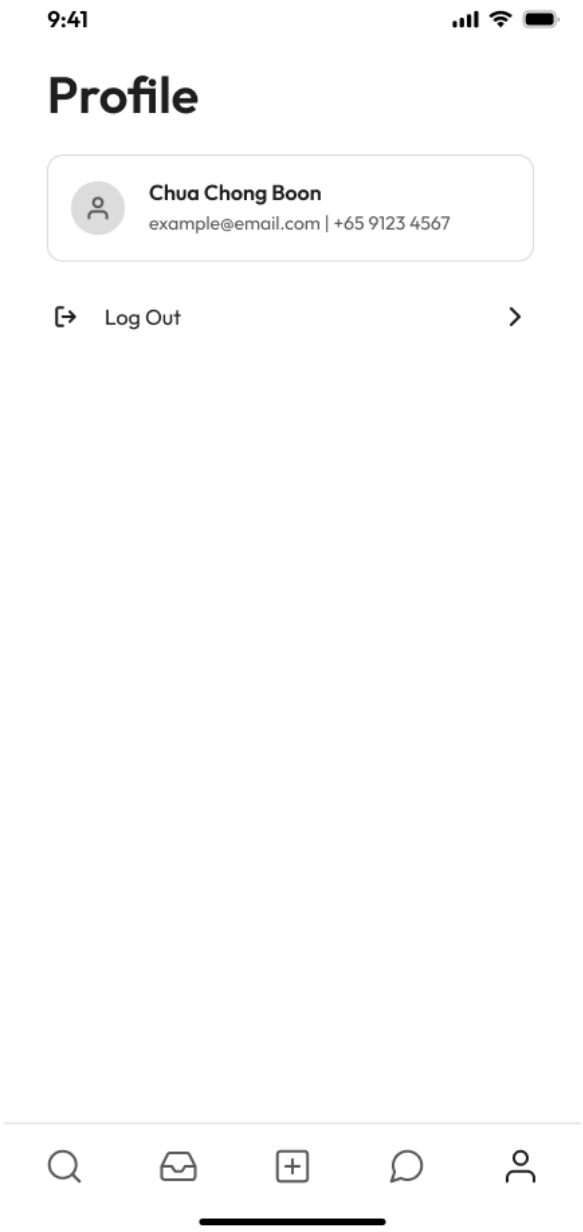
3.1.9.4 View Reviews



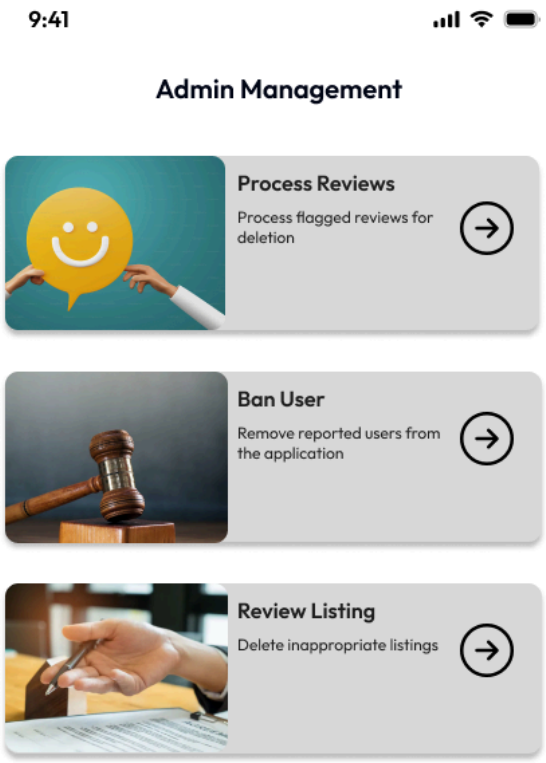
3.1.9.5 Report User



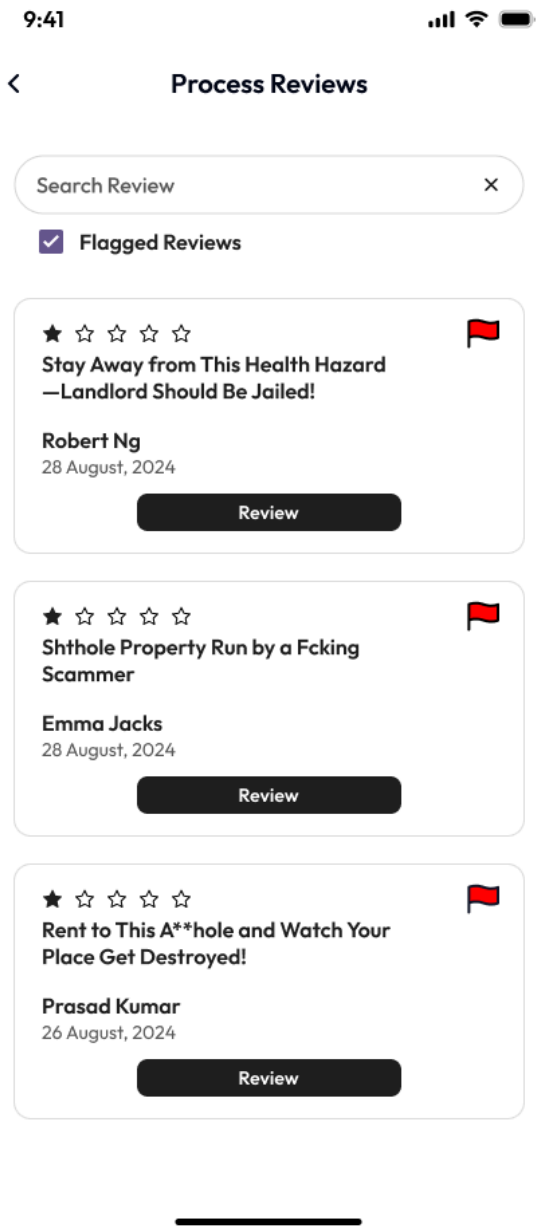
3.1.10 Profile

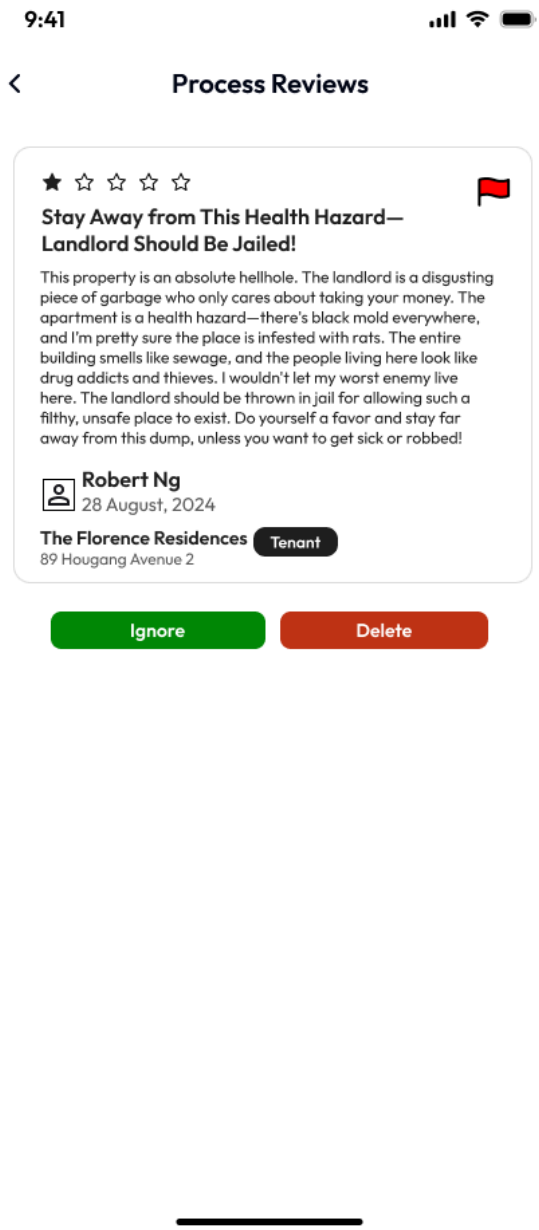


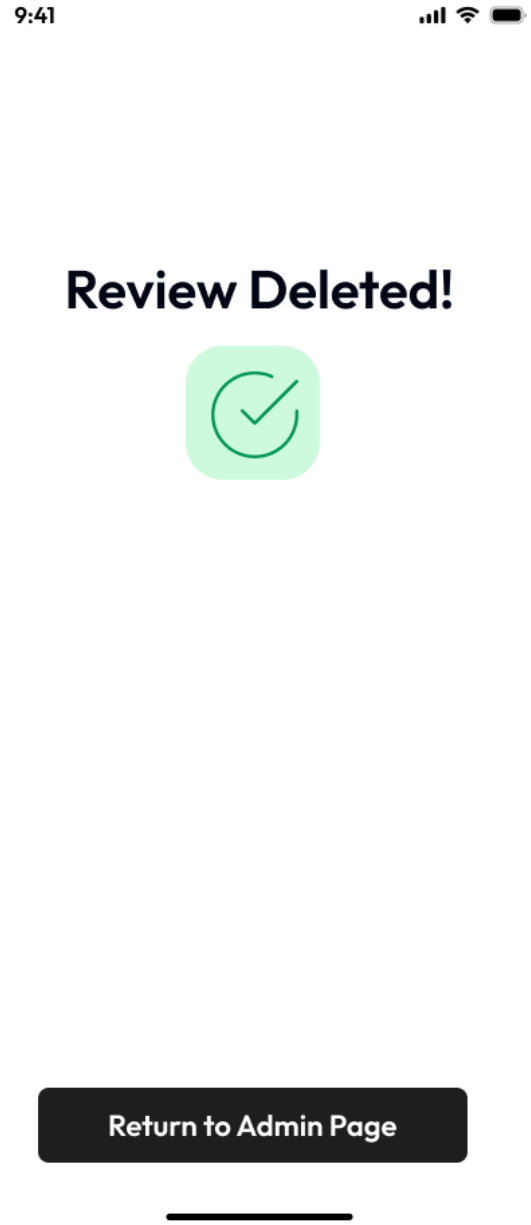
3.1.11 Admin Management Portal



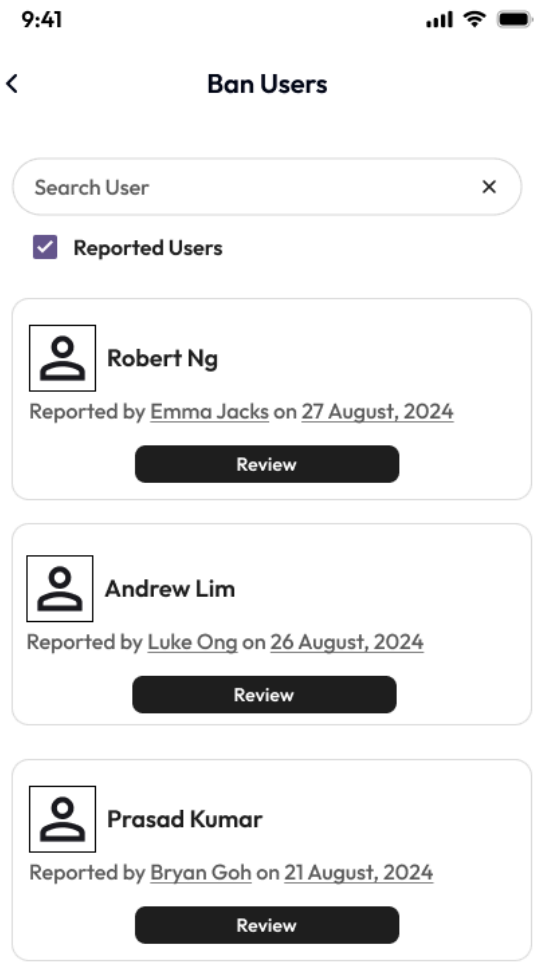
3.1.11.1 Process Reviews

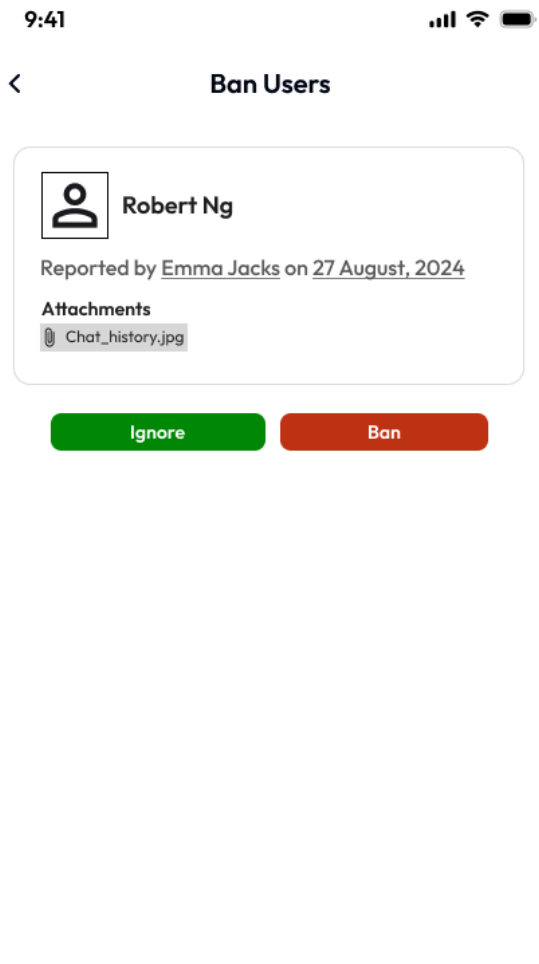


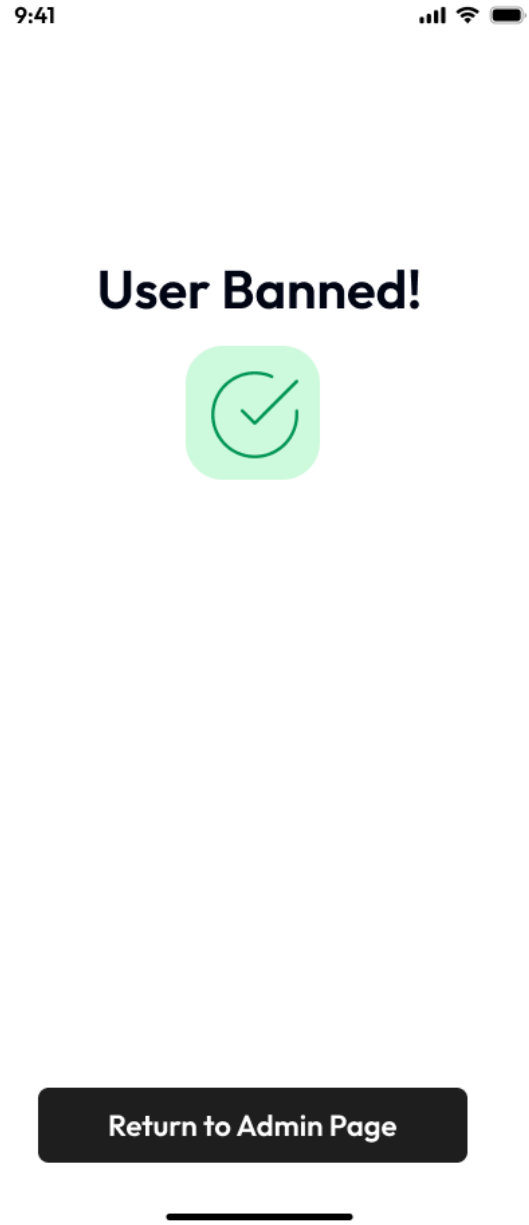



9:41
Review Deleted!
Return to Admin Page

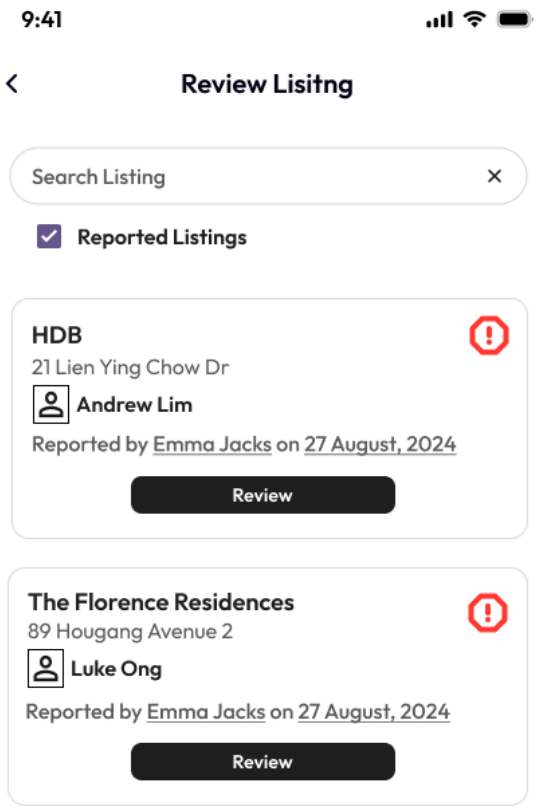
3.1.11.2 Ban Users

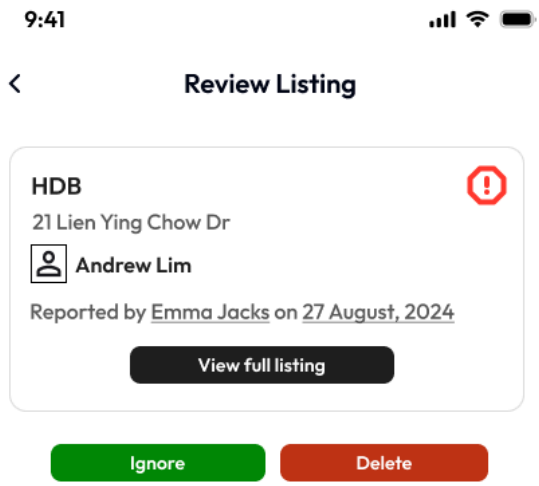


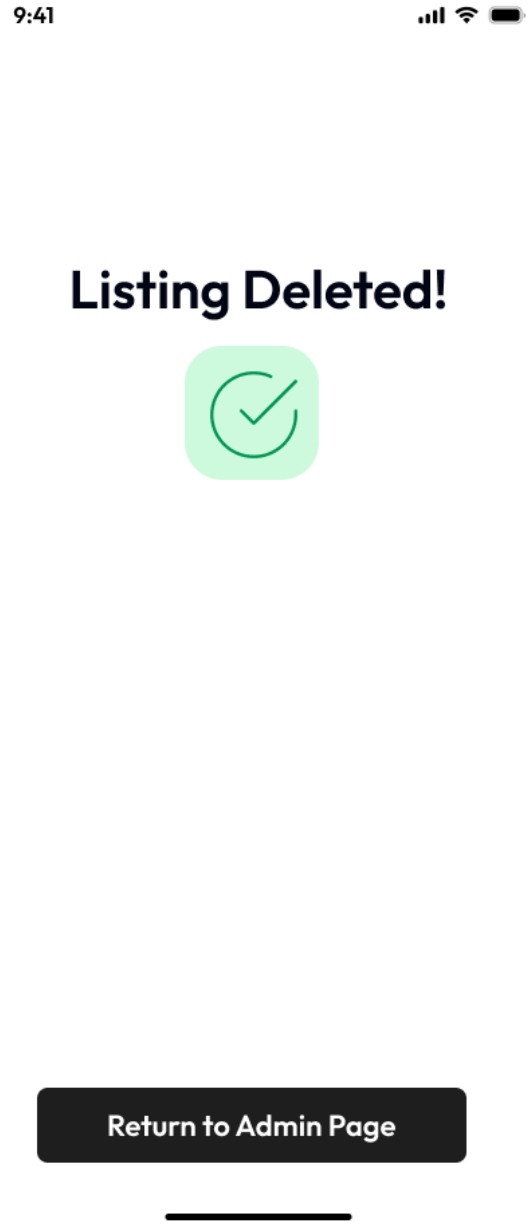



9:41
User Banned!
Return to Admin Page

3.1.11.3 Review Listings






9:41
Listing Deleted!
Return to Admin Page

3.2 Hardware Interfaces

The HomeGoWhere application is designed to be compatible with most mobile devices.

3.3 Software Interfaces

The HomeGoWhere application is designed to be compatible with iOS and Android mobile operating systems.

The HomeGoWhere application uses a relational database to manage and store persistent data. Below are the primary tables and their schemas:

- Users
    - **userid** (long), name (String), email (String), contact (String), photourl (String), flagged (int), password (String, encrypted)
- Listings
    - **listingid** (long), owneruserid (= userid), tenantuserid (= userid), name (String), type (String), floor (int) , unitnumber (String), location (String), postal (int), price (int), size (int), beds (int), bathroom (int), description (String), photourl (String), flagged (boolean)
- Rentals
    - **rentalid** (long), listingid (= listingid), tenantuserid (= userid), rentalprice (long), depositprice (long), rentaldate (Date), leaseexpiry (Date), paymenthistory (String), status (String)
- Payment
    - **paymentid** (long), rentalid (= rentalid), amount (Long), date (Date)
- Request
    - **requestid** (long), rentalid (= rentalid), status (String), refundamount (long)
- Reviews
    - **reviewid** (long), userid (= userid), reviewerid (= userid), rating (int), title (String), text (String), flagged (boolean)
- ChatHistory
    - **messageid** (long), receiverid (= userid), senderid (= userid), message (String), date (Date), rentalid (= rentalid), requestid (=requestid)

4. System Features

4.1 Functional Requirements

1. HomeGoWhere shall enable admins to perform administrative tasks.

    1.1. HomeGoWhere shall allow Admins to view all reported Users, Listings and Reviews, and take appropriate follow up actions.

    1.2. If there are inappropriate rental listings, HomeGoWhere shall allow Admins to remove the listings from the platform.

        1.2.1. If any listing is reported by a user as inappropriate or misleading, HomeGoWhere shall allow admins to remove them from the platform after reviewing them.

    1.3. If there is usage misconduct or violation of terms, HomeGoWhere shall allow admins to ban these users from the platform.

        1.3.1. If any user is reported by another user for inappropriate messages within the platform's chat function, HomeGoWhere shall allow admins to ban these users from the platform after reviewing them.

        1.3.2. If any user repeatedly posts inappropriate reviews or listings, HomeGoWhere shall allow admins to ban these users from the platform.

    1.4. If there are unfair or inappropriate reviews that are flagged by users, HomeGoWhere shall allow Admins to remove the reviews from the platform.

        1.4.1. If the review for Owners contains information not relevant to the quality of the rental experience (such as property condition, communication, or Tenant satisfaction), HomeGoWhere shall allow Admins to remove the review from the platform.

        1.4.2. If the review for Tenants contains information not relevant to the quality of their tenancy (such as cleanliness, payment timeliness, or respect for

property), HomeGoWhere shall allow Admins to remove the review from the platform.

1.4.3. If a review contains any other sensitive content that is not relevant, HomeGoWhere shall allow Admins to remove the review from the platform.

2. HomeGoWhere shall allow Owners to perform Owner-specific tasks.

2.1. HomeGoWhere shall allow Owners to make a new rental listing.

2.1.1. Owners can enter the address of their rental property.
2.1.2. Owners can select the type of property they are renting.
2.1.3. Owners can state the listing price of their rental property.
2.1.4. Owners can state the number of bedrooms and bathrooms in their rental property.
2.1.5. Owners can state the floor size of their rental property.
2.1.6. Owners can provide a title and description for their rental property.
2.1.7. Owners can upload pictures of their listing.

