# Home Rentals

**Final Year Project**

**Session 2021-2025**

A project submitted in partial fulfillment of the degree of

BS in Gamming and Multimedia

Department of Computer Science

Faculty of Computer Science & Information Technology

The Superior University, Lahore

Fall 2025

| | |
|---|---|
| Type (Nature of project) | [] **D**evelopment [] **R**esearch [✓ ] **R**&**D** |
| Area of specialization | Mobile Application |
| FYP ID | FYP-BGMM-F24-014 |

**Project Group Members**

| Sr.# | Reg. # | Student Name | Email ID | *Signature |
|---|---|---|---|---|
| (i) | SU92-BSGMM-F23-003 | ABDUL HASEEB | SU92-BSGMM-F23-003@SUPERIOR.EDU.PK | |
| (ii) | SU92-BSCSM-F23-349 | AYESHA ARIF | SU92-BSCSM-F23-349@SUPERIOR.EDU.PK | |

*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to work of others

## Plagiarism Free Certificate

This is to certify that, I **Abdul Haseeb** S/D of **Muhammad Qasim Khan**, group leader of FYP under registration no **FYP-BGMM-F24-014** at Computer Science Department, The Superior University, Lahore. I declare that my FYP report is checked by my supervisor.

Date: Name of Group Leader: **Abdul Haseeb** Signature: _____

Name of Supervisor: Ms. Misbah Javed

Designation: Junior Lecturer

Signature: _____

HoD: Dr. Muhammad Azam

Signature: _____

# Project Report

## [Home Rentals]

## Change Record

| Author(s) | Version | Date | Notes | Supervisor's Signature |
|---|---|---|---|---|
| Abdul Haseeb | 1.0 | 26-10-2024 | In this Project we are going to make Home Rentals App for online Rent/Buy Properties to the Landlord/Renters. | |
| Ayesha Arif | 1.2 | 19-11-2024 | Literature reviews with references | |
| Ayesha Arif | 1.3 | 19-12-2024 | Improved features and scope based on faculty suggestions, including user experience and functionality refinements. | |
| Abdul Haseeb | 1.4 | 19-03-2025 | Defined project phases, milestones, and timeline for development. | |
| Abdul Haseeb | 1.5 | 19-02-2025 | Chat option is compulsory for buyer and renter contact each other. | |

# APPROVAL

**PROJECT SUPERVISOR**

Comments: _____

_____

Name: _____

Date: _____ Signature: _____

**PROJECT MANAGER**

Comments: _____

_____

Date: _____ Signature: _____

**HEAD OF THE DEPARTMENT**

Comments: _____

_____

Date: _____ Signature: _____

# Dedication

This project becomes a dedicated offering to God Almighty who served as my creator and strong foundation and source of inspiration for wisdom and understanding and knowledge. From the beginning of this program God provided essential support while we achieved everything through His guidance. This work is dedicated first to my creator God Almighty and then to my friend Nauman Bahadur for continuous encouragement and to my parents who have suffered every burden from this expedition. Thank you. The depth of my affection toward all of you extends beyond measurement. God bless you.

# Acknowledgements

Department of BS Computer Science at University of **"Superior"** received our deepest gratitude for the support provided by "**Ms. Misbah Javed**." The project received continuous guidance from her throughout its entire duration. The completion of our project was possible because of the unspoken backing provided by our friends and family members.

# Executive Summary

Mobile application leverages Flutter framework technology to establish an effective solution that links property owners with prospective renters. This application features a solution for rental market issues by enabling property owners to outline listings with mandatory information and add location data and rental amounts and property images. The AI chatbot system provides owners assistance in property proposals and question response functions. The search functionality within the rental application grants users modern filtering tools to locate homes matching their precise property requirements including price and location. The application depends on Firebase to handle data processing alongside user authentication and provides real-time functionality and secure environments to users. The solution provides an updated rental processing system through its intuitive interface which caters to all property owners and renters.

# Table of Contents

Dedication ..... v
Acknowledgements ..... vi
Executive Summary ..... vii
Table of Contents ..... viii
List of Figures ..... xii
List of Tables ..... xiii
Chapter 1 ..... 1
Introduction ..... 1
1.1. Background ..... 2
1.2. Motivations and Challenges ..... 3
1.3. Goals and Objectives ..... 3
1.4. Literature Review/Existing Solutions ..... 4
1.5. Gap Analysis ..... 7
1.6. Proposed Solution ..... 7
1.7. Project Plan ..... 8
1.7.1. Work Breakdown Structure ..... 8
1.7.2. Roles & Responsibility Matrix ..... 9
1.7.3. Gantt Chart ..... 9
1.8. Report Outline ..... 10
1.9. Empathy Map ..... 10
Chapter 2 ..... 11
Software Requirement Specifications ..... 11
2.1. Introduction ..... 12
2.1.1. Purpose ..... 12
2.1.2. Document Conventions ..... 13
2.1.3. Intended Audience and Reading Suggestions ..... 13
2.1.4. Product Scope ..... 14

2.1.5. References ..... 15
2.2. Overall Description ..... 15
2.2.1. Product Perspective ..... 15
2.2.2. Product Functions ..... 15
2.2.5. Operating Environment ..... 16
2.2.6. Design and Implementation Constraints ..... 16
2.2.9. Assumptions and Dependencies ..... 17
2.3. External Interface Requirements ..... 17
2.3.1. User Interfaces ..... 17
2.3.2. Hardware Interfaces ..... 18
2.3.3. Software Interfaces ..... 18
2.3.4. Communications Interfaces ..... 18
2.4. System Features ..... 18
2.4.1. System Feature 1 ..... 19
2.4.1.1. Description and Priority ..... 19
2.4.1.2. Stimulus/Response Sequences ..... 19
2.4.1.3. Functional Requirements ..... 19
2.4.2. System Feature 2 ..... 21
2.4.2.1. Description and Priority ..... 21
2.4.2.2. Stimulus/Response Sequences ..... 21
2.4.2.3. Functional Requirements ..... 21
2.4.3. System Feature 3 ..... 22
2.4.3.1. Description and Priority ..... 22
2.4.3.2. Stimulus/Response Sequences ..... 23
2.4.3.3. Functional Requirements ..... 23
2.5. Other Nonfunctional Requirements ..... 25
2.5.1. Performance Requirements ..... 25
2.5.2. Safety Requirements ..... 25
2.5.3. Security Requirements ..... 26

2.5.4. Software Quality Attributes ..... 26
Chapter 3 ..... 27
Use Case Analysis ..... 27
3.1. Use Case Model ..... 28
3.2. Use Case Descriptions ..... 29
Chapter 4 ..... 36
System Design ..... 36
4.1. Architecture Diagram ..... 37
4.2. Domain Model ..... 38
4.3. Entity Relationship Diagram with data dictionary ..... 39
4.4. Class Diagram ..... 40
4.5. Sequence / Collaboration Diagram ..... 41
4.6. Operation contracts ..... 41
4.7. Activity Diagram ..... 42
4.8. State Transition Diagram ..... 43
4.9. Component Diagram ..... 44
4.10. Deployment Diagram ..... 45
4.11. Data Flow diagram ..... 46
Chapter 5 ..... 47
Implementation ..... 47
5.1. Important Flow Control/Pseudo codes ..... 48
5.2. Components, Libraries, Web Services and stubs ..... 50
5.3. Deployment Environment ..... 51
5.4. Tools and Techniques ..... 51
5.5. Best Practices / Coding Standards ..... 52
5.6. Version Control ..... 53
Chapter 6 ..... 54
Testing and Evaluation ..... 54
6.1. Use Case Testing ..... 55

6.2. Equivalence partitioning ..... 60
6.3. Boundary value analysis ..... 61
6.4. Data flow testing ..... 62
6.5. Unit testing ..... 63
6.6. Integration testing ..... 64
6.7. Performance testing ..... 65
6.8. Stress Testing ..... 66
Chapter 7 ..... 67
Summary, Conclusion and Future Enhancements ..... 67
7.1. Project Summary ..... 68
7.2. Achievements and Improvements ..... 68
7.3. Critical Review ..... 69
7.4. Lessons Learnt ..... 69
7.5. Future Enhancements/Recommendations ..... 70
Appendices ..... 71
Appendix A: User Manual ..... 72
Appendix B: Administrator Manual ..... 75
Reference and Bibliography ..... 78
Index ..... 82