2.2. HomeGoWhere shall allow Owners to edit their rental listing.

2.2.1. HomeGoWhere shall display a “Edit Rental Listing” button in the rental info
2.2.2. When the “Edit Rental Listing” button is selected, HomeGoWhere shall allow Owners to upload new pictures and remove uploaded pictures.
2.2.3. When the “Edit Rental Listing” button is selected, HomeGoWhere shall allow Owners to modify the listing text.
2.2.4. HomeGoWhere shall display a “Update Listing” button to allow Owners to save the changes they made permanently.

2.3. HomeGoWhere shall allow Owners to manage Tenants.

2.3.1. HomeGoWhere shall display the Tenants living under the Owner’s listings.
2.3.2. HomeGoWhere shall display the Tenant’s payment history when the Owner clicks onto the individual Tenant’s name.

2.4. HomeGoWhere shall allow Owners to communicate with Tenants through in-app messaging

2.4.1. HomeGoWhere shall allow Owners to send a text message to the Tenant.
2.4.2. HomeGoWhere shall allow Owners to report the Tenant.
2.4.3. HomeGoWhere shall display a “Flag” button beside the chat box to allow Owners to flag an Tenant.
2.4.4. If the “Flag” button is selected, HomeGoWhere shall add the Tenant to the list of flagged Users for the Admin to process.

2.5. HomeGoWhere shall allow Owners to make termination requests.

2.5.1. HomeGoWhere shall display a terminate lease page when Owner selects the “Terminate Lease” button shown in Tenant overview page.
2.5.2. HomeGoWhere shall display “Collect Payment”, “Refund Tenant” and “Submit Termination” buttons for Owner to select.
2.5.3. If the Owner selects the “Collect Payment” button, the Owner shall enter the amount that they intend to collect from Tenant.
2.5.4. If the Owner selects the “Refund Tenant” button, the Owner shall enter the amount that they intend to refund to Tenant.
2.5.5. When the Owner selects the “Submit Termination” button, HomeGoWhere shall submit the termination of lease request into the chat log with the respective Tenant.
2.5.6. HomeGoWhere shall display the current status of the submitted termination lease.

2.6. HomeGoWhere shall allow Owners to make rental offers.

2.6.1. HomeGoWhere shall display a make rental offer screen page when Owner selects the “Make Rental Offer” button shown in the chat interface with a tenant.
2.6.2. HomeGoWhere shall display input fields for the Owner to specify the monthly rental amount, required deposit amount, and end of lease date.

2.6.3. When the Owner selects the “Confirm Offer” button, HomeGoWhere shall submit the rental offer into the chat log with the respective Tenant.

2.6.4. HomeGoWhere shall display the current status of the submitted rental offer.

3. HomeGoWhere shall allow Tenants to perform Tenant-specific tasks.

3.1. HomeGoWhere shall allow Tenants to communicate with Owner through in-app messaging

3.1.1. HomeGoWhere shall allow Tenants to send a text message to the Owner.

3.1.2. HomeGoWhere shall allow Tenants to review rental offers sent by the Owner.

3.1.3. HomeGoWhere shall display both “accept” and “reject” buttons in the rental offer offered by the Owner.

3.1.4. When Tenant selects the “accept” button, HomeGoWhere will display a confirmation page that indicates a successful acceptance of the rental offer.

3.1.5. HomeGoWhere shall allow Tenants to make a deposit.

3.1.6. HomeGoWhere shall display a payment page after the Tenant has selected the “accept” button in the rental offer.

3.1.7. HomeGoWhere shall display a confirmation page after the Tenant has successfully made payment.

3.1.8. HomeGoWhere shall allow Tenants to report the Owner

3.1.9. HomeGoWhere shall display a “Flag” button beside the chat box to allow Tenants to flag an Owner.

3.1.10. If the “Flag” button is selected, HomeGoWhere shall add the Owner to the list of flagged Users for the Admin to process.

3.1.11. HomeGoWhere shall allow Tenants to review termination request sent by Owner

3.1.12. HomeGoWhere shall display an “accept” and “reject” button for the Tenant to select.

3.2. HomeGoWhere shall display a rent payment page for Tenants to make payment.

3.2.1. HomeGoWhere shall display Tenant's outstanding payments.

3.2.2. If the "Pay" button is selected, HomeGoWhere shall display a payment page for the Tenant to make payment.

3.2.3. HomeGoWhere shall display the history of Tenant's payment history in the rent payment page via a list interface.

4. HomeGoWhere shall provide users with general functionality to interact with the application.

4.1. HomeGoWhere shall allow Users to be authenticated and use the application.

4.1.1. HomeGoWhere shall allow Users to create an account

4.1.1.1. HomeGoWhere shall allow users to input their Full Name, E-Mail, password and phone no.

4.1.1.2. HomeGoWhere shall verify the user through a One-Time Password message sent to the user's phone number.

4.1.1.3. HomeGoWhere shall create an account with the information entered by the user.

4.1.2. HomeGoWhere shall allow Users to sign in using the account they have created previously.

4.1.2.1. HomeGoWhere shall allow Users to enter their E-Mail and password to log into the application.

4.1.2.2. HomeGoWhere shall mask the password entered by the Users by replacing actual text with dots unless the Users choose to unmask it.

4.1.2.3. If the E-Mail and password do not match, HomeGoWhere shall display "E-mail and password do not match" to the user.

4.1.2.4. If the E-mail and password match, HomeGoWhere shall log the User in and navigate the User to the home screen of the application.

4.1.3. HomeGoWhere shall allow Users to edit their profile details.

4.1.3.1. HomeGoWhere shall allow Users to edit their name.
4.1.3.2. HomeGoWhere shall allow Users to edit their mobile number.
4.1.3.3. HomeGoWhere shall allow Users to edit their email.
4.1.3.4. HomeGoWhere shall allow Users to edit their profile photo.

4.2. HomeGoWhere shall allow Users to view and query all listings.

4.2.1. HomeGoWhere shall display all listings via a list interface to provide a comprehensive view of available options.
4.2.2. HomeGoWhere shall provide a search option that allows users to search for listings based on location.
4.2.3. HomeGoWhere shall allow users to filter listings based on the property type and rental type.
4.2.4. When a listing is clicked, HomeGoWhere shall show detailed information of the selected listing.

4.3. HomeGoWhere shall allow Users to view detailed information of all listings

4.3.1. HomeGoWhere shall allow Users to view details about the location of the listing.

4.3.1.1. HomeGoWhere shall display a real-time map that displays the locations of bus stops and MRT stations nearby the listing.
4.3.1.2. HomeGoWhere shall display a real-time map that displays the locations of schools nearby the listing.
4.3.1.3. HomeGoWhere shall display a real-time map that displays the locations of hawkers nearby the listing.
4.3.1.4. HomeGoWhere shall allow the real-time map to be zoomed in, zoomed out, and panned.

4.3.2. HomeGoWhere shall allow Users to view the price insights of the listing.

4.3.2.1. HomeGoWhere shall display a list of past rental contracts from the area that were transacted in the past year.

4.3.2.2. HomeGoWhere shall display the street of the rental property from the list of past rental contracts.

4.3.2.3. HomeGoWhere shall display the floor area of the rental property from the list of past rental contracts.

4.3.2.4. HomeGoWhere shall display the monthly rent price of the rental property from the list of past rental contracts.

4.3.3. HomeGoWhere shall allow Users to view the reviews of the listing.

4.3.3.1. HomeGoWhere shall display the reviews left by past tenants of the property.

4.3.3.2. HomeGoWhere shall display the full review when Users click onto the review.

4.3.4. HomeGoWhere shall allow Users to flag reviews.

4.3.4.1. HomeGoWhere shall display a “Flag” button on every review to allow Users to flag a review.

4.3.4.2. If the “Flag” button is selected, HomeGoWhere shall add that review to a list of flagged reviews for the Admin to process.

4.3.5. HomeGoWhere shall allow Users to start a chat with Owners.

4.3.5.1. HomeGoWhere shall display a “Message Owner” button on every listing to allow Users to start a chat with the Owner of the property.

4.3.6. HomeGoWhere shall allow Users to report listing

4.3.6.1. HomeGoWhere shall display a “Report Listing” button on every listing to allow Users to flag a listing.

4.3.6.2. If the “Flag” button is selected, HomeGoWhere shall add that review to a list of flagged reviews for the Admin to process.

4.4. HomeGoWhere shall allow Users to submit reviews for other Users.

4.4.1. HomeGoWhere shall allow Owners to submit reviews on Tenants.

4.4.1.1. When the “Leave Review” button in the Tenant overview is selected, the Owner can leave a review text of the Tenant.

4.4.2. HomeGoWhere shall allow Owners to edit their review on Tenants.

4.4.2.1. HomeGoWhere shall allow Owners to modify their review text and star rating in the leave review page.

4.4.2.2. If the “Submit/Update Review” Button is selected, HomeGoWhere will save the changes they made.

4.4.3. HomeGoWhere shall allow Owners to view reviews on Tenants.

4.4.3.1. HomeGoWhere shall display the Tenant’s star rating in the chat log.

4.4.3.2. When the “Star” button in the chat log is selected, HomeGoWhere shall display a list of the text reviews of the Tenant.

4.4.4. HomeGoWhere shall allow Tenants to submit their review on Owners

4.4.4.1. HomeGoWhere shall allow Tenants to rate the specific Owner and their listing by clicking on the star buttons.

4.4.4.2. HomeGoWhere shall allow Tenants to enter review titles and text about a specific Owner and their listing.

4.4.4.3. HomeGoWhere shall display a “Submit/Update Review” Button for Tenants to confirm their review submissions before saving and submitting the reviews.

4.4.5. HomeGoWhere shall allow Tenants to edit their review on Owners

4.4.5.1. HomeGoWhere shall allow Tenants to modify their review text and star rating in the leave review page.

4.4.5.2. If the “Submit/Update Review” Button is selected, HomeGoWhere will save the changes they made.

4.4.6. HomeGoWhere shall allow Tenants to delete their review on Owners.

4.4.6.1. HomeGoWhere shall display a “Delete Review” Button in the leave review page for Tenants to delete his/her review.

4.4.6.2. If the “Delete Review” Button is selected, HomeGoWhere shall display the “Confirm Deletion” button.

4.4.6.3. If the “Confirm Deletion” button is selected, the review would be removed from the listing page.

4.4.7. HomeGoWhere shall allow Tenants to view reviews on Owners.

4.4.7.1. HomeGoWhere shall display the Owner’s star rating in the chat log.

4.4.7.2. When the “Star” button in the chat log is selected, HomeGoWhere shall display a list of the text reviews of the Owner.

4.2 Use Case Descriptions

4.2.1 Functional Requirement #1

4.2.1.1 SelectAdminActivity

Use Case ID:	#1-1		
Use Case Name:	SelectAdminActivity		
Created By:	Lau Zhan You	Last Updated By:	Lau Zhan You
Date Created:	28th August 2024	Date Last Updated:	28th August 2024

Actor:	Admin
Description:	Allows the admin to review listings posted by Owners, ban users for inappropriate behaviour and facilitates the management of reviews posted by Owners and Tenants.
Preconditions:	Admin is logged in and is authenticated.
Postconditions:	None
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system uses the included use case Login to verify the Owner. 2. Once logged in, the system prompts the Admin to select the desired activity: PROCESS_REVIEW, BAN_USER or REVIEW_LISTING. 3. If the admin selects the activity REVIEW_LISTING, then the admin uses the included use case **ReviewListing** to approve or reject a user's application to post a listing. 4. If the admin selects the activity PROCESS_REVIEWS, then the Admin uses the included use case **ProcessReviews** to decide whether a review is to be removed. 5. If the admin selects the activity BAN_USER, then the Admin uses the included use case **BanUser** to decide whether a user should be banned. 6. If the Admin selects the activity QUIT, the system returns to

	the login screen.
Alternative Flows:	None
Exceptions:	None
Includes:	1. ReviewListing 2. BanUser 3. ProcessReview
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.1.2 ReviewListing

Use Case ID:	#1-2		
Use Case Name:	ReviewListing		
Created By:	Lau Zhan You	Last Updated By:	Lau Zhan You
Date Created:	28th August 2024	Date Last Updated:	28th August 2024

Actor:	Admin
Description:	Allows the admin to review listings posted by Owners for deletion or keeping.
Preconditions:	Admin is logged in and is authenticated.
Postconditions:	Flagged review is deleted or ignored.
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. The HomeGoWhere lists the listings that have been flagged by it or Users. 2. Admin selects the listing that the Admin wants to process. 3. The system displays the content of the listing and prompts the Admin to select the “DELETE” or “IGNORE” button. 4. If the Admin selects the “DELETE” button, the system will delete the flagged listing.
Alternative Flows:	AF-S3: If the Admin selects IGNORE 1. If the Admin selects the “IGNORE” button, the system will ignore the flagged listing and remove it from the list of flagged listings. 2. The system will then return to the list of remaining flagged listings.
Exceptions:	None
Includes:	None
Special Requirements:	None

Assumptions:	None
Notes and Issues:	None

4.2.1.3 BanUser

Use Case ID:	#1-3		
Use Case Name:	BanUser		
Created By:	Lau Zhan You	Last Updated By:	Lau Zhan You
Date Created:	28th August 2024	Date Last Updated:	28th August 2024

Actor:	Admin
Description:	Allows the admin to ban a User by their user ID.
Preconditions:	Admin is logged in and is authenticated.
Postconditions:	Users are either banned or not banned by the admin.
Priority:	High
Frequency of Use:	Low
Flow of Events:	1. Admin reviews the lists of flagged users that have been flagged by other users. 2. Admin selects the flagged users case that the Admin wants to process. 3. The system displays the content of the case and prompts the Admin to select the “IGNORE" or “BAN” button. 4. If the Admin selects the “BAN” button, the system will ban the user.
Alternative Flows:	AF-S3: If the Admin selects IGNORE 1. If the Admin selects the “IGNORE” button, the system will ignore the flagged users case and remove it from the list of flagged listings. 2. The system will then return to the list of remaining flagged users.
Exceptions:	None
Includes:	None
Special Requirements:	None

Assumptions:	None
Notes and Issues:	None

4.2.1.4 ProcessReviews

Use Case ID:	#1-4		
Use Case Name:	ProcessReviews		
Created By:	Lau Zhan You	Last Updated By:	Lau Zhan You
Date Created:	28th August 2024	Date Last Updated:	28th August 2024

Actor:	Admin
Description:	Allows the admin to process a review for deletion or for keeping.
Preconditions:	Admin is logged in and is authenticated.
Postconditions:	Flagged review is deleted or ignored.
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. The HomeGoWhere lists the reviews that have been flagged by it or Users. 2. Admin selects the review that the Admin wants to process. 3. The system displays the content of the review and prompts the Admin to select the “IGNORE" or “DELETE” button. 4. If the Admin selects the “DELETE” button, the system will delete the flagged review.
Alternative Flows:	AF-S3: If the Admin selects IGNORE 1. If the Admin selects the “IGNORE” button, the system will ignore the flagged review and remove it from the list of flagged reviews. 2. The system will then return to the list of remaining flagged reviews.
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	None

Notes and Issues:	None

4.2.2 Functional Requirement #2

4.2.2.1 ManageListings

Use Case ID:	#2-1		
Use Case Name:	ManageListings		
Created By:	Lim Jing Rong	Last Updated By:	Lim Jing Rong
Date Created:	30th August 2024	Date Last Updated:	30th August 2024

Actor:	Owner
Description:	Allows Owner to manage the listing for the properties they are renting out or selling.
Preconditions:	Owner is logged in and is authenticated.
Postconditions:	None
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The Owner will be provided with 2 use cases a. SubmitListing b. EditListing 2. If the Owner selects the activity SUBMIT_LISTING, then the Owner uses the included use case SubmitListing to post a listing. 3. If the Owner selects the activity EDIT_LISTING, then the Owner uses the included use case EditListing to edit a listing.
Alternative Flows:	None
Exceptions:	None
Includes:	1. SubmitListing 2. EditListing

Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.2.2 SubmitListing

Use Case ID:	#2-2		
Use Case Name:	SubmitListing		
Created By:	Loo Ping Wee	Last Updated By:	Lim Jing Rong
Date Created:	26th August 2024	Date Last Updated:	30th August 2024

Actor:	Owner
Description:	Allows Owner to create a listing for the property they are renting out or selling, and it gets sent to the database for Users to view.
Preconditions:	1. Owner is logged in and is authenticated 2. All required fields in the form are filled
Postconditions:	None
Priority:	High
Frequency of Use:	Medium
Flow of Events:	1. Owner fills all the required fields in the form presented to them. 2. Owner clicks the “Submit” button, and SubmitListing is called. 3. Checks that all required fields are filled. 4. Formdata is sent to the database and saved.
Alternative Flows:	<u>AF-S3: Owner did not fill in the required fields of the form</u> 1. Owner has not filled in all the required fields of the form. 2. Owner clicks the “Submit” button, and SubmitListing is called. 3. Checks that all required fields are filled are not passed. 4. Formdata is not sent to the database, Owner is prompted to field in all required fields, no changes to the form.
Exceptions:	None
Includes:	None
Special Requirements:	None

Assumptions:	None
Notes and Issues:	None

4.2.2.3 EditListing

Use Case ID:	#2-3		
Use Case Name:	EditListing		
Created By:	Lim Jing Rong	Last Updated By:	Lim Jing Rong
Date Created:	30th August 2024	Date Last Updated:	30th August 2024

Actor:	Owner
Description:	Allows Owner to edit a listing for the property they are renting out or selling, and it updates the database for Users to view.
Preconditions:	1. Owner is logged in and is authenticated. 2. All required fields in the form are filled.
Postconditions:	None
Priority:	High
Frequency of Use:	Medium
Flow of Events:	1. Owner clicks on his listing, he will be allowed to edit the listing 2. Owner clicks the “Submit” button, and SubmitListing is called. 3. Checks that all required field are filled 4. Formdata is sent to the database and saved.
Alternative Flows:	AF-S3: Owner did not fill in the required fields of the form 1. Owner has not filled in all the required fields of the form. 2. Owner clicks the “Submit” button, SubmitPropertylisting is called. 3. Checks that all required fields are filled and are not passed. 4. Formdata is not sent to the database, Owner is prompted to field in all required fields, no changes to the form.
Exceptions:	None
Includes:	None
Special Requirements:	None

Assumptions:	None
Notes and Issues:	None

4.2.2.4 ManageTenants

Use Case ID:	#2-4		
Use Case Name:	ManageTenants		
Created By:	Loo Ping Wee	Last Updated By:	Lim Jing Rong
Date Created:	26th August 2024	Date Last Updated:	31st August 2024

Actor:	Owner
Description:	Allows Owner view the information of all Tenants that have rented from the Owner.
Preconditions:	Owner is logged in and is authenticated.
Postconditions:	None
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system uses the included use case Login to verify the Owner. 2. Once logged in, the system prompts the Owner to select ManageTenants. 3. The Owner selects ManageTenants. 4. The Owner can choose to manage the Tenants that are currently renting from the Owner. 5. If the Owner selects the activity SELECT_TENANT, then the Owner uses the included use case **SelectTenant**.
Alternative Flows:	None
Exceptions:	None
Includes:	1. SelectTenant
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.2.5 SelectTenant

Use Case ID:	#2-5		
Use Case Name:	SelectTenant		
Created By:	Loo Ping Wee	Last Updated By:	Lim Jing Rong
Date Created:	26th August 2024	Date Last Updated:	31st August 2024

Actor:	Owner
Description:	Allows Owner to select a Tenant they wish to interact with.
Preconditions:	Owner is logged in and is authenticated.
Postconditions:	None
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The Owner can choose to terminate the Tenants that are currently renting from the Owner. 2. If the Owner selects the activity MAKE_TERMINATION_REQUEST, then the Owner uses the included use case **MakeTerminationRequest**.
Alternative Flows:	None
Exceptions:	None
Includes:	1. MakeTerminationRequest
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.2.6 MakeTerminationRequest

Use Case ID:	#2-6		
Use Case Name:	MakeTerminationRequest		
Created By:	Lim Jing Rong	Last Updated By:	Lim Jing Rong
Date Created:	30th August 2024	Date Last Updated:	30th August 2024

Actor:	Owner
Description:	Allows Owner to select a Tenant they wish to terminate.
Preconditions:	Owner is logged in and is authenticated.
Postconditions:	None
Priority:	High
Frequency of Use:	Medium
Flow of Events:	1. The Owner will select the Tenant that he wishes to terminate. 2. When the Owner presses the “Terminate Lease” button, it will prompt a request to terminate the lease.
Alternative Flows:	None
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.2.7 MakeRentalOffer

Use Case ID:	#2-7		
Use Case Name:	MakeRentalOffer		
Created By:	Lim Jing Rong	Last Updated By:	Lim Jing Rong
Date Created:	30th August 2024	Date Last Updated:	30th August 2024

Actor:	Owner
Description:	Allows Owner to make a rental offer to Tenant.
Preconditions:	Owner is logged in and is authenticated.
Postconditions:	None
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The Owner can send a rental offer to the Tenant a. The rental offer will include the rent per month. b. The rental offer will include the rental deposit required. c. The rental offer will include the end of lease date. 2. Owner clicks the “Send Offer” button. 3. Checks that all required fields are filled are present. 4. Offer sent to Tenant to accept.
Alternative Flows:	AF-S3: Required fields not filled in 1. Owner has not filled in all the required fields of the form. 2. Owner clicks the “Send Offer” button. 3. Checks that all required fields are filled are not passed. 4. Formdata is not sent to the database, Owner is prompted to field in all required fields, no changes to the form.
Exceptions:	None
Includes:	None
Special Requirements:	None

Assumptions:	None
Notes and Issues:	None

4.2.3 Functional Requirement #3

4.2.3.1 ReviewRentalOffer

Use Case ID:	#3-1		
Use Case Name:	ReviewRentalOffer		
Created By:	Lim Jing Rong	Last Updated By:	Lim Jing Rong
Date Created:	24th August 2024	Date Last Updated:	30th August 2024

Actor:	Tenant
Description:	Allows Tenants to review rental offer
Preconditions:	Tenant is logged in and is authenticated
Postconditions:	None
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The Tenant will click on the “Accept” button to accept the rental offer. 2. This will lead them to a card payment for the Tenant to pay the rental deposit. 3. After paying, it will display to the Tenant “Rental Offer Accepted”.
Alternative Flows:	<u>AF-S2: When the Tenant puts an invalid card number</u> 1. The Tenant inputs an invalid card number. 2. System will display “Error, Invalid credit card details”. <u>AF-S2: When the Tenant did not finish filling up the credit card information</u> 1. The Tenant did not input at least 1 of the required fields for the credit card details. 2. System will display “Missing required field”.
Exceptions:	None
Includes:	None

Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.3.2 MakeMonthlyPayment

Use Case ID:	#3-2		
Use Case Name:	MakeMonthlyPayment		
Created By:	Lim Jing Rong	Last Updated By:	Lim Jing Rong
Date Created:	24th August 2024	Date Last Updated:	24th August 2024

Actor:	Tenant
Description:	Allows Tenant to make monthly payment to Owner
Preconditions:	Tenant is logged in and is authenticated
Postconditions:	The payment is successfully made
Priority:	High
Frequency of Use:	Medium
Flow of Events:	1. Tenant navigates to the payment screen. 2. Tenant input the amount to pay. 3. Tenant confirms the submission by pressing the “Confirm” button.
Alternative Flows:	None
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.3.3 ReviewTerminationRequest

Use Case ID:	#3-3		
Use Case Name:	ReviewTerminationRequest		
Created By:	Lim Jing Rong	Last Updated By:	Lim Jing Rong
Date Created:	24th August 2024	Date Last Updated:	24th August 2024

Actor:	Tenant
Description:	Allow the Tenant to review the termination request
Preconditions:	Tenant is logged in and is authenticated
Postconditions:	Lease will be terminated and Tenant to move out
Priority:	High
Frequency of Use:	Medium
Flow of Events:	1. The Tenant will be notified about the termination request. 2. The Tenant chooses to accept the termination request a. The lease will be terminated.
Alternative Flows:	AF-S2: The Tenant does not accept the termination request 1. The lease will not be terminated. 2. The Tenant and Owner will discuss this matter in real life to come to a common consensus.
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.4 Functional Requirement #4

4.2.4.1 CreateAccount

Use Case ID:	#4-1		
Use Case Name:	CreateAccount		
Created By:	Lau Zhan You	Last Updated By:	Lau Zhan You
Date Created:	28th August 2024	Date Last Updated:	28th August 2024

Actor:	User
Description:	Allows a person to create an Account in HomeGoWhere to become a user.
Preconditions:	None
Postconditions:	An Account with a specific role (Admin, Owner or Tenant) is created for the User.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system prompts the User to enter their name, address, password, email, contact number, profile picture, and role. 2. The User enters the required information and selects the "Create Account" button to confirm his/her inputs. 3. An Account with a specific role (Admin, Owner or Tenant) is created for the User.
Alternative Flows:	AF-S2: Returning to Login page 1. If the User selects the back navigation icon, the system will navigate the User to the login page.
Exceptions:	1. If any required information is missing or invalid, an error message is displayed.
Includes:	None
Special Requirements:	System needs to validate user input data.

Assumptions:	None
Notes and Issues:	None

4.2.4.2 Login

Use Case ID:	#4-2		
Use Case Name:	Login		
Created By:	Lau Zhan You	Last Updated By:	Lau Zhan You
Date Created:	28th August 2024	Date Last Updated:	28th August 2024

Actor:	User
Description:	AllowsUser to login to his/her HomeGoWhere Account using his/her email and password.
Preconditions:	None
Postconditions:	User is logged into the HomeGoWhere application and is navigated to the home screen of the application.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system allows the User to login with either (a) Email and password, or (b) SingPass. 2. The User chooses to login with email and password. 3. The User enters his/her email and password. The password is masked as dots, but the User can choose to unmask it by clicking on the eye icon. 4. The User selects the "Login" button.
Alternative Flows:	AF-S1: User chooses to login with SingPass 1. The User selects the “Login with Singpass” button. 2. The User logins into the system using Singpass login credentials using the included use case SingpassLogin.
Exceptions:	1. If any required information is missing or invalid, an error message is displayed. 2. If email and password do not match when the User tries to login in Step 4, HomeGoWhere shall display “Email and password do not match” to the user.
Includes:	SingpassLogin

Special Requirements:	System needs to validate user input data.
Assumptions:	The User has an existing HomeGoWhere Account.
Notes and Issues:	None

4.2.4.3 SearchAndViewListing

Use Case ID:	#4-3		
Use Case Name:	SearchAndViewListing		
Created By:	Lau Zhan You	Last Updated By:	Lau Zhan You
Date Created:	28th August 2024	Date Last Updated:	28th August 2024

Actor:	User
Description:	Allows users to search for homes by name, type or location via a search bar and get a filtered list of houses.
Preconditions:	The User must be logged in and authenticated.
Postconditions:	A filtered list of homes is returned.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The User can search for homes according to their block, type or location via a search bar. 2. The system will return a filtered list of homes for the User.
Alternative Flows:	None
Exceptions:	1. If no home data matches the search criteria, the system shall display a message stating no property is found.
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.4.4 FlagListing

Use Case ID:	#4-4		
Use Case Name:	FlagListing		
Created By:	Lau Zhan You	Last Updated By:	Lau Zhan You
Date Created:	28th August 2024	Date Last Updated:	28th August 2024

Actor:	User
Description:	Allows User to flag a listing for the Admin to process for deletion.
Preconditions:	Entry Condition: Called by the use case **SearchAndViewListing** when the "Flag Listing" function is selected. The User must be logged in and authenticated.
Postconditions:	A listing is flagged.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system displays an option to flag a listing for every listing that the User sees. 2. If the User flags the listing, the system will prompt the user to input the reason as to why they have flagged the listing. 3. If the user selects CONFIRM, the system will flag the listing and add it to the list of flagged listings for the Admins to process.
Alternative Flows:	AF-S3: If the user selects CANCEL 1. If the user selects CANCEL, the system will cancel the operation and return the User back to the listings.
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	None

Notes and Issues:	None

4.2.4.5 Chat

Use Case ID:	#4-5		
Use Case Name:	Chat		
Created By:	Lau Zhan You	Last Updated By:	Lau Zhan You
Date Created:	28th August 2024	Date Last Updated:	1st September 2024

Actor:	User
Description:	Allows Users to communicate with Owners via text in the in-built Chat function.
Preconditions:	The User must be logged in and authenticated.
Postconditions:	A chat is initiated with the Owner.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User navigates to the desired listing screen. 2. User selects the Chat icon to initiate a conversation with the Owner via text.
Alternative Flows:	AF-S2: If the user selects the BACK navigation icon 1. If the user selects BACK, the system will cancel the operation and return the User back to the listings. AF-S2: If the user selects the STAR ratings icon 1. If the user selects the STAR ratings, the system will navigate the user to a page with a list of reviews for that person.
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.4.6 FlagUser

Use Case ID:	#4-6		
Use Case Name:	FlagUser		
Created By:	Lau Zhan You	Last Updated By:	Lau Zhan You
Date Created:	28th August 2024	Date Last Updated:	28th August 2024

Actor:	User
Description:	Allows users to flag either party .
Preconditions:	Entry Condition: Called by the use case **Chat** when the "Flag User" function is selected. The User must be logged in and authenticated.
Postconditions:	User is flagged.
Priority:	Low
Frequency of Use:	Medium
Flow of Events:	1. When either party has behaved inappropriately or in an offensive manner, either party shall flag the User and add the User to the list of flagged users.
Alternative Flows:	None
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.4.7 ViewReviews

Use Case ID:	#4-7		
Use Case Name:	ViewReviews		
Created By:	Lau Zhan You	Last Updated By:	Lau Zhan You
Date Created:	1st September 2024	Date Last Updated:	1st September 2024

Actor:	User
Description:	Allows Users to view the ratings and reviews that other Users have given a particular person.
Preconditions:	1. The User must be logged in and authenticated. 2. User has selected the STAR ratings icon.
Postconditions:	Successfully display all ratings and reviews of the specified user.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system will display the most updated list of reviews and star ratings that have been given to a specified user by other users.
Alternative Flows:	AF-S1: If the user selects the BACK navigation icon 1. If the user selects BACK, the system will cancel the operation and return the User back to the listings.
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.4.8 LeaveReview

Use Case ID:	#4-8		
Use Case Name:	LeaveReview		
Created By:	Lim Jing Rong	Last Updated By:	Lim Jing Rong
Date Created:	24th August 2024	Date Last Updated:	24th August 2024

Actor:	Tenant/Owner
Description:	Tenant/Owner selects the user to review.
Preconditions:	Tenant/Owner is logged in and is authenticated.
Postconditions:	Tenant/Owner can leave a review on the specified User.
Priority:	High
Frequency of Use:	Medium
Flow of Events:	1. Tenant/Owner will select the Owner/Tenant to review. 2. Tenant/Owner confirms the selection by pressing the “Submit/Update Review” button. 3. Tenant/Owner uses the included use case **SubmitReview** to submit the review.
Alternative Flows:	None
Exceptions:	1. If no Tenant/Owner has been selected, display an error message
Includes:	SubmitReview
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.4.9 SubmitReview

Use Case ID:	#4-9		
Use Case Name:	SubmitReview		
Created By:	Lim Jing Rong	Last Updated By:	Lim Jing Rong
Date Created:	24th August 2024	Date Last Updated:	24th August 2024

Actor:	Tenant/Owner
Description:	Tenant/Owner can submit a review on a Owner/Tenant by uploading pictures and writing a review text
Preconditions:	Tenant/Owner is logged in and is authenticated
Postconditions:	Review is successfully submitted
Priority:	High
Frequency of Use:	Medium
Flow of Events:	1. The system displays a “Leave Review” button within the “Rental Info” page for Tenant/Owner to submit a review on a particular Tenant/Owner. 2. When a Tenant selects the “Leave Review” button, the system will prompt the Tenant/Owner to enter rating, review title and text. 3. Tenant/Owner confirms the review by pressing the “Submit Review” button. 4. If the Tenant/Owner selects the “Submit Review” Button, the system will save the review (title, text and rating) in the database and display that the review is submitted.
Alternative Flows:	None
Exceptions:	1. If any required information is missing or invalid, display an error message.
Includes:	1. FlagReview 2. EditReview 3. DeleteReview

Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.4.10 FlagReview

Use Case ID:	#4-10		
Use Case Name:	FlagReview		
Created By:	Lim Jing Rong	Last Updated By:	Lim Jing Rong
Date Created:	24th August 2024	Date Last Updated:	24th August 2024

Actor:	Tenant/Owner
Description:	Tenant/Owner can flag the review if it is inappropriate.
Preconditions:	Tenant/Owner is logged in and is authenticated.
Postconditions:	Flag is sent to admin to process the disputes.
Priority:	Medium
Frequency of Use:	Low
Flow of Events:	1. If the Tenant/Owner deems that the review is inappropriate, they can submit a request to flag the review to the admins.
Alternative Flows:	AF-S1: If the user selects CANCEL 1. If the user selects CANCEL, the system will cancel the operation and return the User to the review.
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

4.2.4.11 EditReview

Use Case ID:	#4-11		
Use Case Name:	EditReview		
Created By:	Lim Jing Rong	Last Updated By:	Lim Jing Rong
Date Created:	24th August 2024	Date Last Updated:	24th August 2024

Actor:	Tenant/Owner
Description:	Tenant/Owner can edit his/her review by uploading pictures, removing uploaded pictures or editing the review text.
Preconditions:	Tenant/Owner is logged in and is authenticated.
Postconditions:	The Review is successfully edited.
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. The Tenant/Owner enters the inbox and selects the house. 2. The system will display a “Leave Review” button. 3. Upon Clicking it, it will bring the Tenant/Owner to the review page. 4. The system will display a “Submit/Update Review” button to allow the Tenant/Owner to save the changes made. 5. The system will display a “Delete Review” button to allow the Tenant/Owner to abandon the changes made. 6. When the “Delete Review” button is selected, the use case DeleteReview will be used.
Alternative Flows:	AF-S4: If the “Cancel” button is selected 1. The system will not save the changes made. 2. The system will redirect the Consumer away from the Edit Review Screen.
Exceptions:	None
Includes:	None
Special Requirements:	None

Assumptions:	None
Notes and Issues:	None

4.2.4.12 DeleteReview

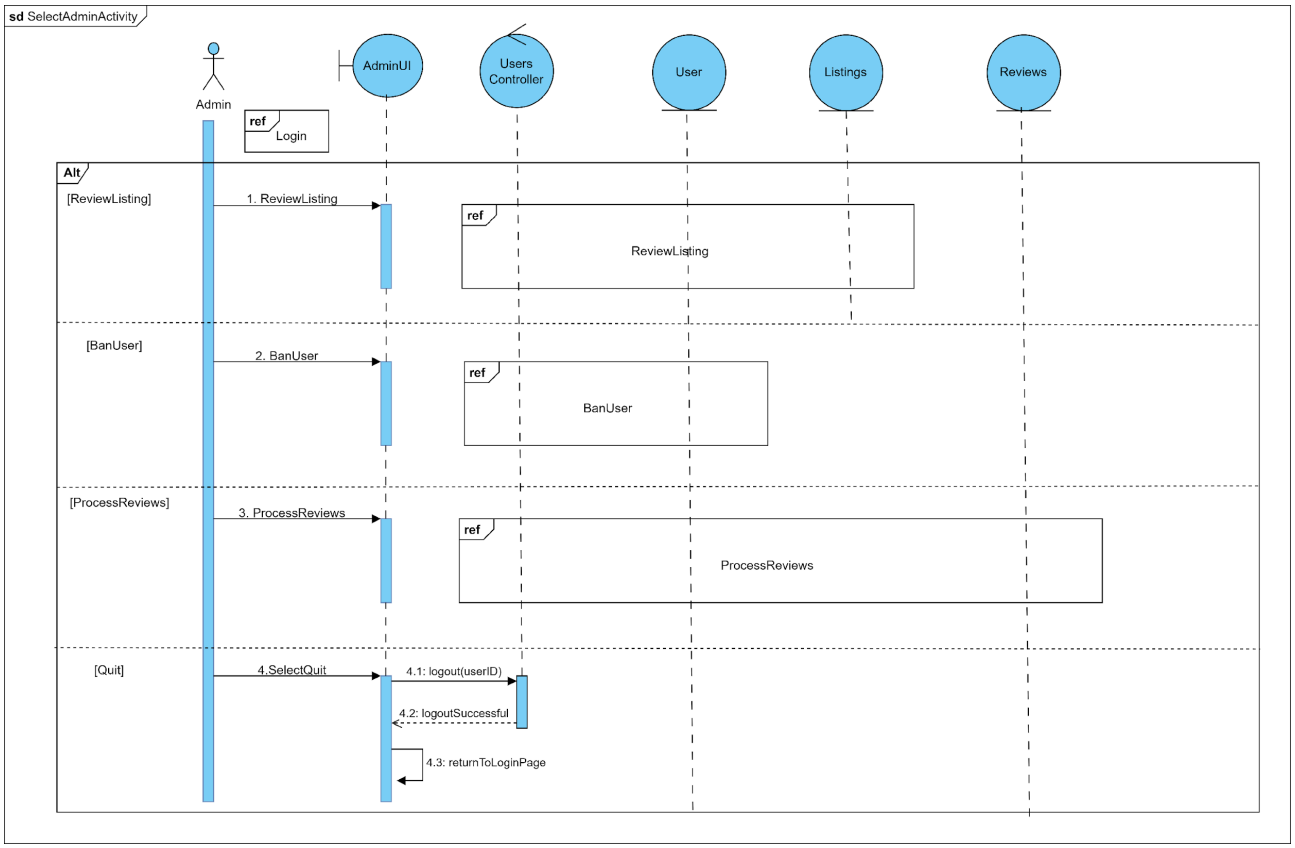
Use Case ID:	#4-12		
Use Case Name:	DeleteReview		
Created By:	Lim Jing Rong	Last Updated By:	Lim Jing Rong
Date Created:	24th August 2024	Date Last Updated:	24th August 2024

Actor:	Tenant/Owner
Description:	Tenant/Owner can delete his/her review
Preconditions:	Tenant/Owner is logged in and is authenticated
Postconditions:	The Review has successfully been deleted
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. If the Tenant/Owner has already made a review, the system will display an “Delete Review” button beside his/her review. 2. When the Tenant/Owner selects the “Delete Review” button, the system will delete this review and remove this review from the database.
Alternative Flows:	None
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