# List of Figures

Figure 1.1 Work Breakdown Structure ..... 8
Figure 1.2 Gantt Chart ..... 9
Figure 1.3 Empthy Map ..... 10
Figure 3.1 Use Case Diagram ..... 28
Figure 4.1 System Architecture Diagram ..... 37
Figure 4.2 Domain Model ..... 38
Figure 4.3 Entity Relationship Diagram (ERD) ..... 39
Figure 4.4 Class Diagram ..... 40
Figure 4.5/4.6 Sequence Diagram/Opreation Contracts ..... 41
Figure 4.7 Activity Diagram ..... 42
Figure 4.8 State Transition Diagram ..... 43
Figure 4.9 Component Diagram ..... 42
Figure 4.10 Deployment Diagram ..... 44
Figure 4.12 Data Flow Diagram ..... 45
Figure 5.1 Implementation Flowchart ..... 48

# List of Tables

Table 1.1 Roles & Responsibility Matrix ..... 9
Table 2.1 Functional Requirements of User Registration ..... 20
Table 2.2 Functional Requirements of Property Listing ..... 22
Table 2.3 Functional Requirements of Search & Filter ..... 24
Table 3.1 Use Case Descriptions ..... 29
Table 5.1 Tools and Libraries Used ..... 50
Table 5.2 Coding Standards and Practices ..... 53
Table 6.1 Use Case Testing Results ..... 55
Table 6.2 Equivalence Partitioning Test Cases ..... 60
Table 6.3 Boundary Value Analysis Test Cases ..... 61
Table 6.4 Data Flow Testing Scenarios ..... 62
Table 6.5 Unit Testing Results ..... 62
Table 6.6 Integration Testing Results ..... 63
Table 6.7 Performance Testing Results ..... 64
Table 6.8 Stress Testing Results ..... 65

# Chapter 1

## Introduction

# **Chapter 1:** Introduction

Application operates through Flutter and serves to improve rental practices between owners and renters. The rental market experiences problems because renters lack sufficient accurate property details and property owners struggle to reach potential tenants which extends their units' vacancy period. The app solves these issues by giving property owners a straightforward system to sign up for registration and make property listings containing essential information about location, rent costs and photographic content. Users can access an intelligent chatbot system which helps property owners by both managing inquiries and providing specific property suggestions. The managed search tool enables renters to narrow down housing options through a location-based approach, a price range selection, and feature-specific search criteria for a quick property search outcome. The app utilizes Firebase real-time management to deliver simplified property rental solutions that serve the needs of landlords and renters.

## 1.1. Background

Running a property or finding a rental unit through traditional methods brings multiple obstacles between owners and renters. Property owners experience difficulties in effective advertising their properties yet renters cannot easily find fitting listings because the available information is restricted and limited to physical search parameters. The continuous development of mobile technology requires innovative time-based solutions. The project solves rental market difficulties through Flutter and Firebase platform development that delivers a simple user-focused platform merging property listing features for owners and rental discovery features for prospective renters. The AI-powered chatbot aids property owners to resolve inquiries and provide customized property suggestions to their queries.

## 1.2. Motivations and Challenges

The mobile application arose from a basic need to optimize property rental processes because conventional systems produce dissatisfaction among property owners and renters through complex procedures and unproductive processes. Through its efficient digital platform, the app allows faster property search and listing and operates with an AI-powered chatbot which automates owner queries as well as presents customized property suggestions. The application faced two primary difficulties in designing simple user interfaces for various users and making the AI model produce correct outputs. The system needed to ensure data security and privacy as well as maintain instant data updates between different users to create a clear experience.