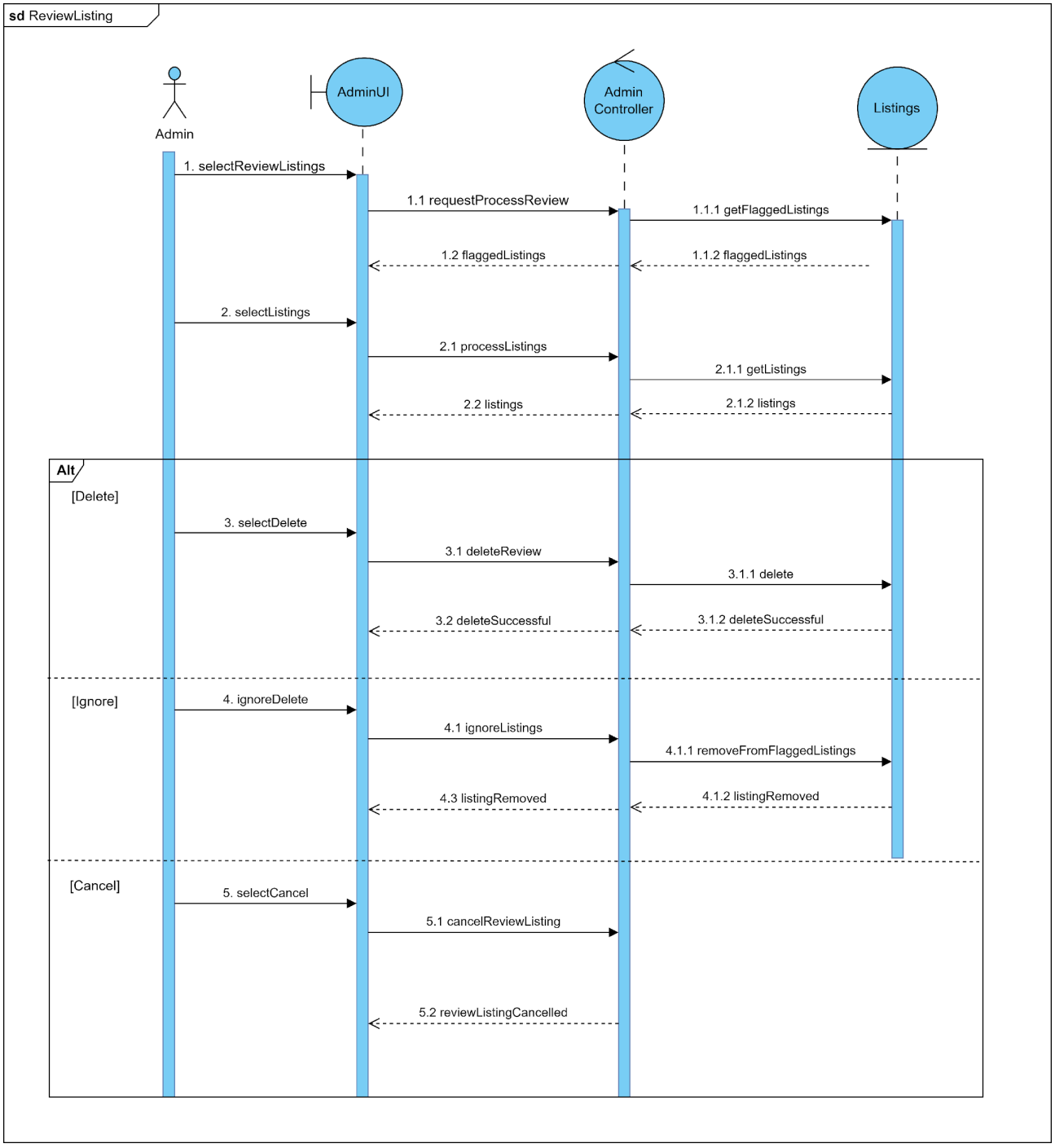
4.3 Sequence Diagrams for Use Cases

4.3.1 For Use Cases under #1

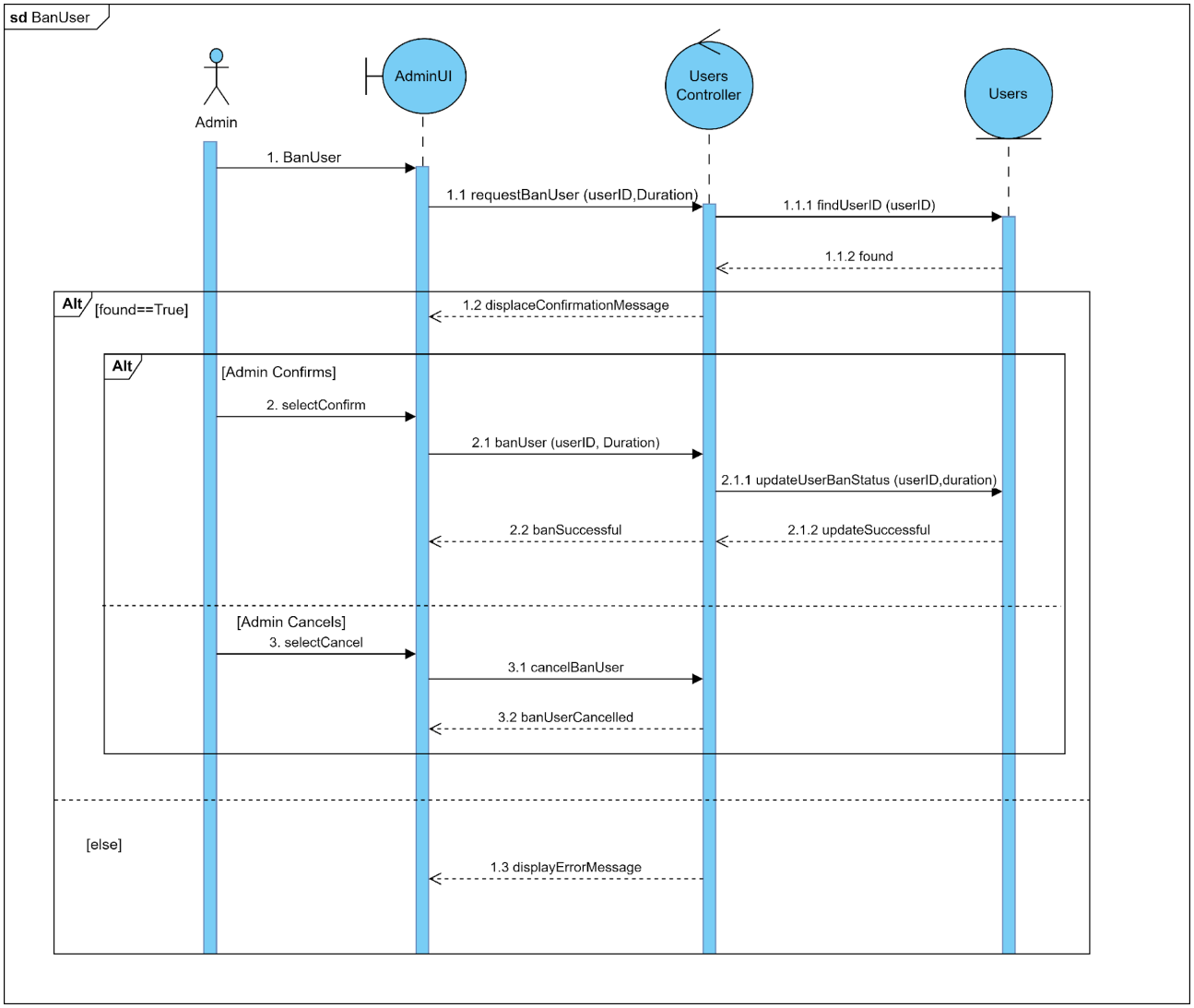
4.3.1.1 SelectAdminActivity



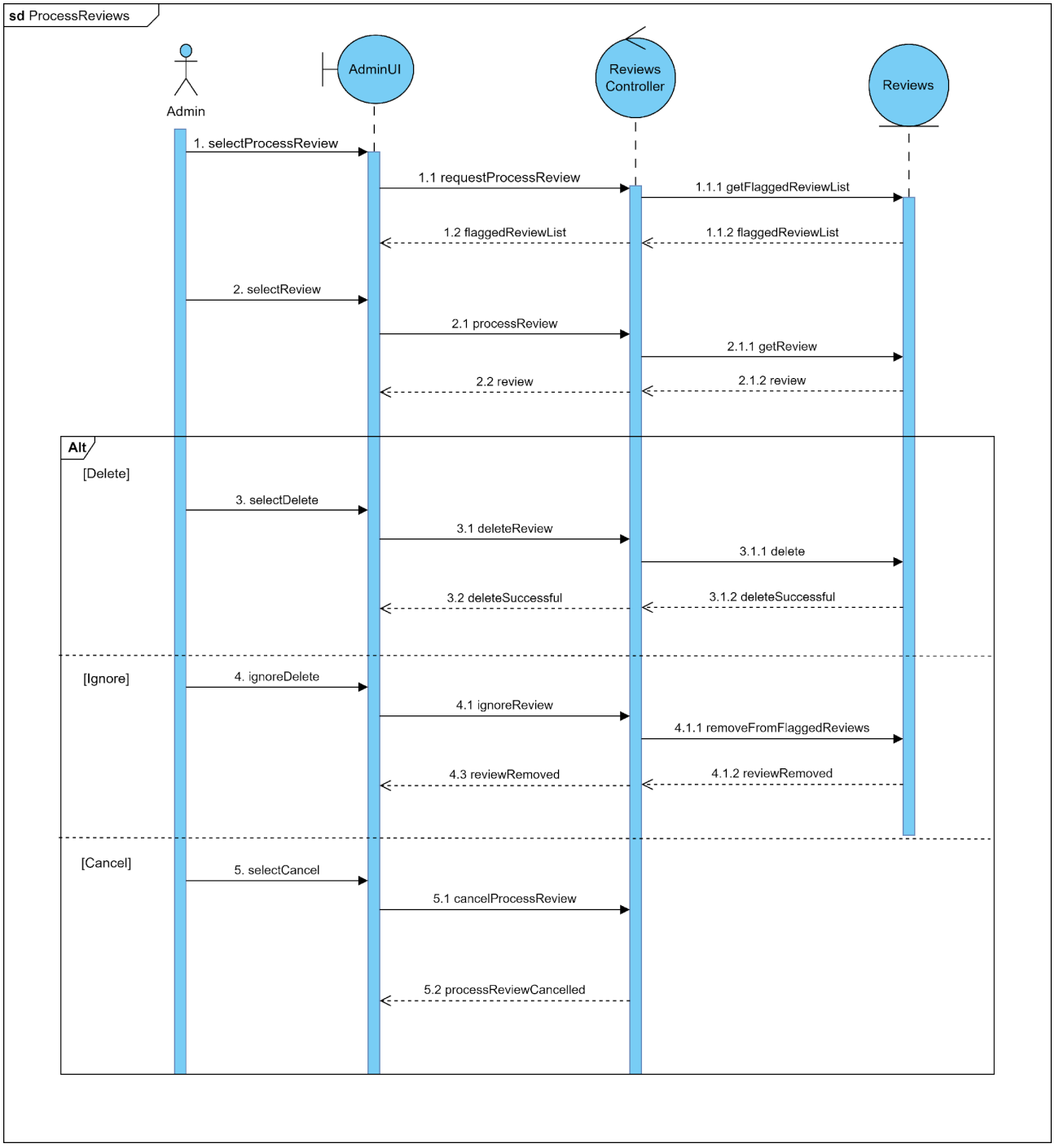
4.3.1.2 ReviewListing



4.3.1.3 BanUser

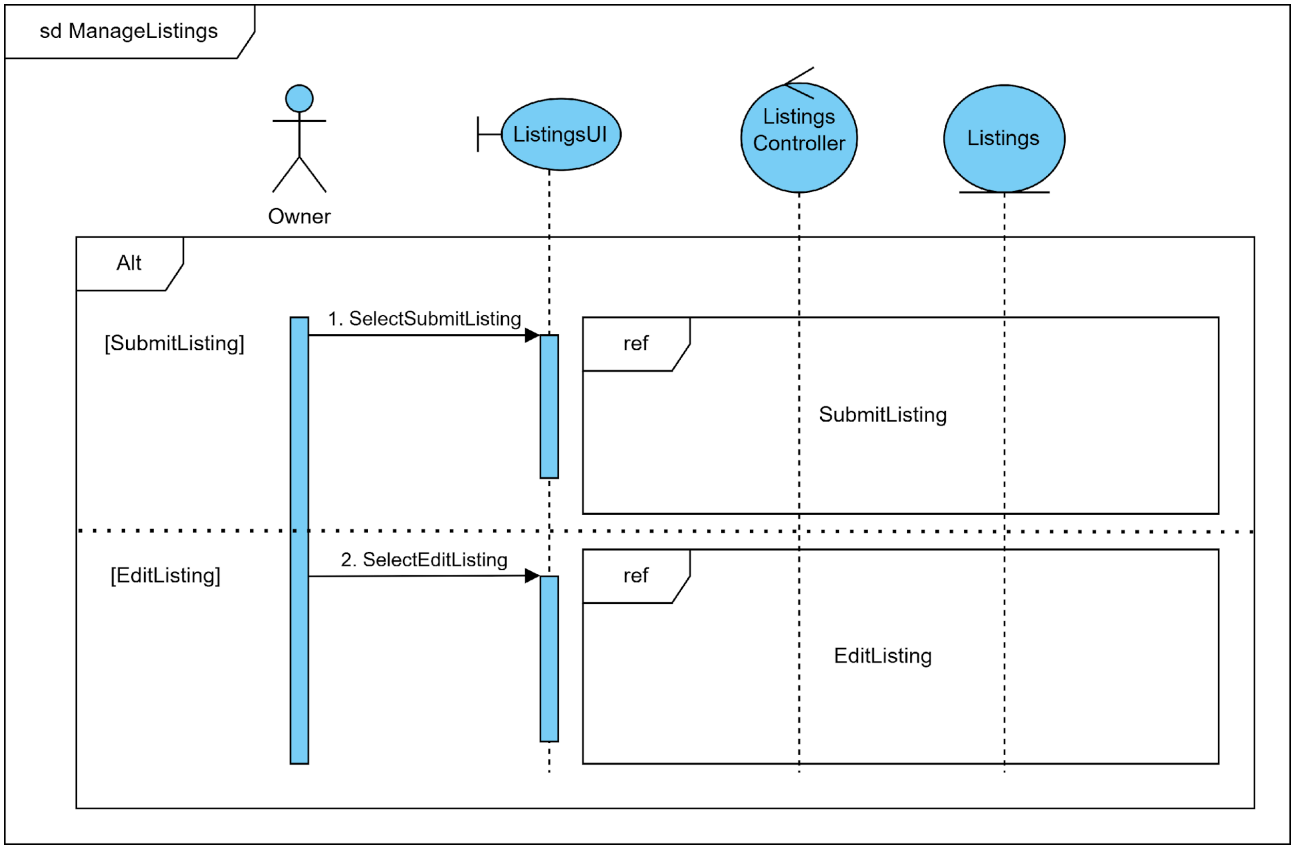


4.3.1.4 ProcessReviews

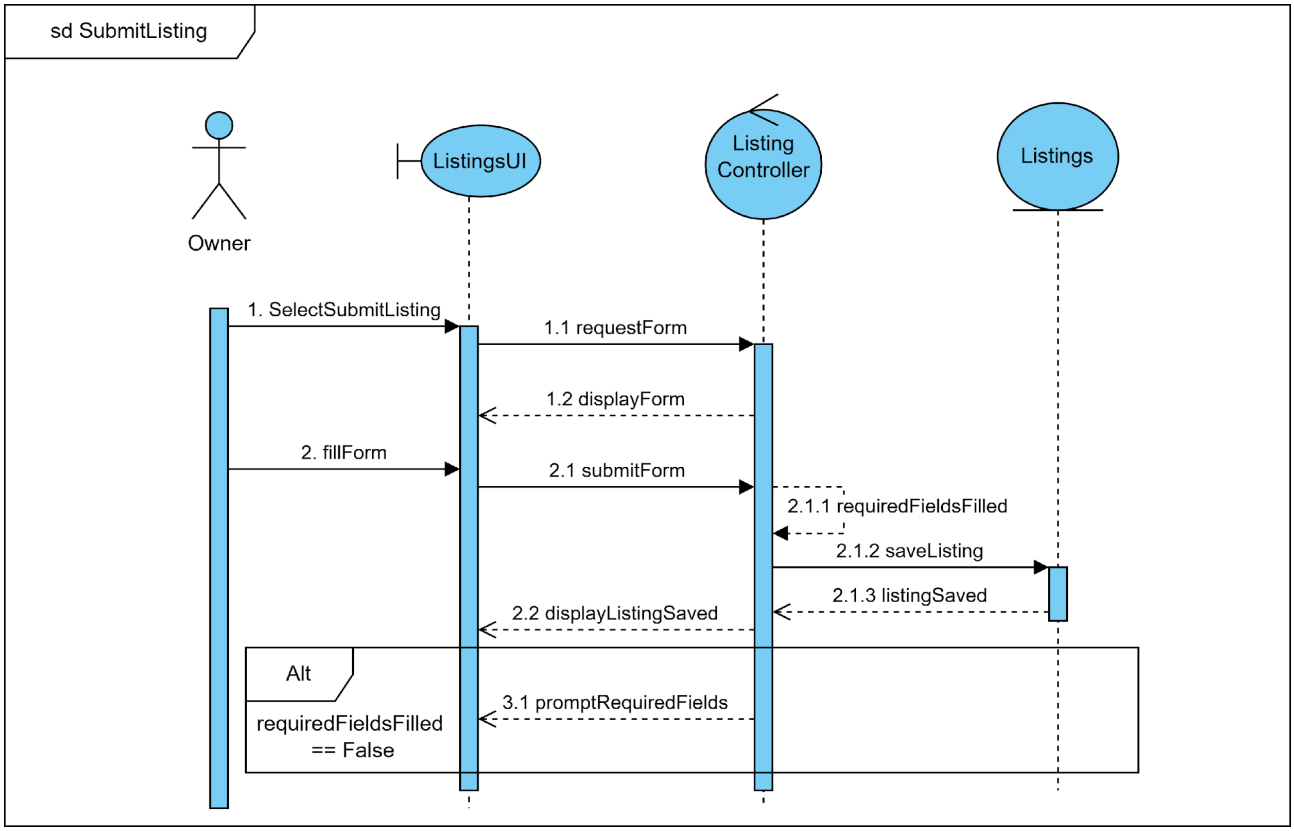


4.3.2 For Use Cases under #2

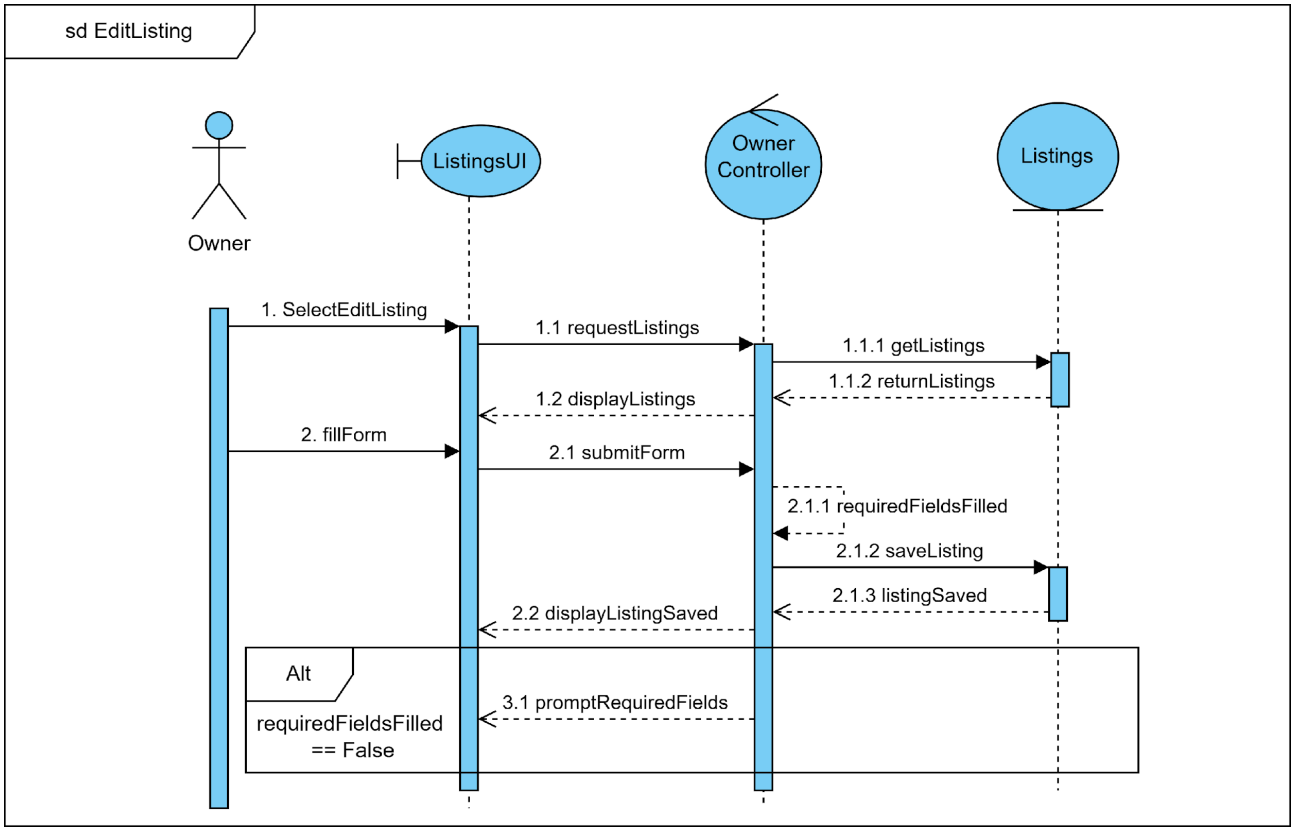
4.3.2.1 ManageListings



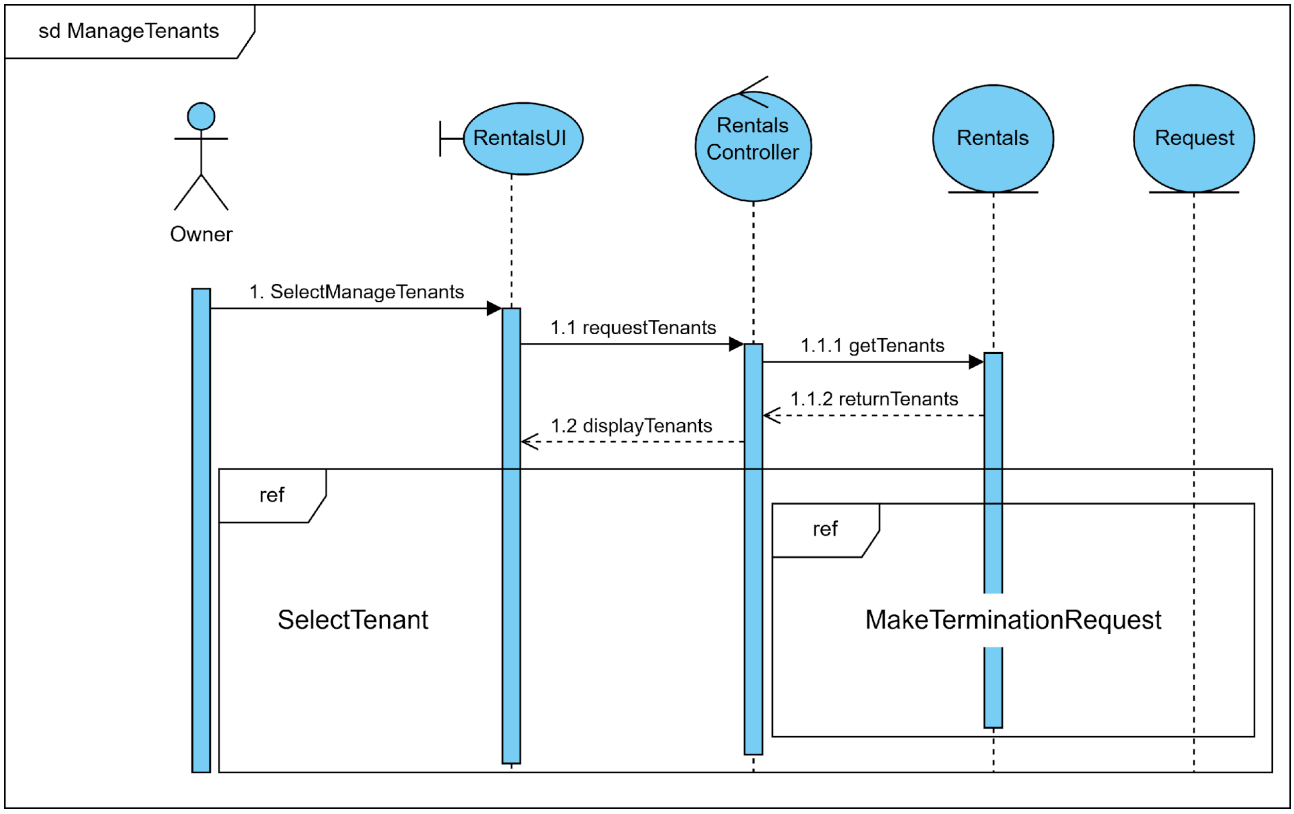
4.3.2.2 SubmitListing



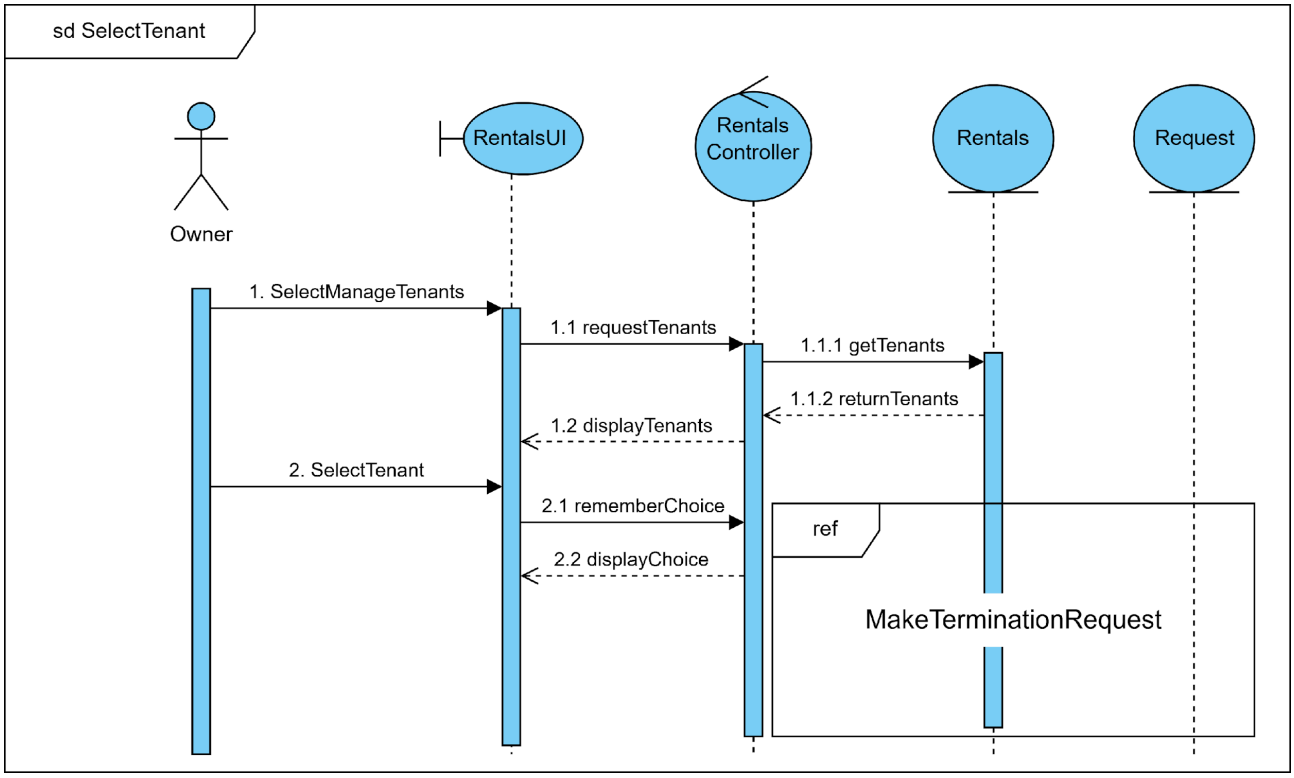
4.3.2.3 EditListing



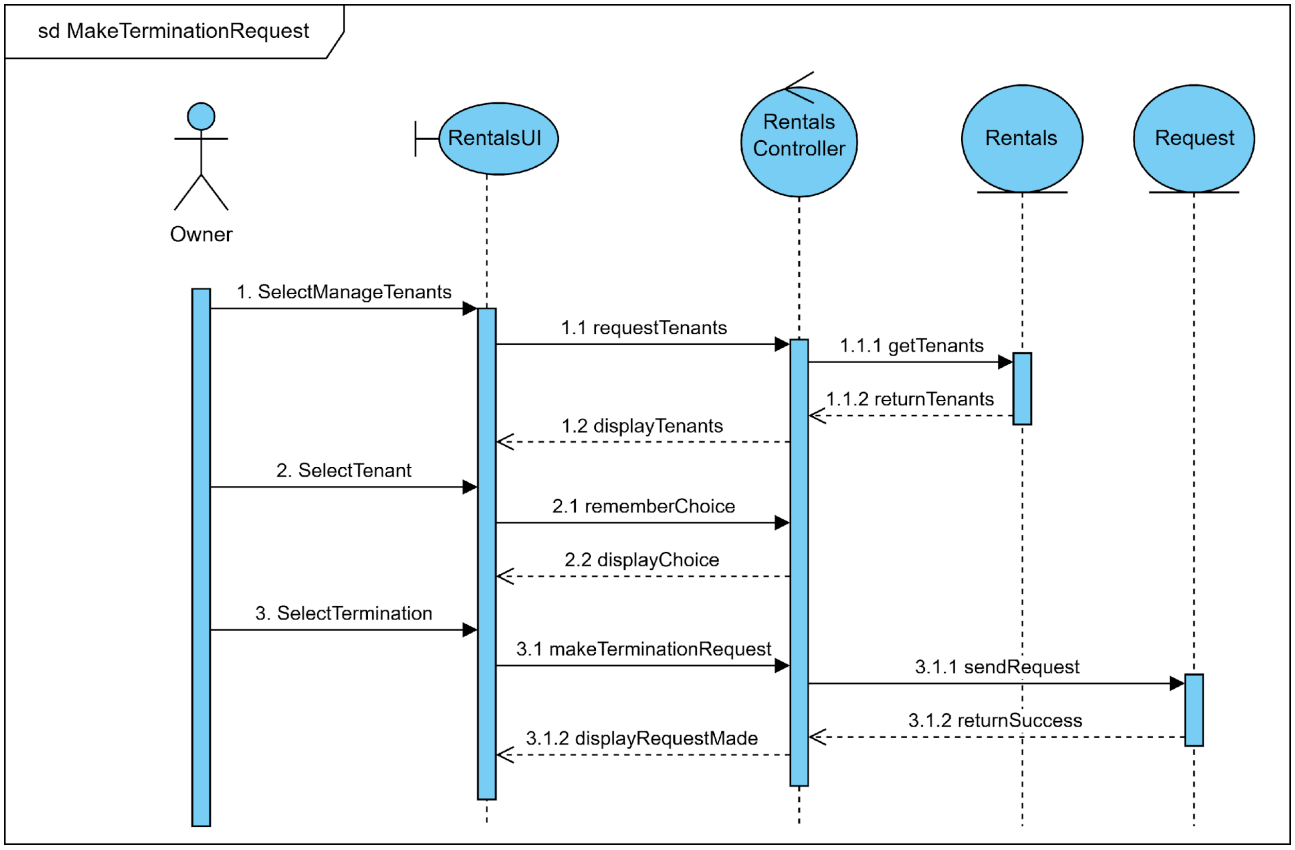
4.3.2.4 ManageTenants



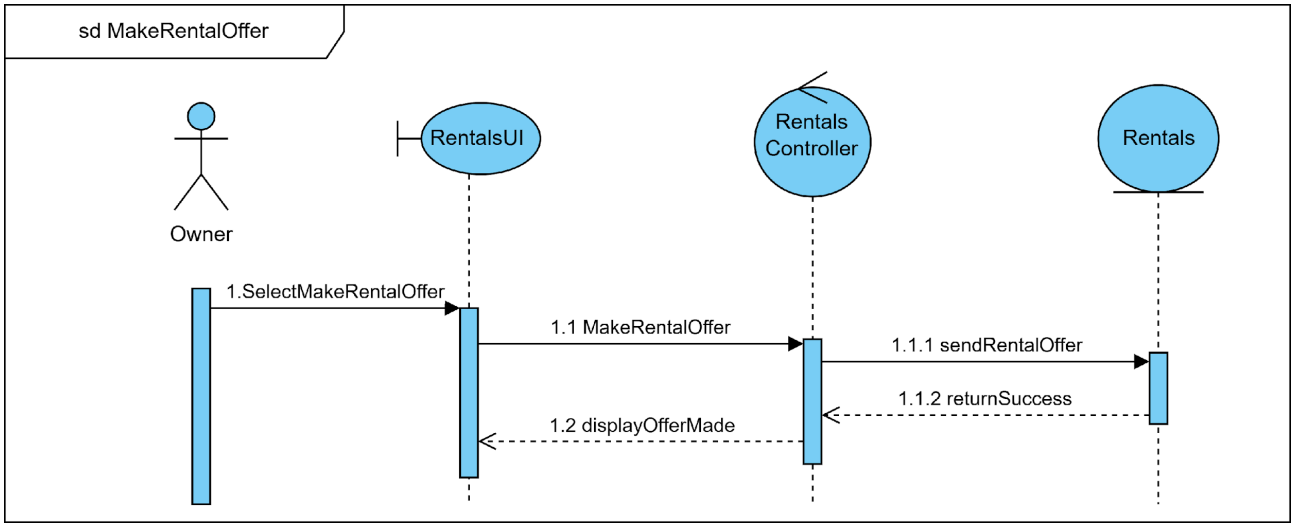
4.3.2.5 SelectTenant



4.3.2.6 MakeTerminationRequest