## 1.3. Goals and Objectives

The main purpose of this mobile application involves developing an instinctive platform through which property owners link with renters to optimize rental transactions. The application enables property owners for registration and listing of their properties with key information like location data alongside rental rate and photograph details and provides an AI chatbot system that responds to inquiries and suggests customized property solutions. Through its rental application renters can uncover powerful search features that help them find ideal properties by their desired criteria like price range and location. By using Firebase middleware, the application can deliver real-time updates which include secure data management to enhance overall user satisfaction. The primary objective is to transform the rental process which should become user-friendly yet efficient for anyone active in this market.

## 1.4. Literature Review/Existing Solutions

The main purpose of this mobile application involves developing an instinctive platform through which property owners link with renters to optimize rental transactions. The application enables property owners for registration and listing of their properties with key information like location data alongside rental rate and photograph details and provides an AI chatbot system that responds to inquiries and suggests customized property solutions. Through its rental application renters can uncover powerful search features that help them find ideal properties by their desired criteria like price range and location. By using Firebase middleware, the application can deliver real-time updates which include secure data management to enhance overall user satisfaction. The primary objective is to transform the rental process which should become user-friendly yet efficient for anyone active in this market.

Modern technological advancements have dramatically reinvented real estate industry operations regarding property listings searches and rentals. Multiple internet platforms now connect property owners with renters through their services. Major real estate platforms serving the population include Housing Anywhere, Agoda.com, Booking.com, Zameen.com, OLX.com, Speed Home, Trulia, and Rent4me. Digital solutions have changed traditional rental practices into better operating systems that offer convenience with lower physical barriers. [1][2].

Users can access the easy-to-use interfaces of **Housing Anywhere** and **Speed Home** to see property listings accompanied by comprehensive descriptions about price along with location and facilities. Such platforms as **Zameen.com** and **OLX.com** now make it easy for renters to search properties through list filters according to their desired criteria including location and price. [3]. The development of new real estate processes has occurred despite current obstacles persisting. Multiple problems on real estate portals include inaccurate and dated property details together with commissions from third-party brokers in addition to hidden fees and hard-to-understand legal terms and basic user support services which reduce trust between parties [4].

The current rental platforms face a major drawback because they operate through intermediaries which lead to that match the monthly rent of properties thus creating financial struggles for tenants[5]. Users of traditional platforms are threatened by privacy breaches because these platforms do not implement adequate security measures to safeguard personal information [6].

The rental market benefits greatly from contemporary technology integration because of artificial intelligence (AI) together with machine learning. Chatbots operated by AI technology now improve customer service through their instant responses as well as their question answering and personalized recommendation services which use user specific requirements and historical data points [7]. Digital transactions receive higher efficiency marks along with higher customer satisfaction through these technological implementations. [8]. According to Brown and Green (2022), AI-powered chatbots in estate applications reduce both communication timing and boost interaction between owners and renters.[9]. Available research by Ahmed and Zhao (2023) shows Firebase and similar scalable database solutions play a crucial role in management and accessibility of extensive real estate data with real-time user update capabilities.[10].

Williams and Oliver (2023) demonstrate how augmented reality technology delivers virtual experiences through research that purposes its application for improving rental platform user experiences. Such technological features help users make more informed choices while lowering the need for actual building inspections to enhance satisfaction levels and convenience in the rental process. [11]. Despite these advancements, the Pakistani rental market still faces specific shortcomings. Users experience reduced trust and engagement rates because both OLX.com and Zameen.com feature incomplete facilities including AI property recommendation systems and real-time database updates.[3][5]. Local platforms have limited access for wide user groups because they often use difficult interfaces without multiple language options [12]. The project targets particular market gaps through mobile application development with Flutter framework integration of AI-powered chatbot functionality and advanced search capabilities and real-time updates through Firebase platform. This mobile application aims to deliver a combined experience of streamlined operations with enhanced security and user-friendly features that caters to Pakistani market characteristics. Emerging Technologies in Rental Applications The rental procedure became significantly more efficient through implementing advanced technologies particularly artificial intelligence (AI). According to Brown and Green (2022) the effectiveness of AI-based chatbots entails quick responses that minimize customer care wait times. The platform's reaction allows personalized assistance and recommendation services combined with process guidance in order to improve user interaction during renting activities

[13]. Similarly, Ahmed and Zhao (2023) note cloud-based database management systems' critical role, such as Firebase, in real-time updates, data security, and scalability. These technologies establish online listing's reliability and consistency which creates better user experiences through time-sensitive accurate information available without limitations. [14]. The research by Williams and Oliver (2023) introduces augmented reality (AR) as a modern method to enhance user satisfaction when choosing properties. AR technology enables distant property inspection through virtual tours which enables better decision-making by prospects without requiring actual site visits and spares them from the burden of physical inspection like in pandemic scenarios [15].