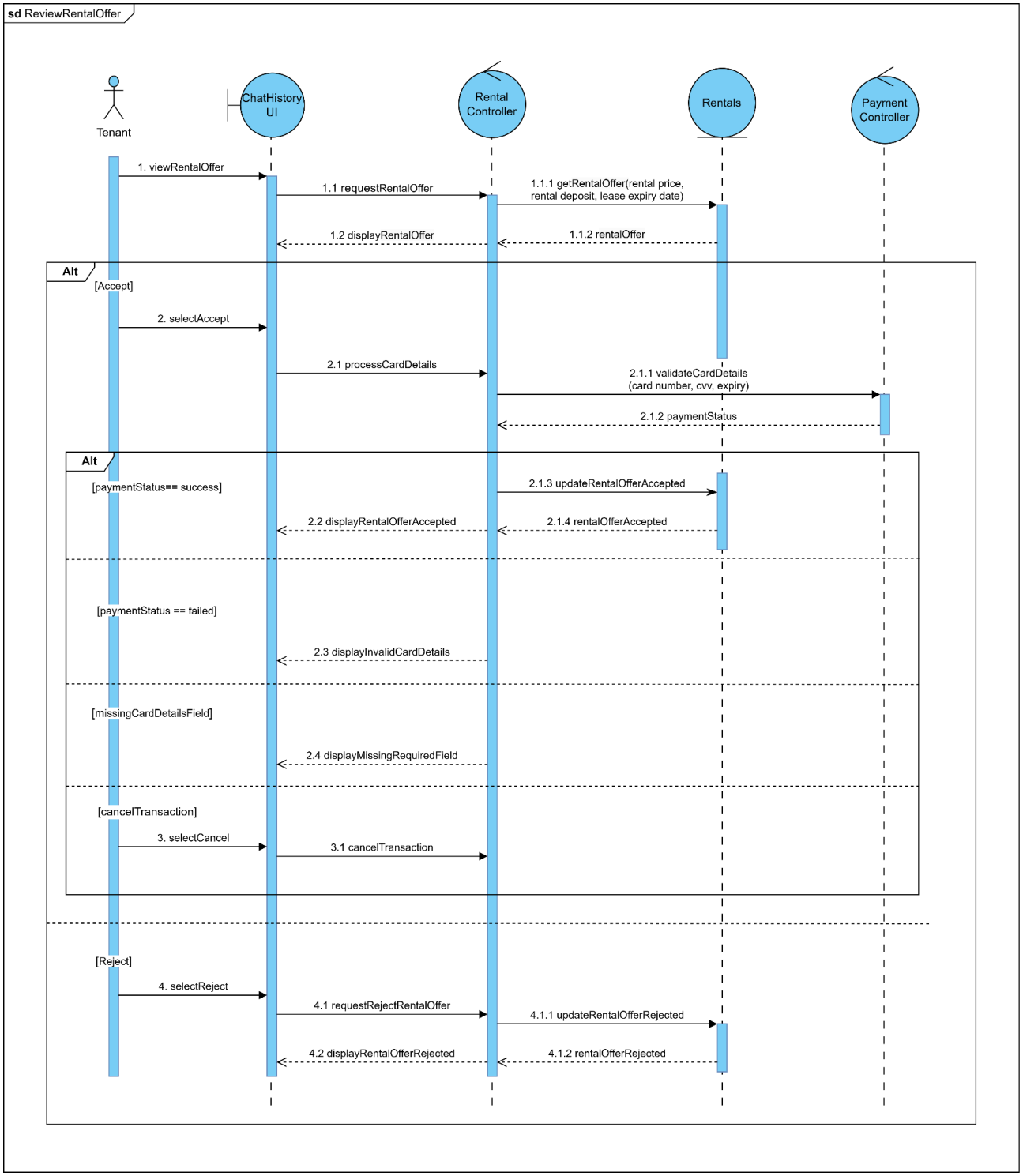


4.3.2.7 MakeRentalOffer

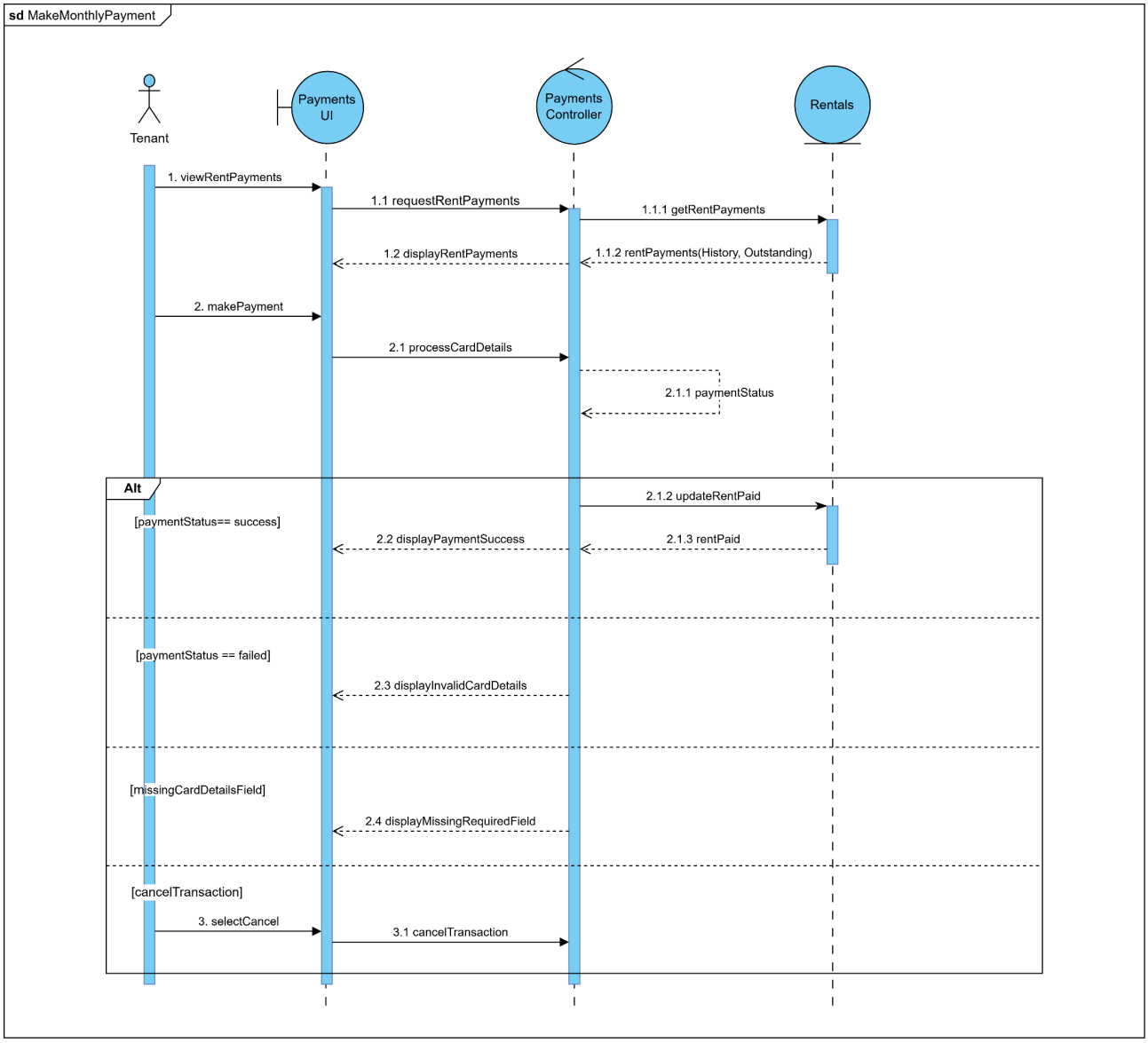


4.3.3 For Use Cases under #3

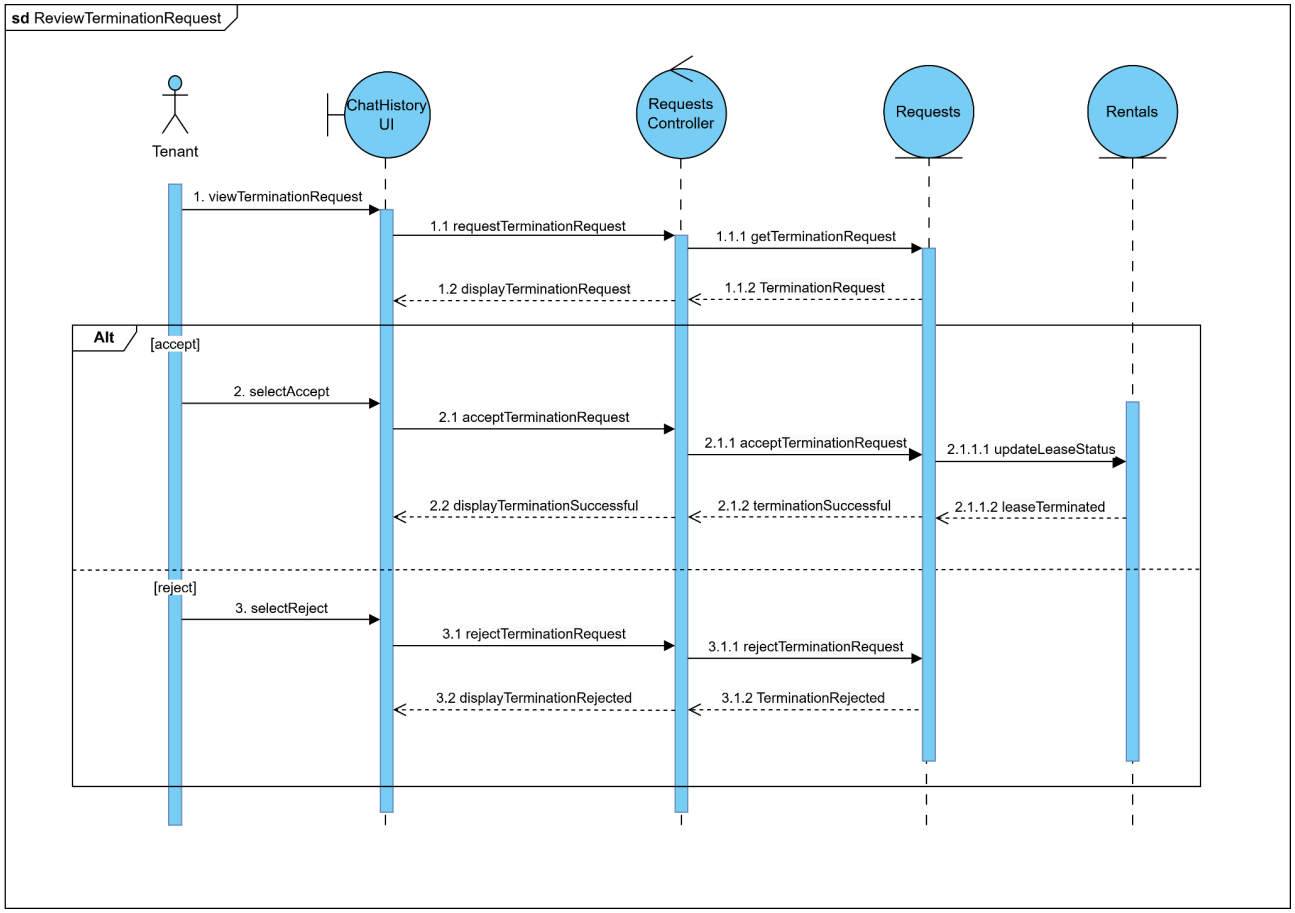
4.3.3.1 ReviewRentalOffer



4.3.3.2 MakeMonthlyPayment

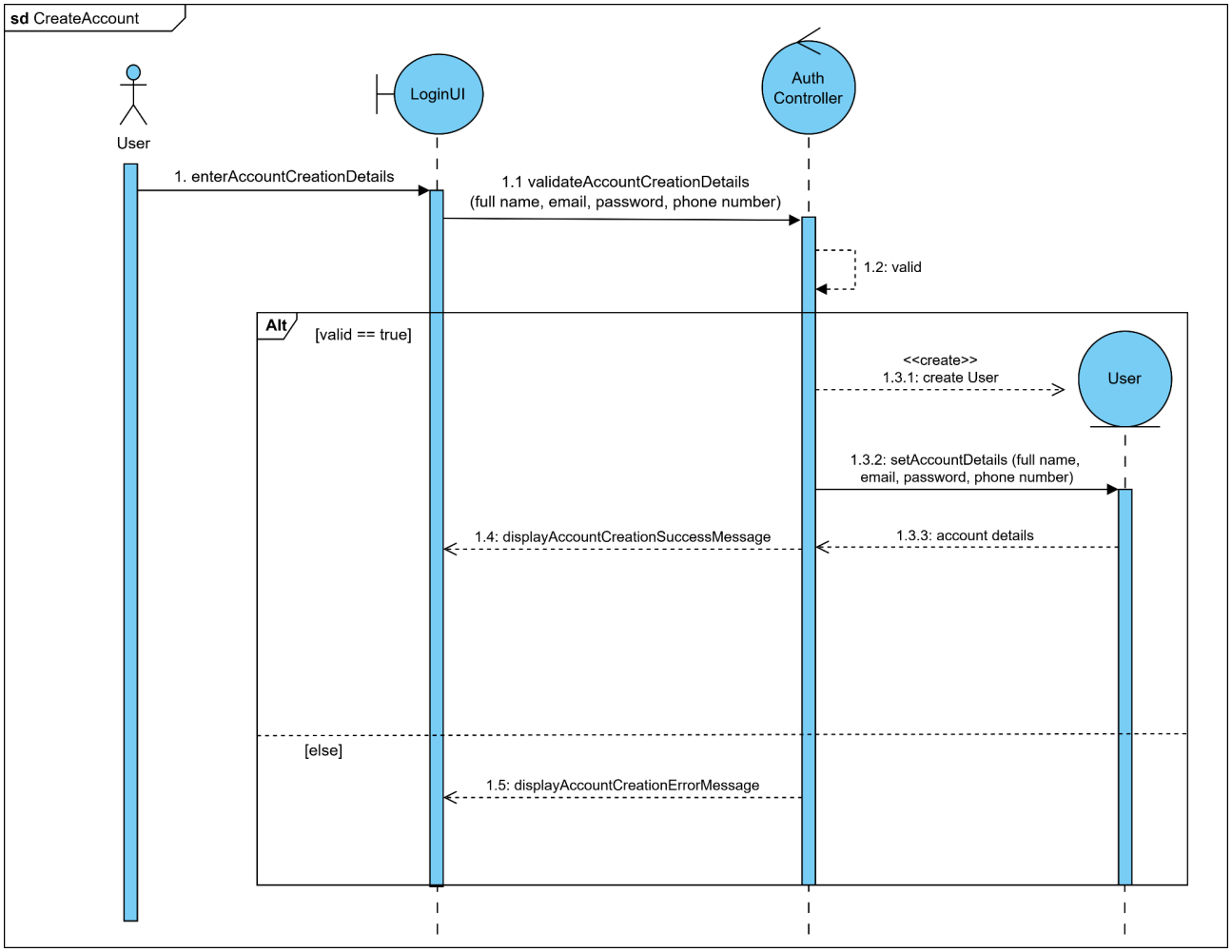


4.3.3.3 ReviewTerminationRequest

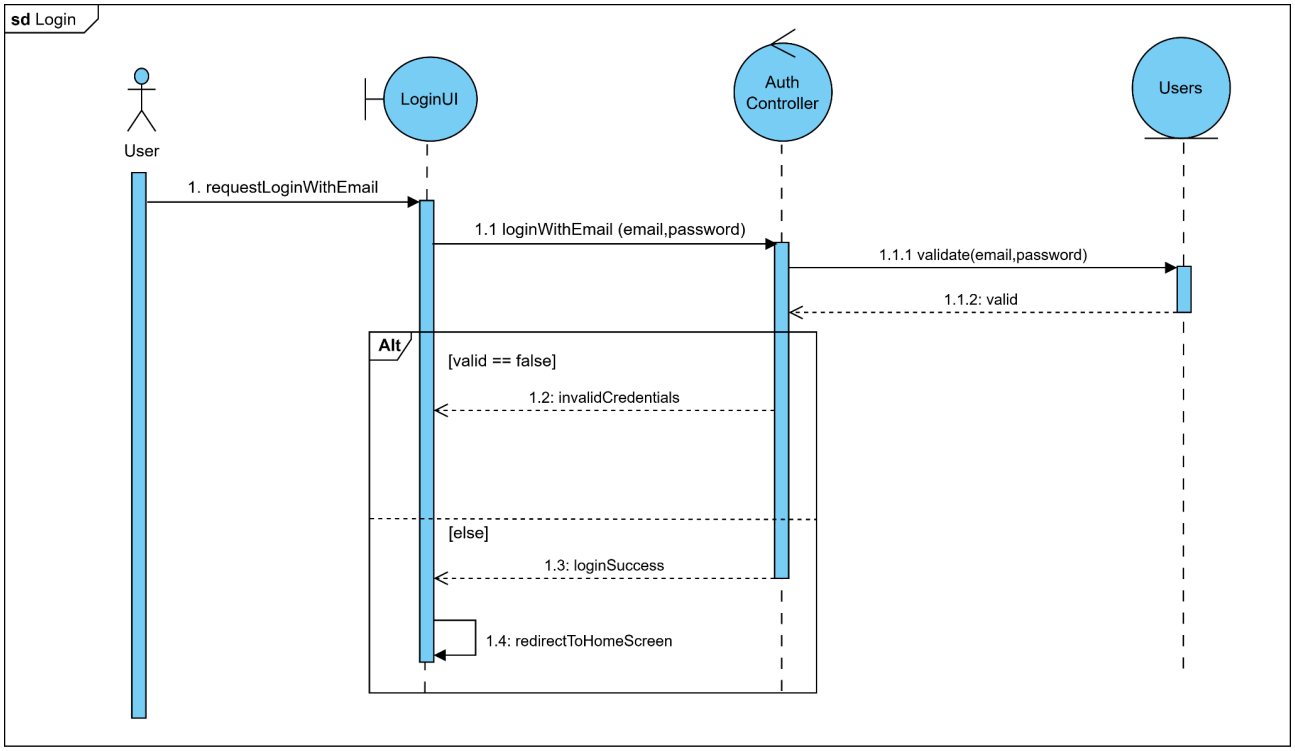


4.3.4 For Use Cases under #4

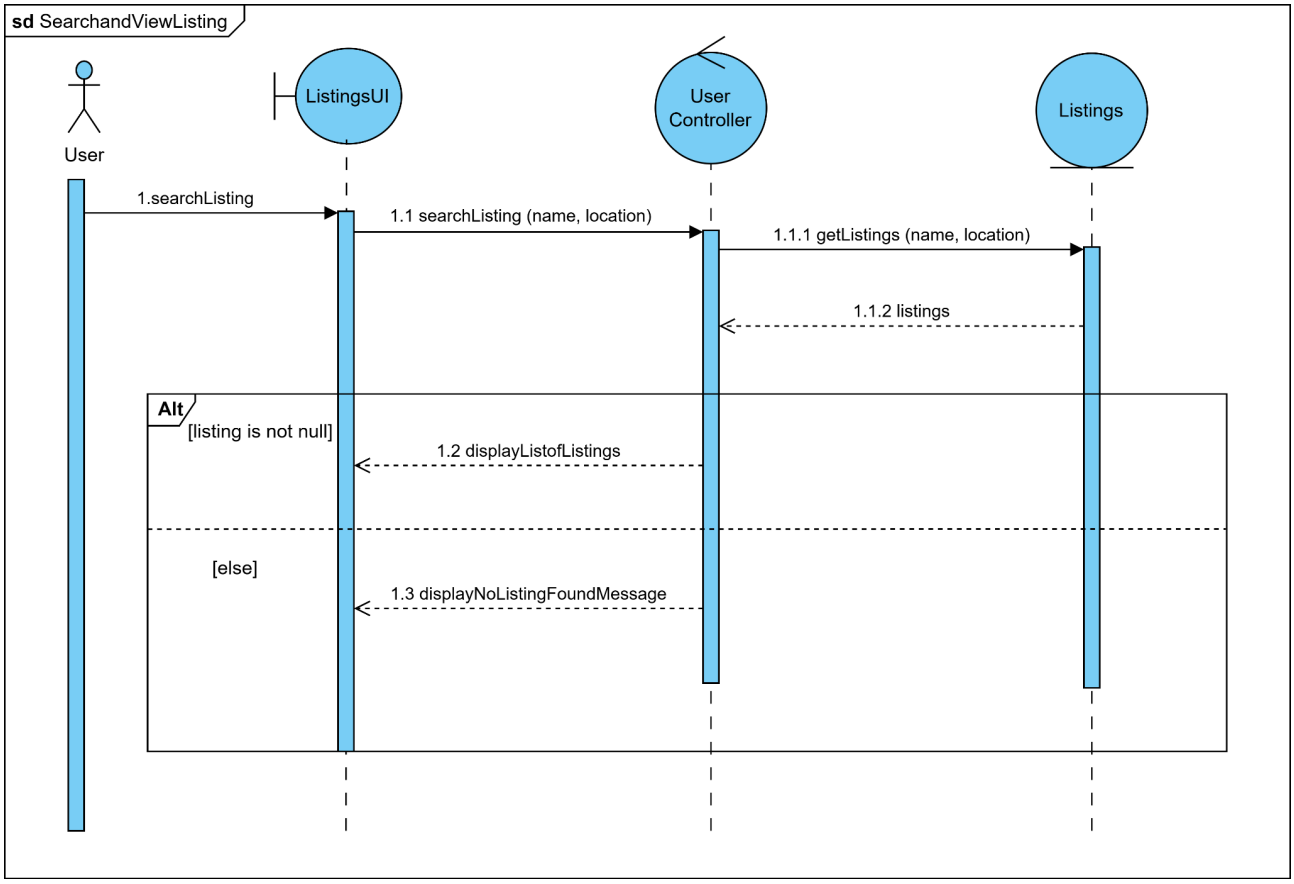
4.3.4.1 Create Account



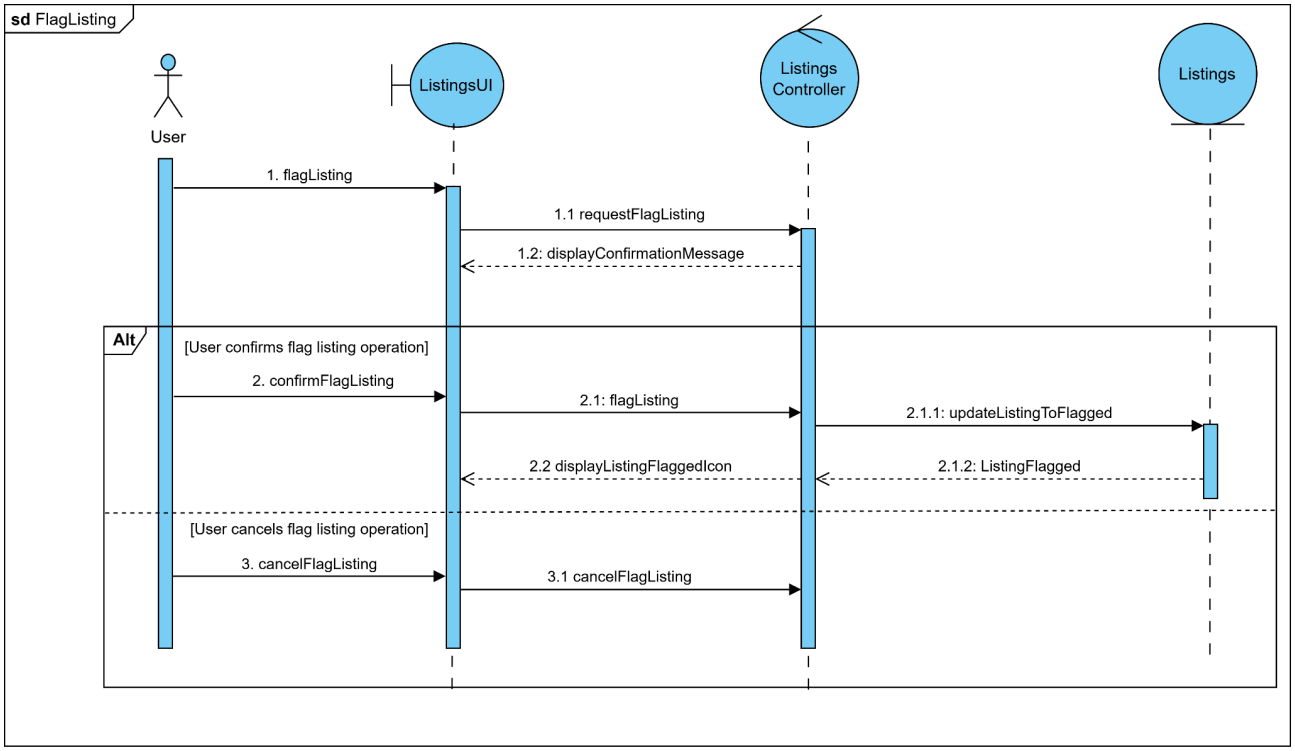
4.3.4.2 Login


sd Login
User
LoginUI
Auth Controller
Users
1. requestLoginWithEmail
1.1 loginWithEmail (email,password)
1.1.1 validate(email,password)
1.1.2: valid
Alt
[valid == false]
1.2: invalidCredentials
[else]
1.3: loginSuccess
1.4: redirectToHomeScreen

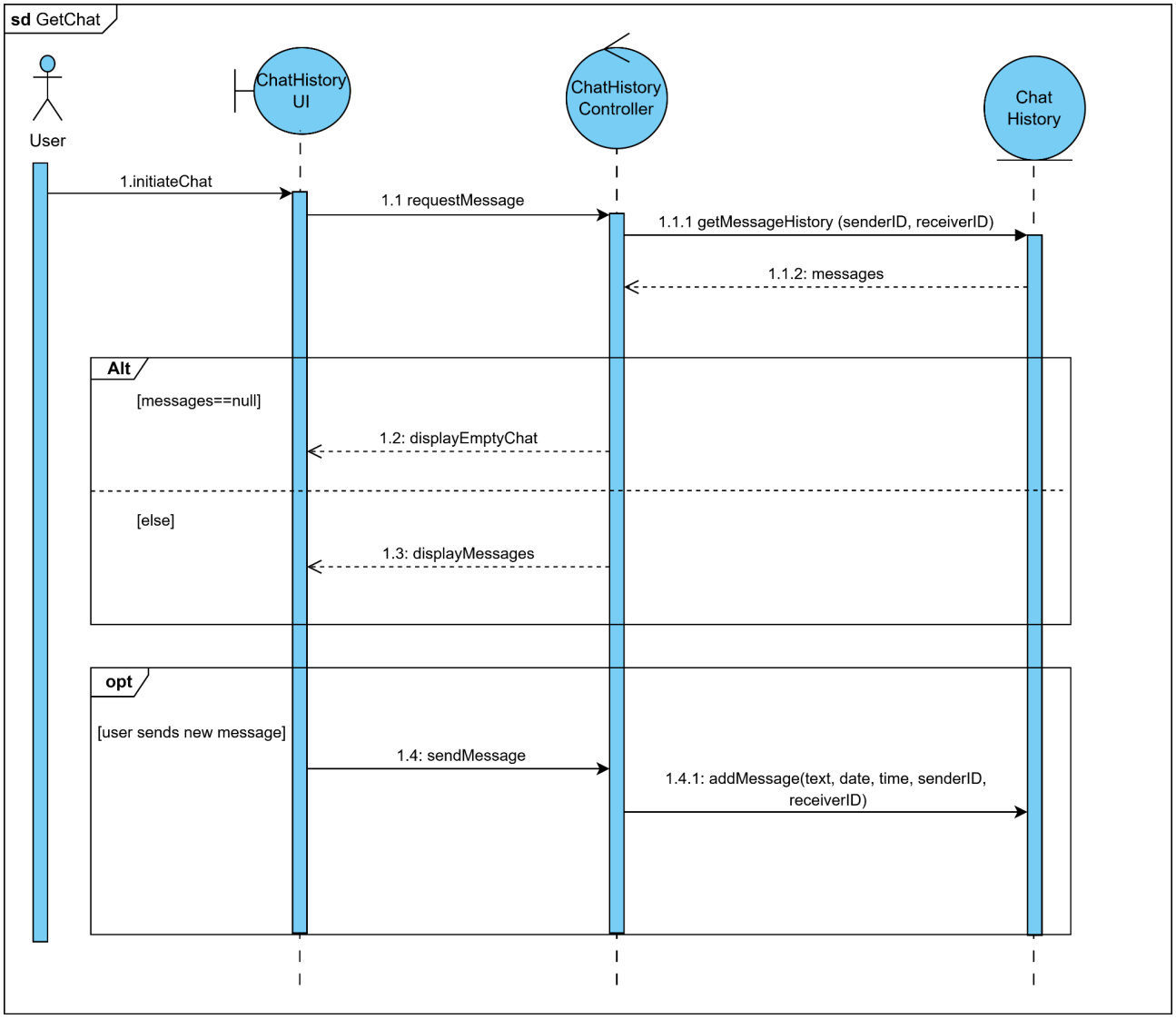
4.3.4.3 SearchAndViewListing



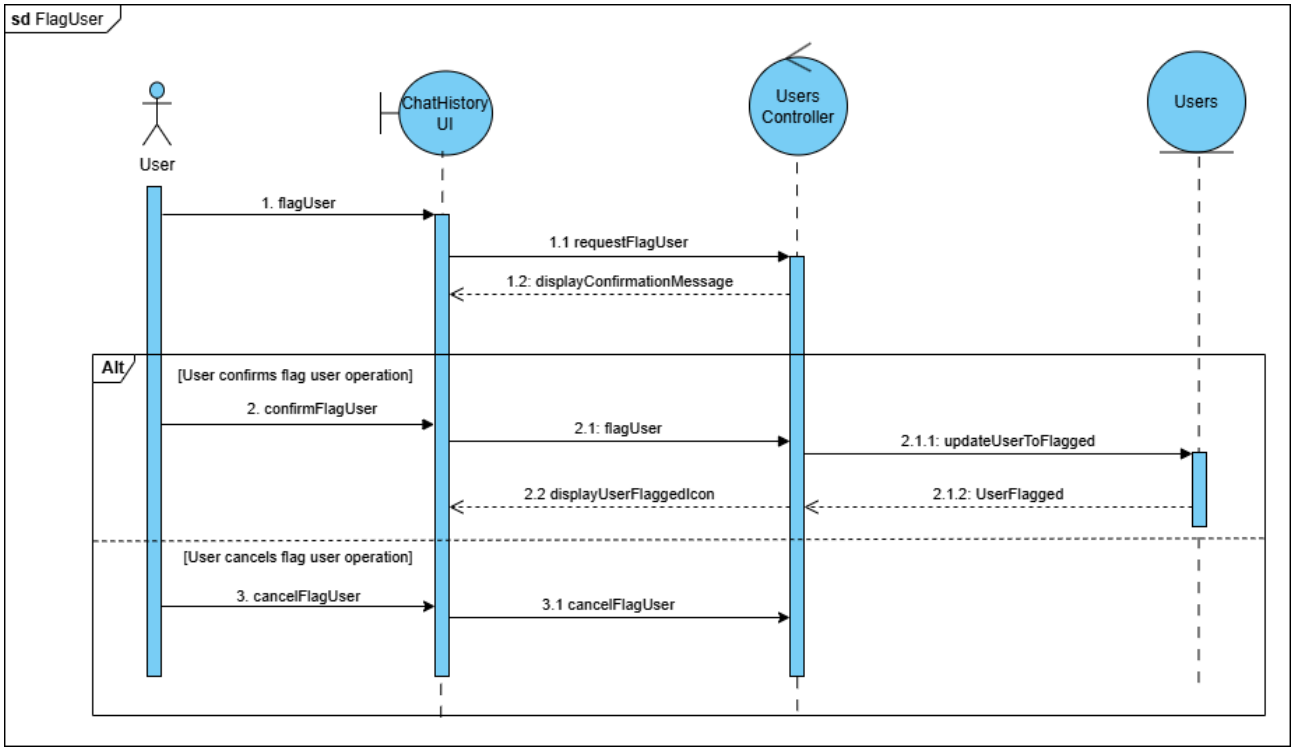
4.3.4.4 FlagListing



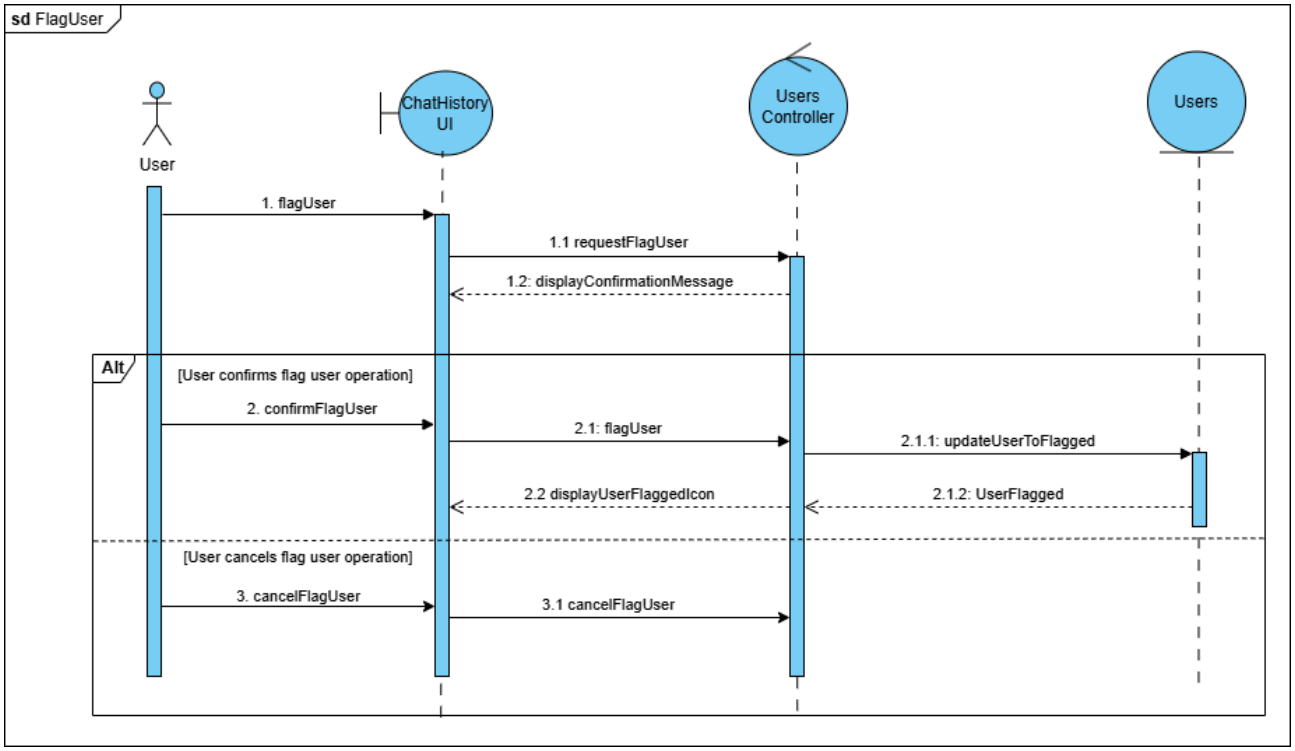
4.3.4.5 GetChat



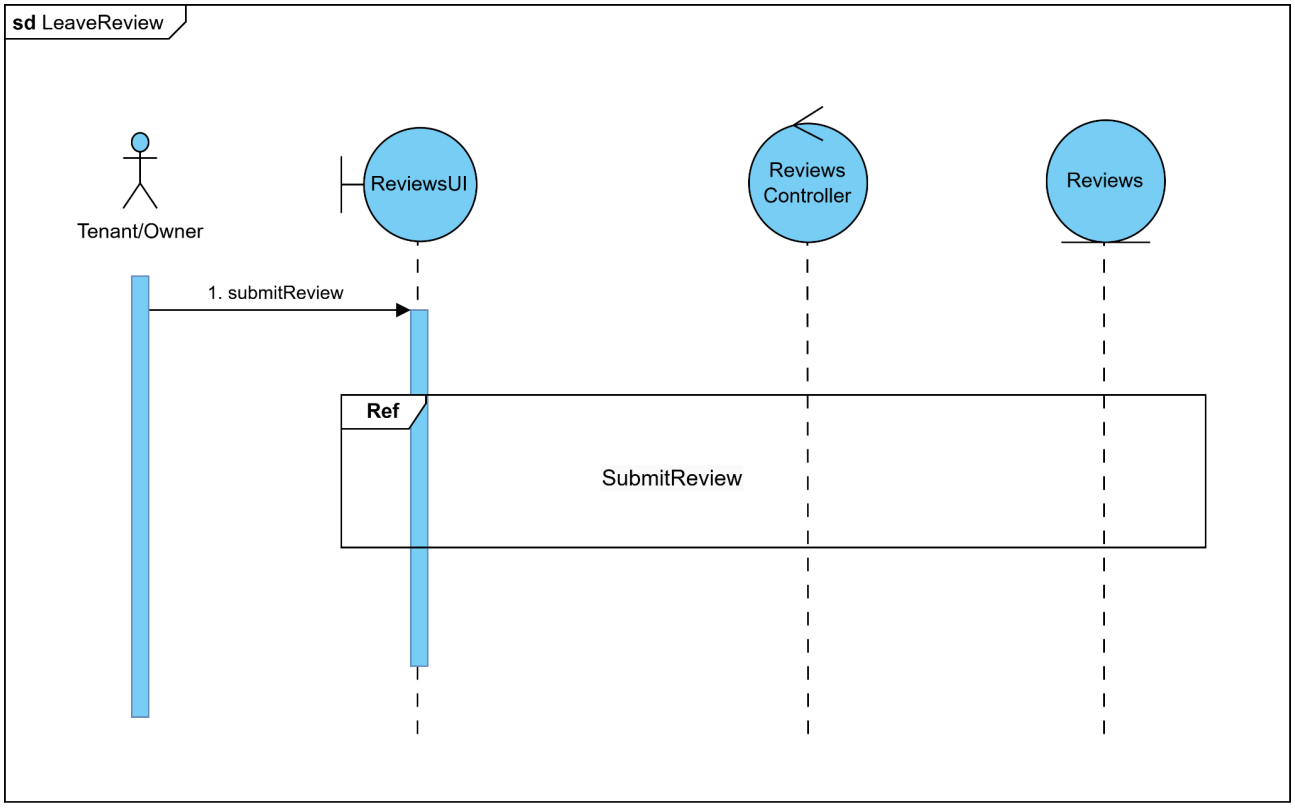
4.3.4.6 FlagUser



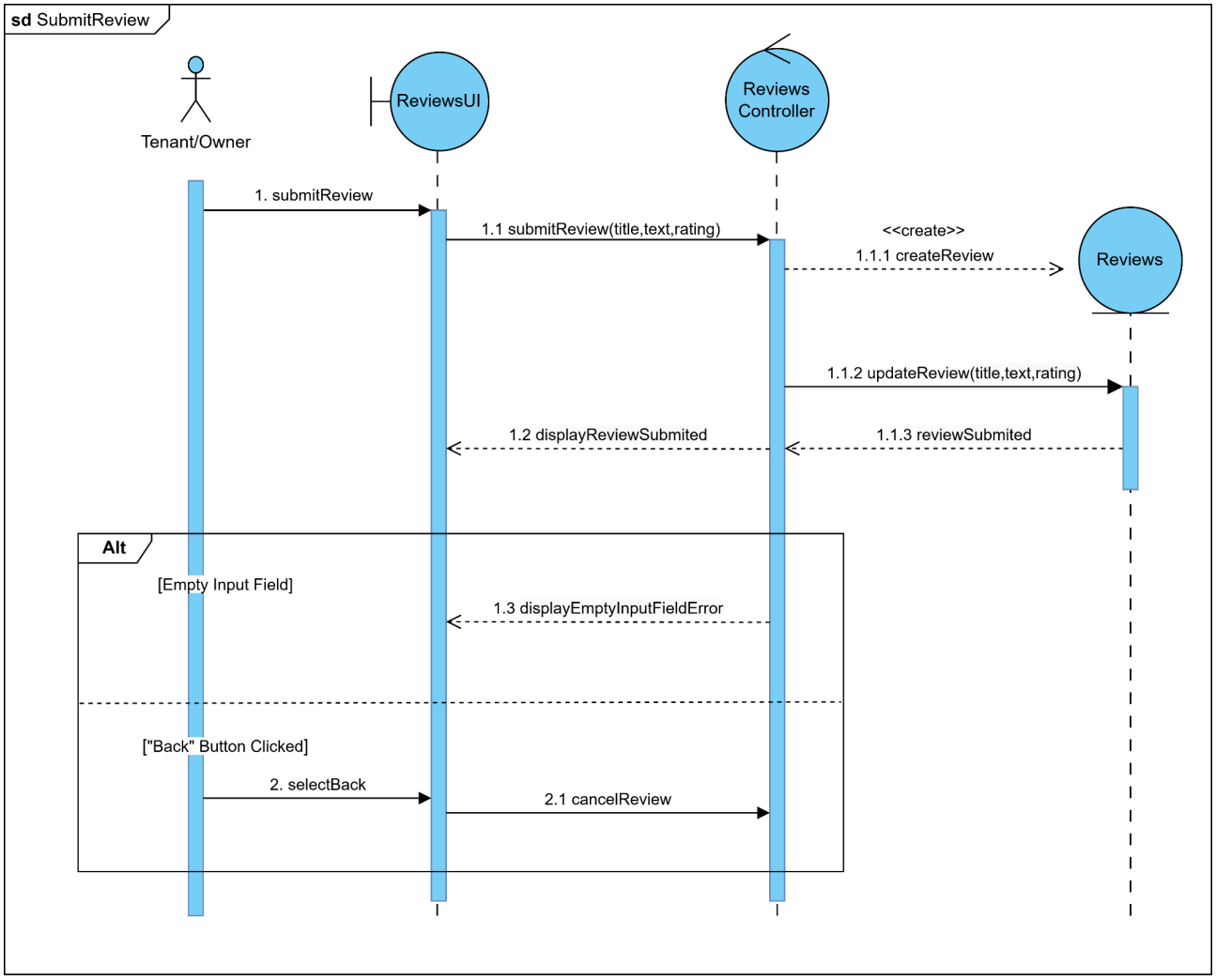
4.3.4.7 ViewReviews



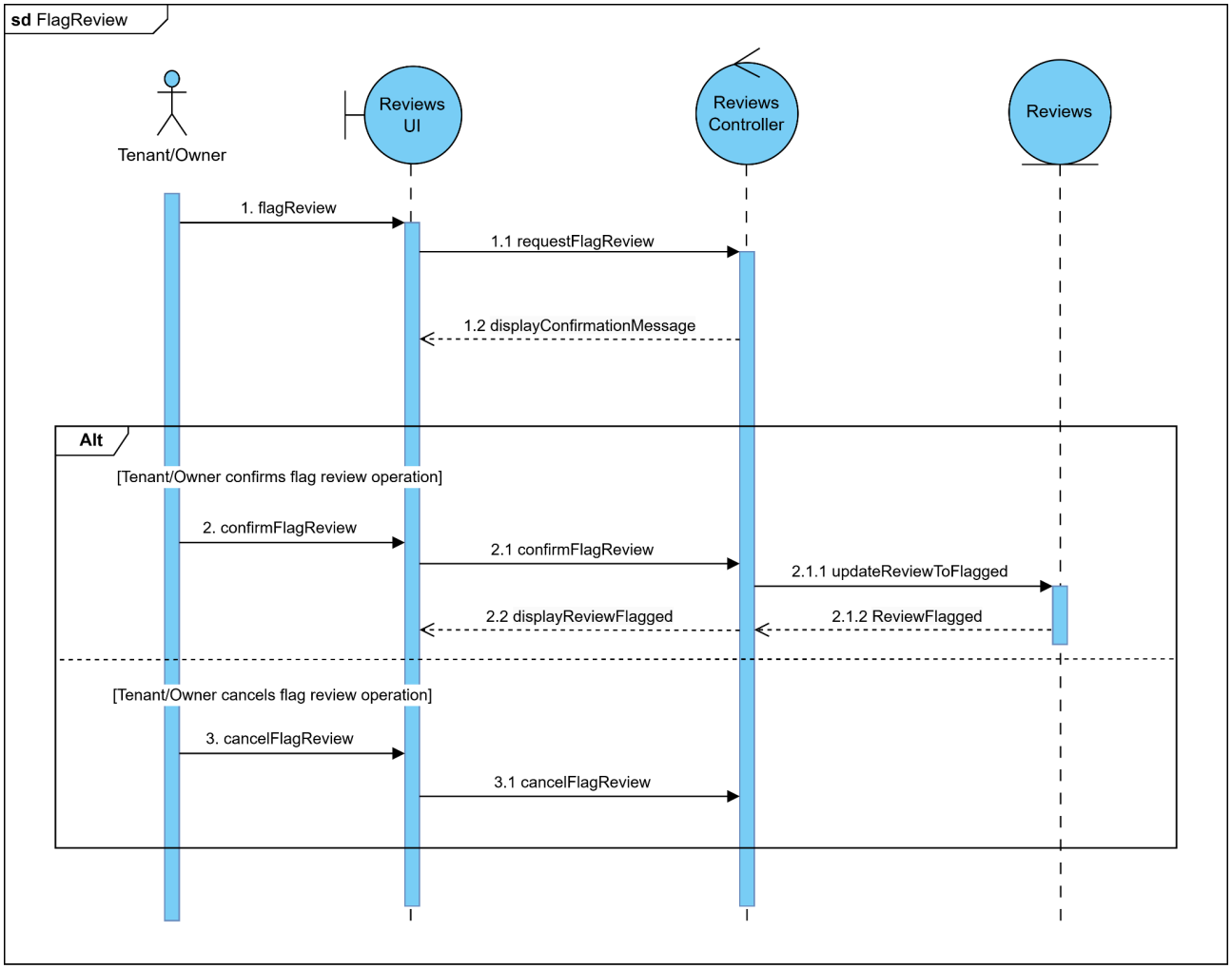
4.3.4.8 LeaveReview



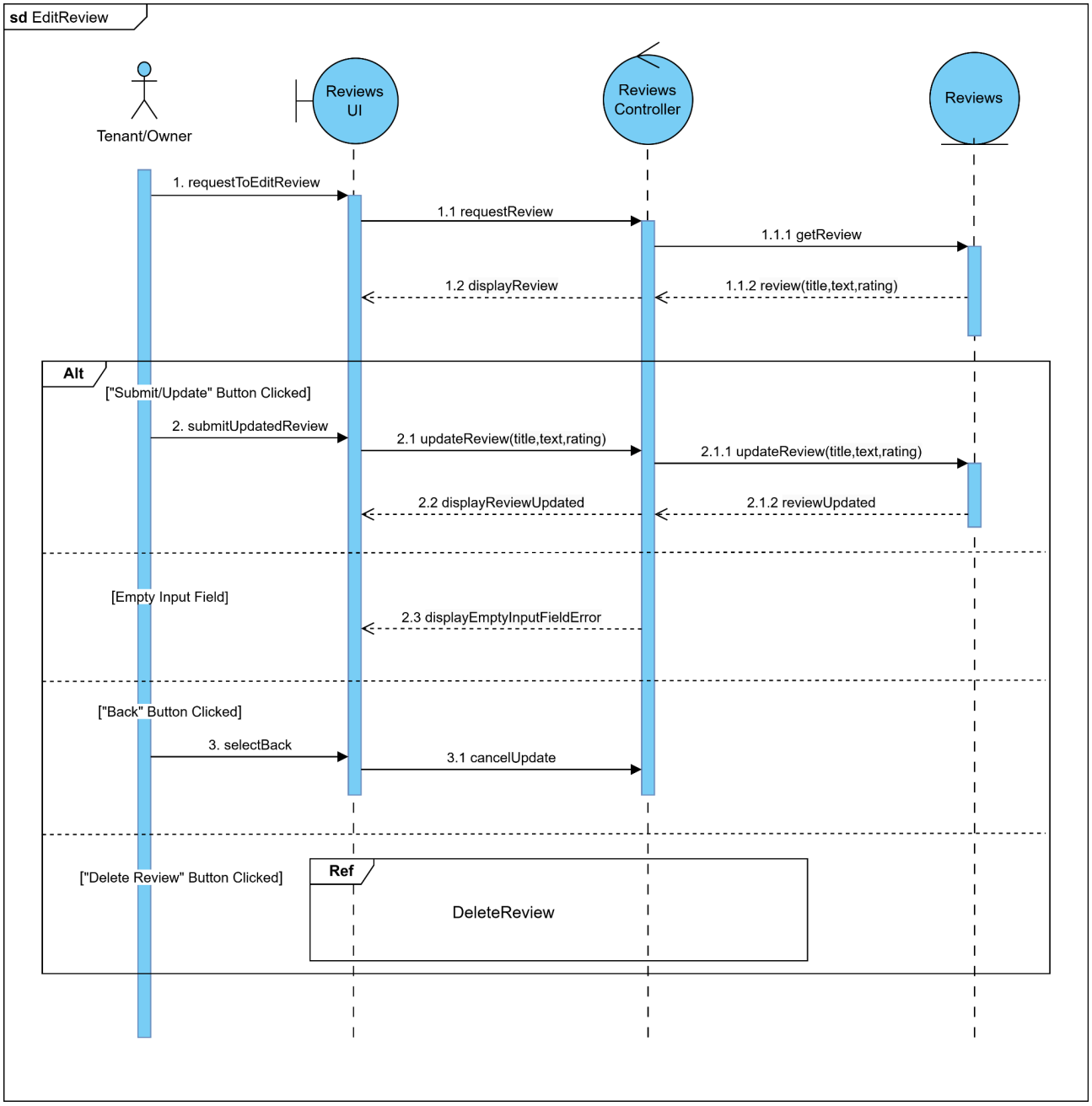
4.3.4.9 SubmitReview



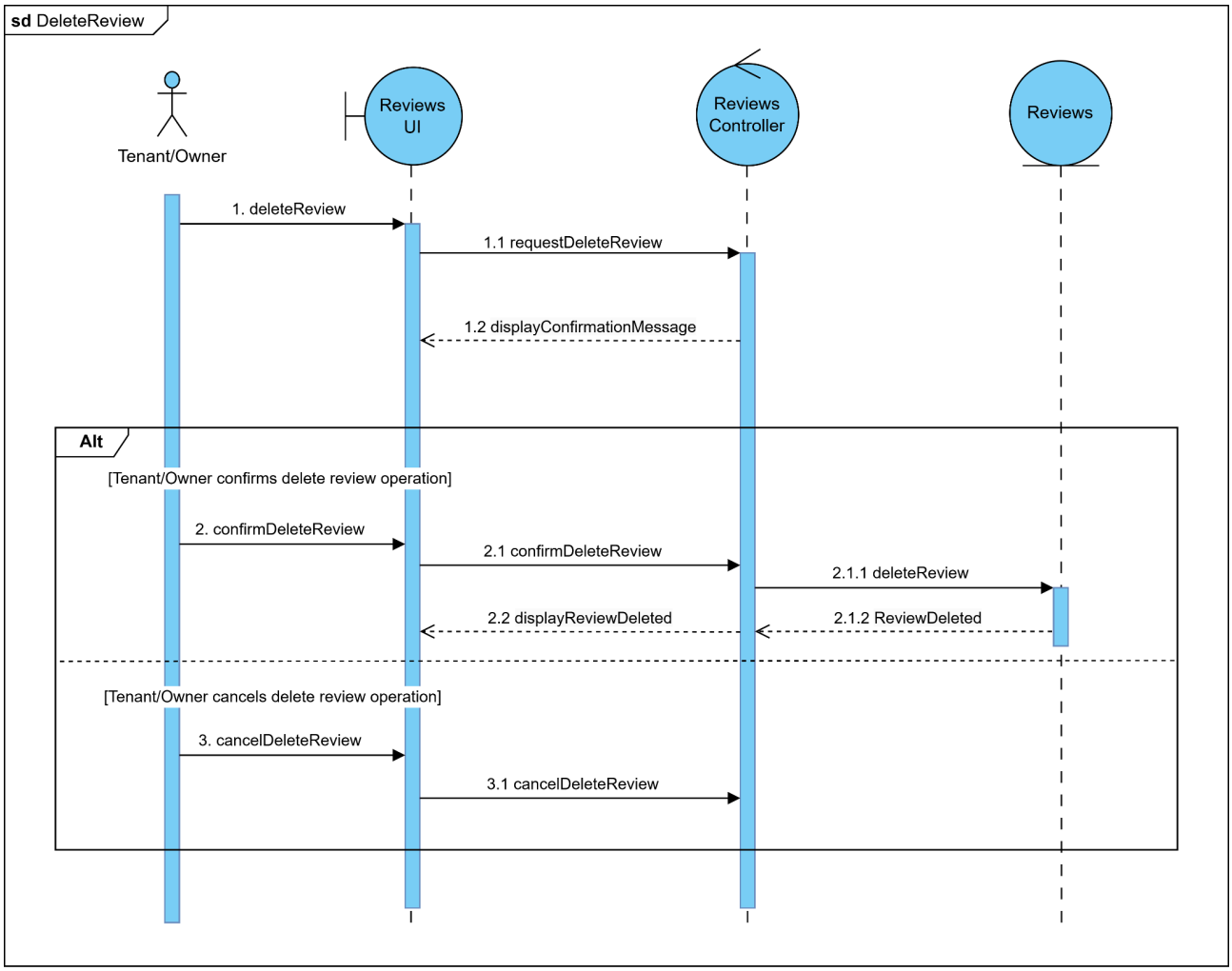
4.3.4.10 FlagReview



4.3.4.11 EditReview



4.3.4.12 DeleteReview



5. Other Nonfunctional Requirements

5.1 Usability Requirements

5.1.1 Responsive User Interface

Users should experience minimal delays (less than 5 seconds) when performing common tasks within the app.

5.1.2 Mobile Responsive

Users must be able to see 100% of the content regardless of what mobile phones they are using.

5.2 Performance Requirements

5.2.1 Data Retrieval Efficiency

The app should retrieve and display data (e.g. nearby schools, hawkers etc) to users within 30 seconds.

5.2.2 Quick App Startup

After rebooting/launching the app, the app should load and all functionalities should be available within 5 seconds.

5.3 Software Quality Attributes

5.3.1 Modular Architecture

The application should have a modular architecture, allowing for easy updates, bug fixes, and enhancements. Each module should be independent to reduce the impact of changes.

5.3.2 Horizontal Scaling

The app should support horizontal scaling, allowing it to handle increased traffic by adding more servers or instances.

5.3.3 Load Balancing

Implement load balancing to distribute user requests evenly across servers, ensuring optimal performance during peak times.

5.4 Business Rules

5.4.1 Encryption

Password to be encrypted before storing inside the database.

5.4.2 Access Control

Actors	Users	Listings	Rentals
Owner	createUser() updateUser() flagUser()	getAllListings() flagListing() createListing() updateListing()	createRental() updateRental() getRental()
Tenant	createUser() updateUser() flagUser()	getAllListings() flagListing()	updateRental() getRental()
Admin	createUser() updateUser() getAllUsers() getFlaggedUsers() banUser()	getAllListings() createListing() updateListing() getFlaggedListings() deleteListing()	createRental() updateRental() getRental() getAllRentals()

Actors	Payment	Request	Reviews
Owner	getPayments()	createRequest() getRequest()	createReview() updateReview() getReview() flagReview()
Tenant	getPayments() createPayment()	getRequest() acceptRequest() denyRequest()	createReview() updateReview() getReview() flagReview()
Admin	getPayments() getAllPayments() createPayment()	createRequest() getRequest() getAllRequests()	createReview() updateReview() getReview()

			getAllReviews() getFlaggedReviews() deleteReview()

Actors	**ChatHistory**
Owner	makeChatHistory() getChatHistory()
Tenant	makeChatHistory() getChatHistory()
Admin	makeChatHistory() getChatHistory() getAllChatHistory()

6. Application Testing

Application testing is to be done using Black Box and White Box testing on selected important control classes and methods that implement complex application logic.

6.1 Black Box Testing

6.1.1 Selected Control Class

The control class selected for testing is the **AuthController**.
The AuthController (Authentication Controller) manages user authentication for the application, including user sign up and user login.

During the user sign up process, the user has to input their email, password and full name. After a successful sign up process, the user's details, as well as the hashed password, will be stored in the database. The user can then login to the application with the same email and password used during the sign up process.

During the user login process, the user has to input their email and password that they used during the sign up process. If the email matches an existing user, and the hashed password input matches the hashed password stored in the database, the user will be authenticated and directed to the home screen.

6.1.2 Equivalence Class Testing

The login and signup process requires discrete values as inputs. As such, Boundary Value Testing will not be applicable

- Login Function
    - **Valid Equivalence Class:** Username and password input values are in correct formats and match an existing user.
    - **Invalid Equivalence Class:** Username and password input values are in incorrect formats or missing or do not match an existing user.

- Sign Up Function
    - **Valid Equivalence Class:** All required fields (Email, Password and Full Name) are correctly filled and email address does not match an existing user.
    - Invalid Equivalence Class: Missing fields (Email, Password and Full Name), invalid email format, or user with the same email address already exists.

6.1.3 Test Cases and Testing Results

6.1.3.1 Login

Input Parameters:

1) Email
2) Password

Test Case Name	Test Input	Expected Output	Actual Output	Test Result
Login-01	**(Valid)** Email: “testuser@gmail.com” **(Valid)** Password: “testpassword”	Login Success	Login Success	Pass
Login-02	**(Valid)** Email: “testuser@gmail.com” **(Invalid)** Password: “wrongpassword”	Login Failed: “Incorrect password”	Login Failed: “Incorrect password”	Pass
Login-03	**(Valid)** Email: “testuser@gmail.com” **(Invalid)** Password: “”	Login Failed: “Password is required”	Login Failed: “Password is required”	Pass
Login-04	**(Invalid)** Email: “fakeuser@gmail.com” **(Valid)** Password: “testpassword”	Login Failed: “Email not registered”	Login Failed: “Email not registered”	Pass
Login-05	**(Invalid)** Email: “” **(Valid)** Password: “testpassword”	Login Failed: “Email is required”	Login Failed: “Email is required”	Pass
Login-06	**(Invalid)** Email: “testuseratgmaildotcom” **(Valid)** Password: “testpassword”	Login Failed: “Email format is invalid”	Login Failed: “Email format is invalid”	Pass

6.1.3.2 Sign Up

Input Parameters:

1) Email
2) Password
3) Full Name

Test Case Name	Test Input	Expected Output	Actual Output	Test Result
Signup-01	**(Valid)** Email: “testuser@gmail.com” **(Valid)** Password: “testpassword” **(Valid)** Full Name: “Test Name”	Sign up Success	Sign up Success	Pass
Signup-02	**(Invalid)** Email: “repeateduser@gmail.com” **(Valid)** Password: “testpassword” **(Valid)** Full Name: “Test Name”	Sign up Failed: “Email already exists.”	Sign up Failed: “Email already exists.”	Pass
Signup-03	**(Invalid)** Email: “” **(Valid)** Password: “testpassword” **(Valid)** Full Name: “Test Name”	Sign up Failed: “Email is required.”	Sign up Failed: “Email is required.”	Pass
Signup-04	**(Invalid)** Email: “testuseratgmaildotcom” **(Valid)** Password: “testpassword” **(Valid)** Full Name: “Test Name”	Sign up Failed: “Email format is invalid.”	Sign up Failed: “Email format is invalid.”	Pass

Signup-05	**(Valid)** Email: “testuser@gmail.com” **(Invalid)** Password: “” **(Valid)** Full Name: “Test Name”	Sign up Failed: “Password is required.”	Sign up Failed: “Password is required.”	Pass
Signup-06	**(Valid)** Email: “testuser@gmail.com” **(Invalid)** Password: “1234” **(Valid)** Full Name: “Test Name”	Sign up Failed: “Password must be at least 6 characters long.”	Sign up Failed: “Password must be at least 6 characters long.”	Pass
Signup-07	**(Valid)** Email: “testuser@gmail.com” **(Valid)** Password: “testpassword” **(Invalid)** Full Name: “”	Sign up Failed: “Full Name is required.”	Sign up Failed: “Full Name is required.”	Pass

6.2 White Box Testing

The 2 methods selected for white box testing will be:

1) SubmitReview
2) MakeTerminationRequest

The **SubmitReview** method allows users to leave reviews for the other user they interacted with during a rental.

This includes 2 scenarios:

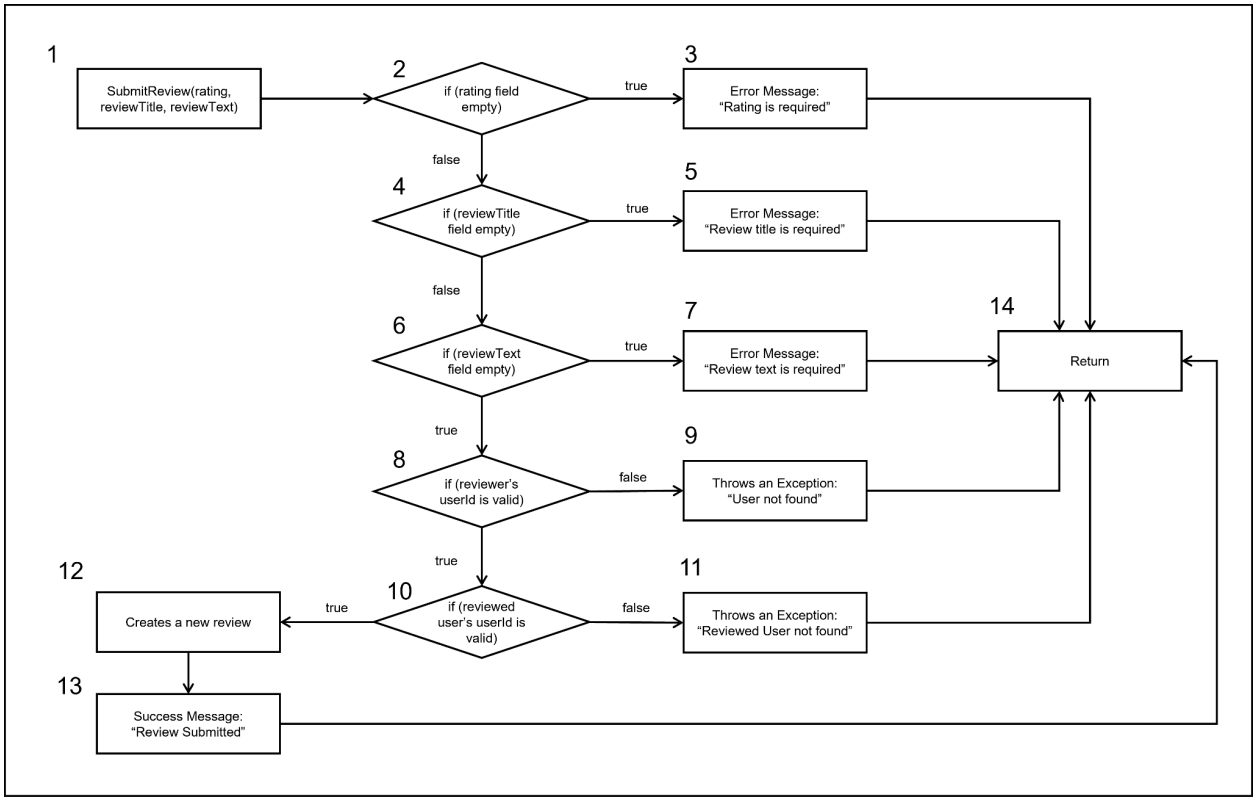
1) Owner leaves review for tenant
2) Tenant leaves review for owner

During the SubmitReview process, the user enters a rating, review title and review text.

The **MakeTerminationRequest** method allows an owner to create a request to terminate a rental contract prematurely, before the actual lease end date specified during the MakeRentalOffer process. During this process, the owner has to enter an amount to refund to the tenant, intended to allow the owner to refund the rental deposit collected earlier. Upon successful creation of the termination request, an automated chat message will be sent to the tenant to approve or deny the request. The termination process will then vary based on the input by the tenant.

6.2.1 SubmitReview

6.2.1.1 Control Flow Graph



6.2.1.2 Cyclomatic Complexity

Cyclomatic Complexity (CC) = | binarydecisionpoint | + 1 = | 5 | + 1 = 6

6.2.1.3 Basis Paths

Basis Path #1 (Baseline): 1,2,4,6,8,10,12,13,14
Basis Path #2: 1,2,3,14
Basis Path #3: 1,2,4,5,14
Basis Path #4: 1,2,4,6,7,14
Basis Path #5: 1,2,4,6,8,9,14
Basis Path #6: 1,2,4,6,8,10,11,14

6.2.1.4 Test Cases and Testing Results

Input Parameters:

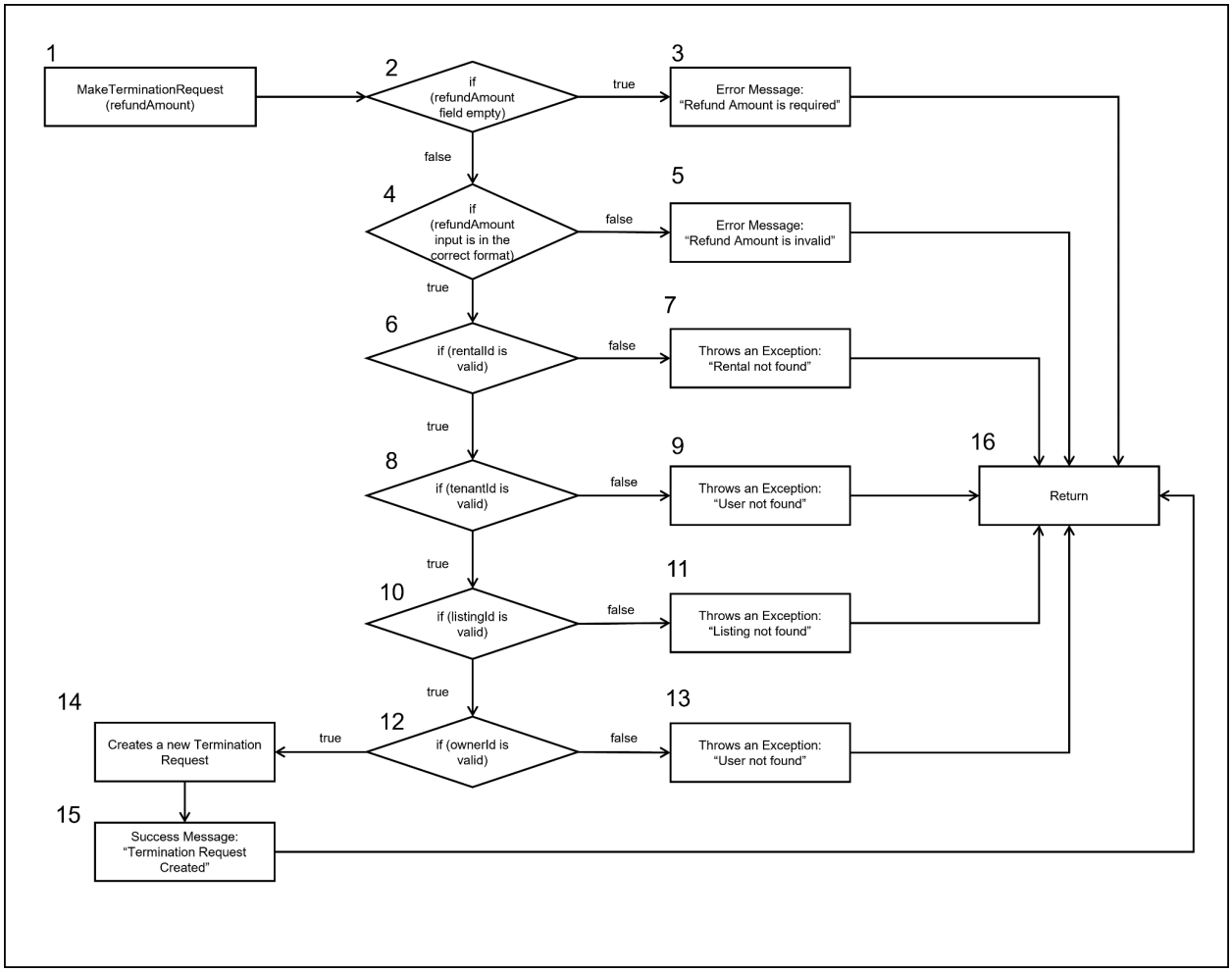
1) rating
2) reviewTitle
3) reviewText
4) Reviewed user's userId (Automatically passed in by referring to the profile page the reviewer navigated from)
5) Reviewer's userId (Automatically passed in by referring to the currently logged in user's userId)

Test Case Name	Test Input	Expected Output	Actual Output	Test Result
Review-01	rating = 5 reviewTitle = "Test Review" reviewText = "This is a test." revieweeId = 1 reviewerId = 2	Success Message: "Review Submitted"	Success Message: "Review Submitted"	Pass
Review-02	rating = reviewTitle = "Test Review" reviewText = "This is a test." revieweeId = 1 reviewerId = 2	Error Message: "Rating is required"	Error Message: "Rating is required"	Pass
Review-03	rating = 5 reviewTitle = "" reviewText = "This is a test." revieweeId = 1 reviewerId = 2	Error Message: "Review title is required"	Error Message: "Review title is required"	Pass
Review-04	rating = 5 reviewTitle = "Test Review" reviewText = "" revieweeId = 1 reviewerId = 2	Error Message: "Review text is required"	Error Message: "Review text is required"	Pass

Review-05	rating = 5 reviewTitle = “Test Review” reviewText = “This is a test.” revieweeId = 1 reviewerId = -1	Error Message: “User not found”	Error Message: “User not found”	Pass
Review-06	rating = 5 reviewTitle = “Test Review” reviewText = “This is a test.” revieweeId = -1 reviewerId = 2	Error Message: “Reviewed user not found”	Error Message: “Reviewed user not found”	Pass

6.2.2 MakeTerminationRequest

6.2.2.1 Control Flow Graph



6.2.2.2 Cyclomatic Complexity

Cyclomatic Complexity (CC) = | binarydecisionpoint | + 1 = | 6 | + 1 = 7

6.2.2.3 Basis Paths

Basis Path #1 (Baseline): 1,2,4,6,8,10,12,14,15,16
Basis Path #2: 1,2,3,16
Basis Path #3: 1,2,4,5,16
Basis Path #4: 1,2,4,6,7,16
Basis Path #5: 1,2,4,6,8,9,16
Basis Path #6: 1,2,4,6,8,10,11,16
Basis Path #7: 1,2,4,6,8,10,12,13,16

6.2.2.4 Test Cases and Testing Results

Input Parameters:

1) refundAmount
2) rentalId (Automatically passed in by referring to the rental page the owner clicked the “Make Termination Request” button on)
3) tenantId (Automatically passed in by referring to tenantId stored by the specific rental retrieved)
4) listingId (Automatically passed in by referring to listingId stored by the specific rental retrieved)
5) ownerId (Automatically passed in by referring to ownerId stored by the specific listing retrieved)

Test Case Name	Test Input	Expected Output	Actual Output	Test Result
request-01	refundAmount = 1000 rentalId = 1 tenantId = 1 listingId = 1 ownerId = 2	Success Message: “Termination Request Created”	Success Message: “Termination Request Created”	Pass
request-02	refundAmount = rentalId = 1 tenantId = 1 listingId = 1 ownerId = 2	Error Message: “Refund Amount is required”	Error Message: “Refund Amount is required”	Pass
request-03	refundAmount = 1000.0.00.0 rentalId = 1 tenantId = 1 listingId = 1 ownerId = 2	Error Message: “Refund Amount is invalid”	Error Message: “Refund Amount is invalid”	Pass
request-04	refundAmount = 1000 rentalId = -1 tenantId = 1 listingId = 1 ownerId = 2	Error Message: “Rental not found”	Error Message: “Rental not found”	Pass

request-05	refundAmount = 1000 rentalId = 1 tenantId = -1 listingId = 1 ownerId = 2	Error Message: “User not found”	Error Message: “User not found”	Pass
request-06	refundAmount = 1000 rentalId = 1 tenantId = 1 listingId = -1 ownerId = 2	Error Message: “Listing not found”	Error Message: “Listing not found”	Pass
request-07	refundAmount = 1000 rentalId = 1 tenantId = 1 listingId = 1 ownerId = -1	Error Message: “User not found”	Error Message: “User not found”	Pass

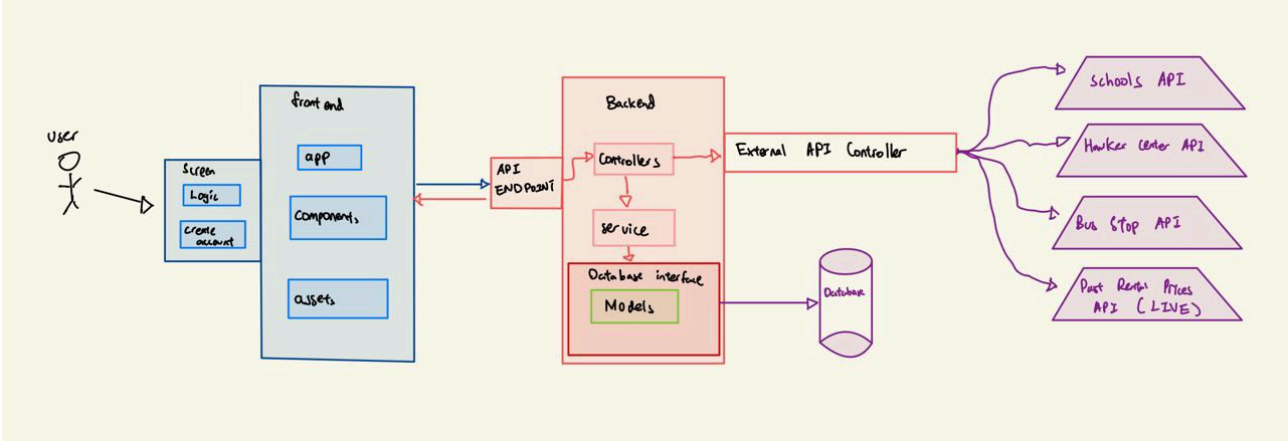
Appendix A: Glossary

Data Dictionary

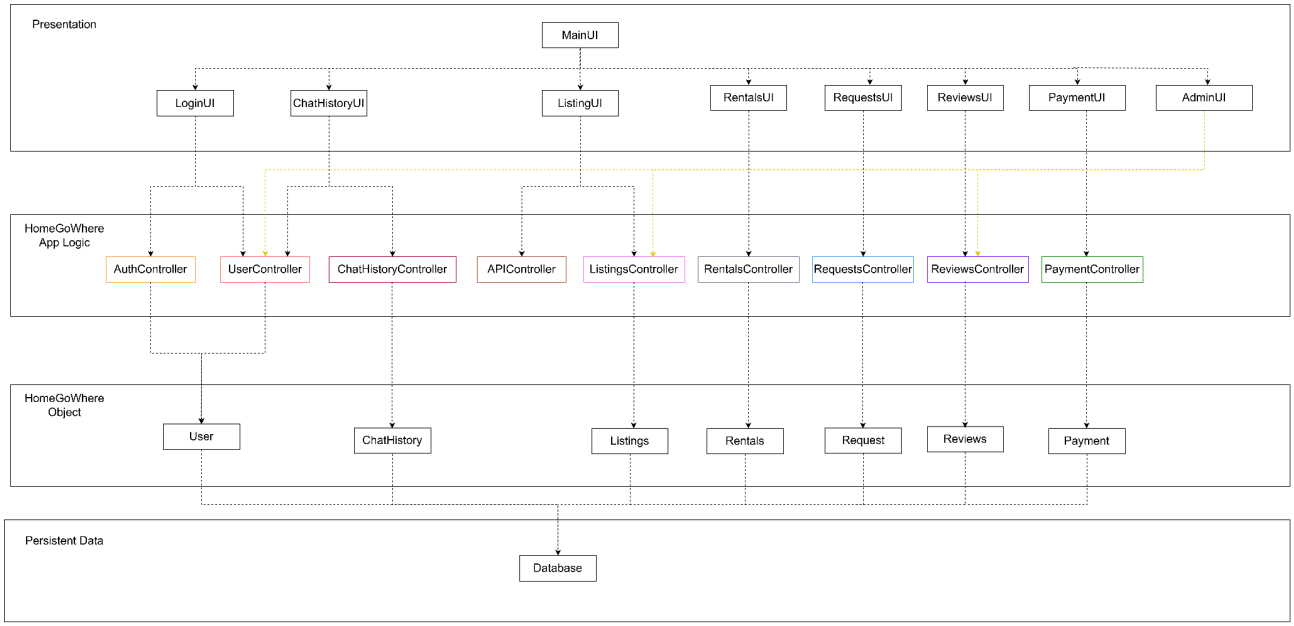
Term	Definition
Account	A registered user's personal profile associated with an application. It may include personal information, contact details, etc.
Admin	A user with special administrative privileges who manages and oversees the operations and features of an application. Admin users have the authority to perform administrative tasks.
Application (App)	A software programme downloaded on users' mobile application that requires internet access in order to operate all the functionality intended by the mobile application. System permissions may be requested in order to operate specific app functionalities.
Ban	An in-app feature that allows Admin to revoke access to the application and its features to a specified User.
Chat	An in-app feature that allows Users to directly message another User for communication purposes.
Dispute	A disagreement in interests between Owner and Tenant, which will be resolved by Admin.
Flag	To bring to the attention of an Admin, for action to be taken by the Admin, typically due to harmful or disruptive content.
Listing	A record submitted by an "Owner" that includes information of a property the "Owner" intends to sell/rent out. Listings are viewable by all "Users" that can initiate a chat with the "Owner" to negotiate and arrange viewings.
Owner	The User who has/intends to list a property on the HomeGoWhere mobile application. Owner does not require a different User account, and may be a Tenant at the same time.
Payment	A financial transaction in which funds are transferred from one party (the payer) to another (the payee) in exchange for the provision of the house, or to fulfill an obligation (rent). Payments can be made using various methods such as credit/debit cards.
Review	Feedback provided by Users about other Users which will help Users to

	make informed decisions on proceeding with transactions.
Search	An in-app feature to filter choices of properties displayed based on select parameters specified in the app.
Tenant	The User who has/intends to rent or buy a property on the HomeGoWhere mobile application. Tenant does not require a different User account, and may be a Owner at the same time.
Termination	A process that can be requested by Users who have ongoing rental/purchasing arrangements, to sever the arrangement, subject to dispute approval by Admin.
User	The person using the HomeGoWhere mobile application after creating a HomeGoWhere account. A User may also be an Admin.

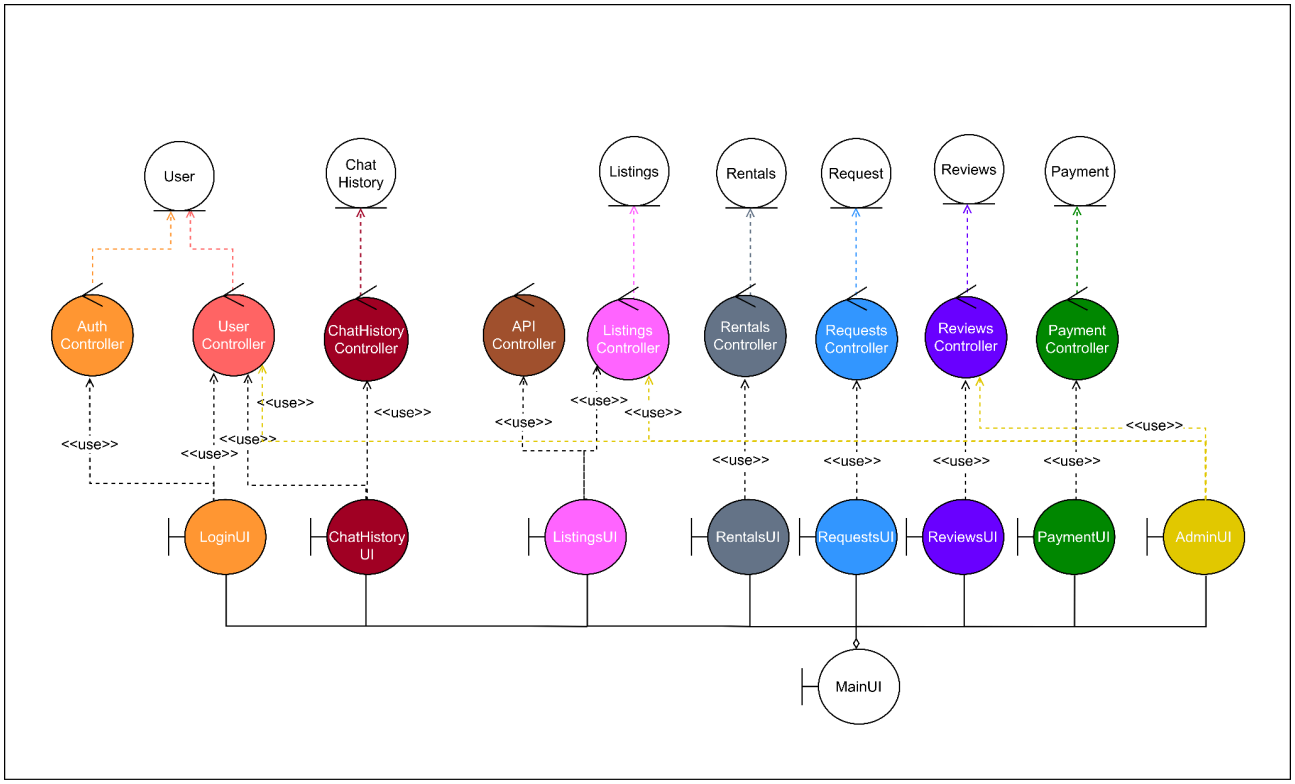
Appendix B: Analysis Models



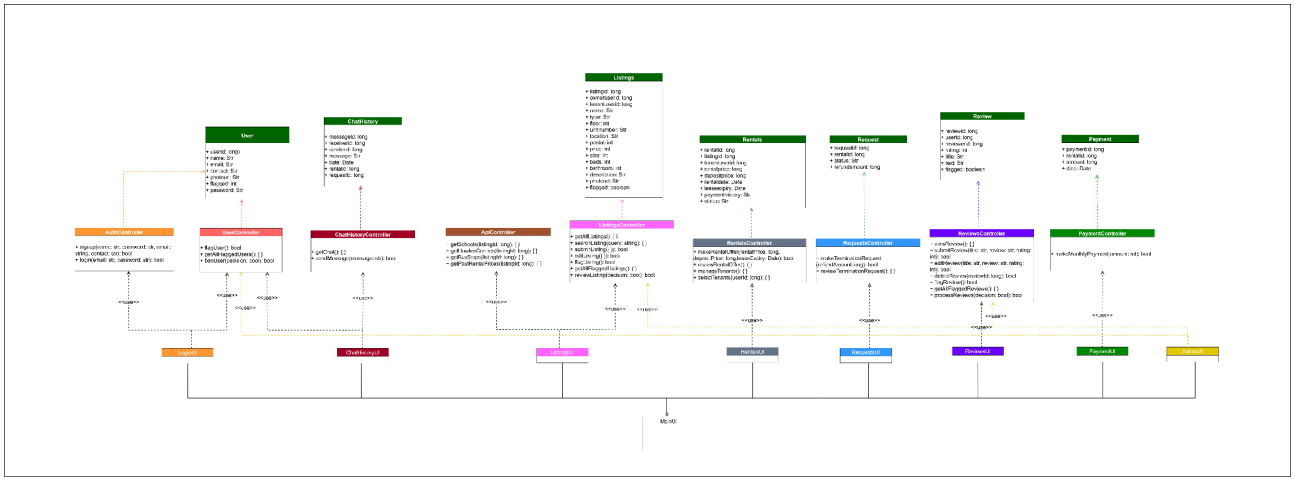
System Overview Diagram



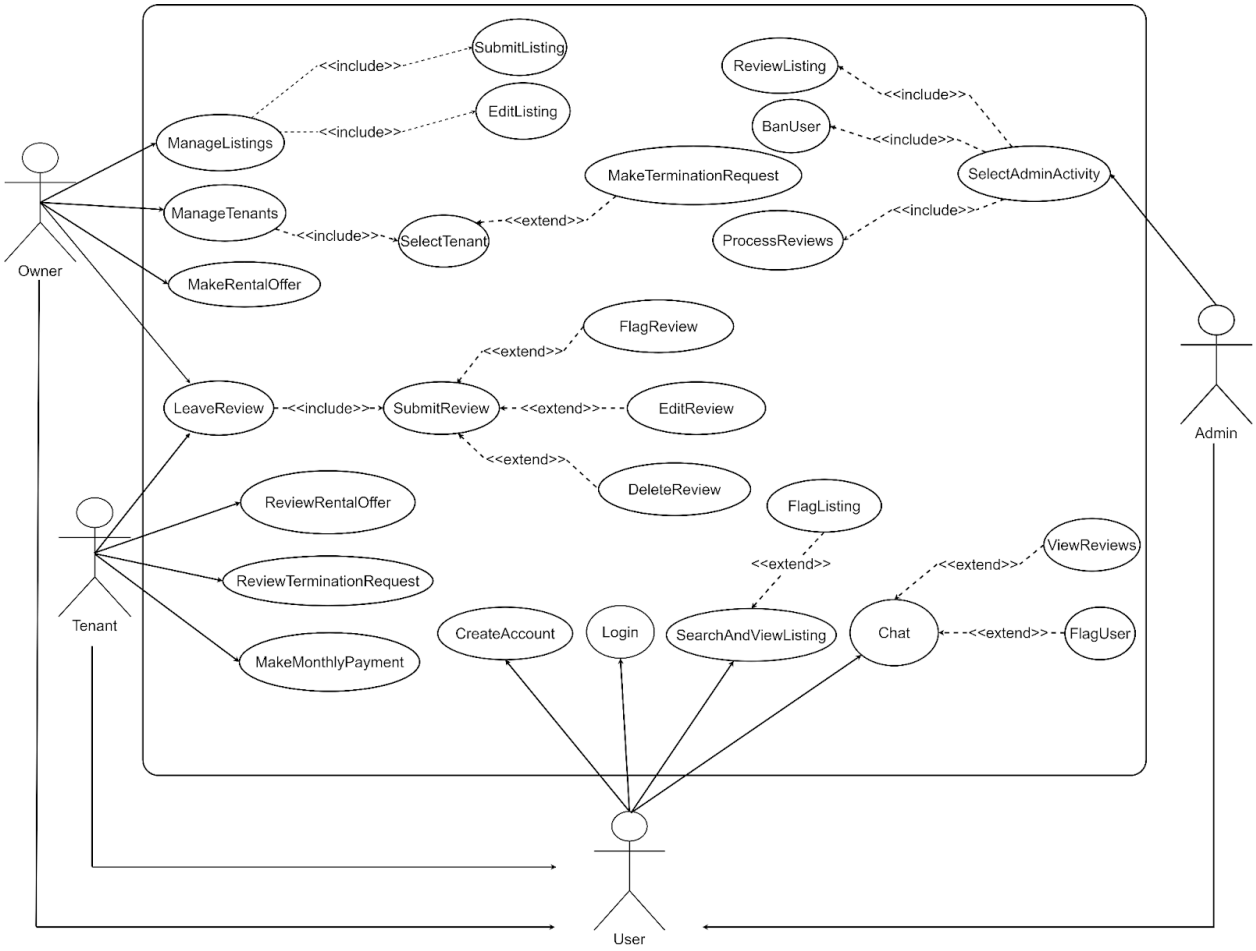
System Architecture Diagram



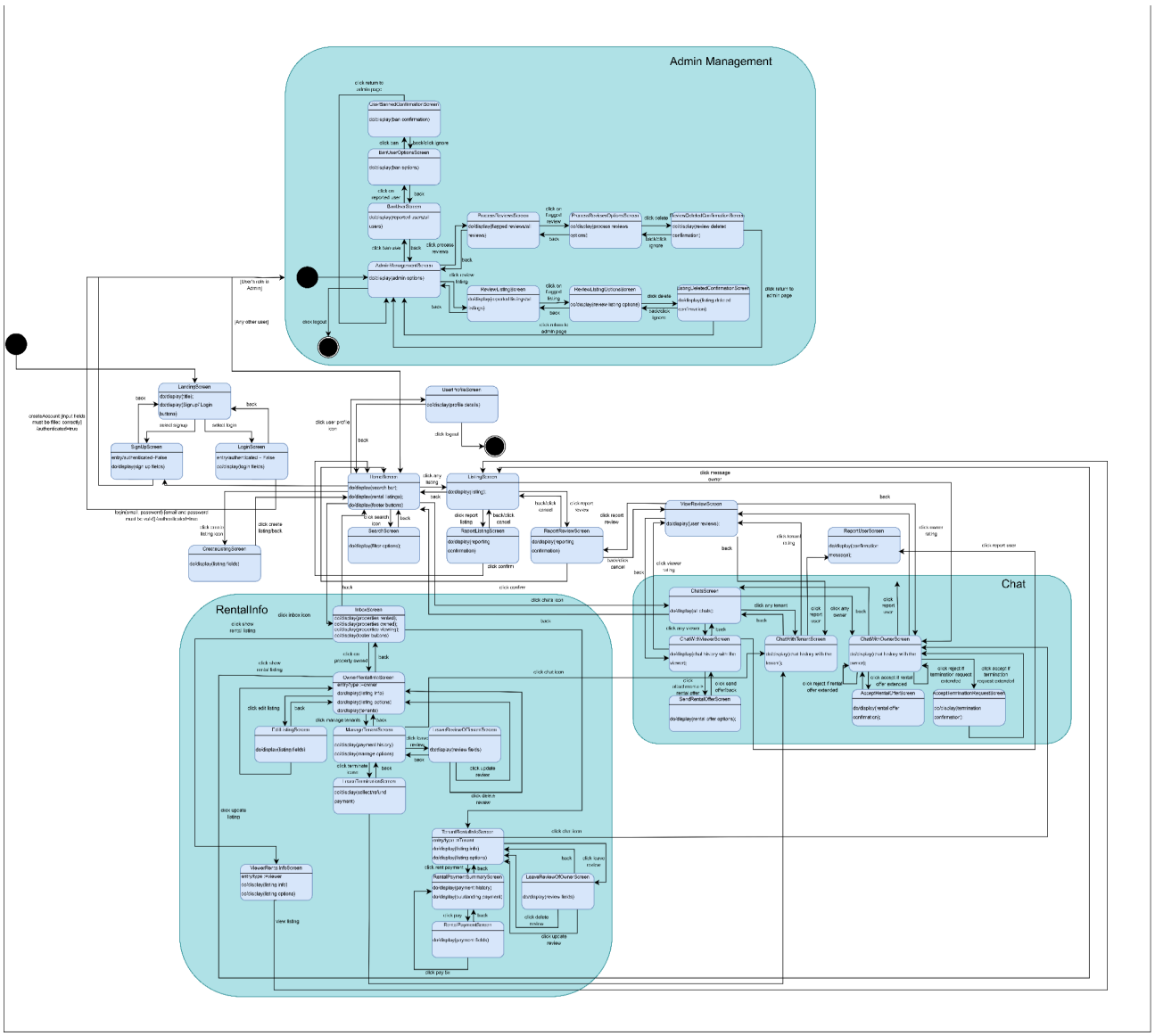
Class Diagram



Key Classes Diagram



Use Case Diagram



Dialog Map