**Current Issues and Gaps in Pakistani Market**

Despite technological advancements, the Pakistani rental market still faces critical challenges. The platforms Zameen.com and OLX.com used in Pakistan provide substandard functionality because they do not contain AI recommendations or real-time data synchronization features that enhance their operational efficiency. Outdated listings and complex interfaces on user interfaces stand as obstacles which prevent users who lack technical expertise from accessing rental listings. [16]. As per Mrisha and Xixiang (2024) the use of third-party brokers for rent transactions creates financial disadvantages by requiring brokers to charge commission fees equivalent to monthly rental rates. Financial burdens based on this practice become a big challenge for renters learning about market practices for the first time specifically as they search for their new home.[17].

According to Sharma (2023) the current platforms do not employ sufficient security standards for protecting user data properly. An insufficient security system produces potential data breaches that both damages user privacy and generates trust-related doubts among platform users[17].

**Opportunities for Improvement**

Multiple possibilities exist to improve digital rental platforms in light of the described challenges. The four main areas of digital rental platform enhancement include implementing AI-powered chatbots with enhanced security protocols and user-friendly interface design together with real-time database synchronization. According to Singh et al. (2024) the relationship between information technology progress and better consumer conduct stands strong because these advancements create simpler processes and enhanced security along with trust-building

features. [11]. Online convenience provides organizations with a valuable opportunity for web user personalization and targeted recommendations. Web personalization leads to improved user engagement with better satisfaction rates which produces positive changes in customer digital platform behavior according to Suvattanadilok (2020).[12].

## 1.5. Gap Analysis

Pakistan's rental market presents numerous possibilities for betterment yet it lacks essential aspects for improvement. The property listing platforms OLX and Zameen.com provide listing services but do not supply AI-generated property descriptions that would potentially enhance rental property owner marketing success. Users struggle to access modern information about rental availability because numerous listings remain outdated. Users unfamiliar with complex tech platforms face confusion because these systems include third-party vendors and brokers who receive payment which results in high costs that equal the property rent. This leads to financial problems for users. The presence of third-party vendors causes regular tenants to become victims of fraud and owners encounter multiple challenges throughout the signing process due to unclear terms and unfamiliar legal boundaries and unsupportive pricing structures and difficult rent rate comparisons and property selection. These platforms need to implement an AI chatbot service between owners and renters for querying about properties. The safety of personal data remains at risk because several electronic platforms fail to establish proper safeguards. The project develops a basic user-friendly mobile application to enhance rental operations between owners and renters.

## 1.6. Proposed Solution

The project presents a mobile application which functions as a user-friendly platform for connecting property owners with potential renters in legitimate transactions. The application permits property owners to build listings containing essential information together with location details and rental costs and image uploads while an AI-based chat function helps owners deal with inquiries by suggesting personalized property choices. The mobile application presents advantageous search capabilities for users seeking properties based on price and location and property attributes. The app enables real-time system updates together with safe management of user data which simplifies and secures the entire rental journey for every participant.

## 1.7. Project Plan

A mobile application development plan for the rental market contains essential steps and procedures for development. We will initiate our project by performing in-depth research on all necessary user requirements targeted at proprietors and tenants. User-friendly interfaces development and planning of the app layout will be addressed next during the design phase. The subsequent phase will be development while using Flutter and AI features to integrate chatbot assists in the application. The testing phase follows development to verify complete functionality of the app while confirming usability, fixing bugs and maintaining top priority encryption with end-to-end security. We will perform a launch of the application followed by user feedback collection for future betterment.

### 1.7.1. Work Breakdown Structure

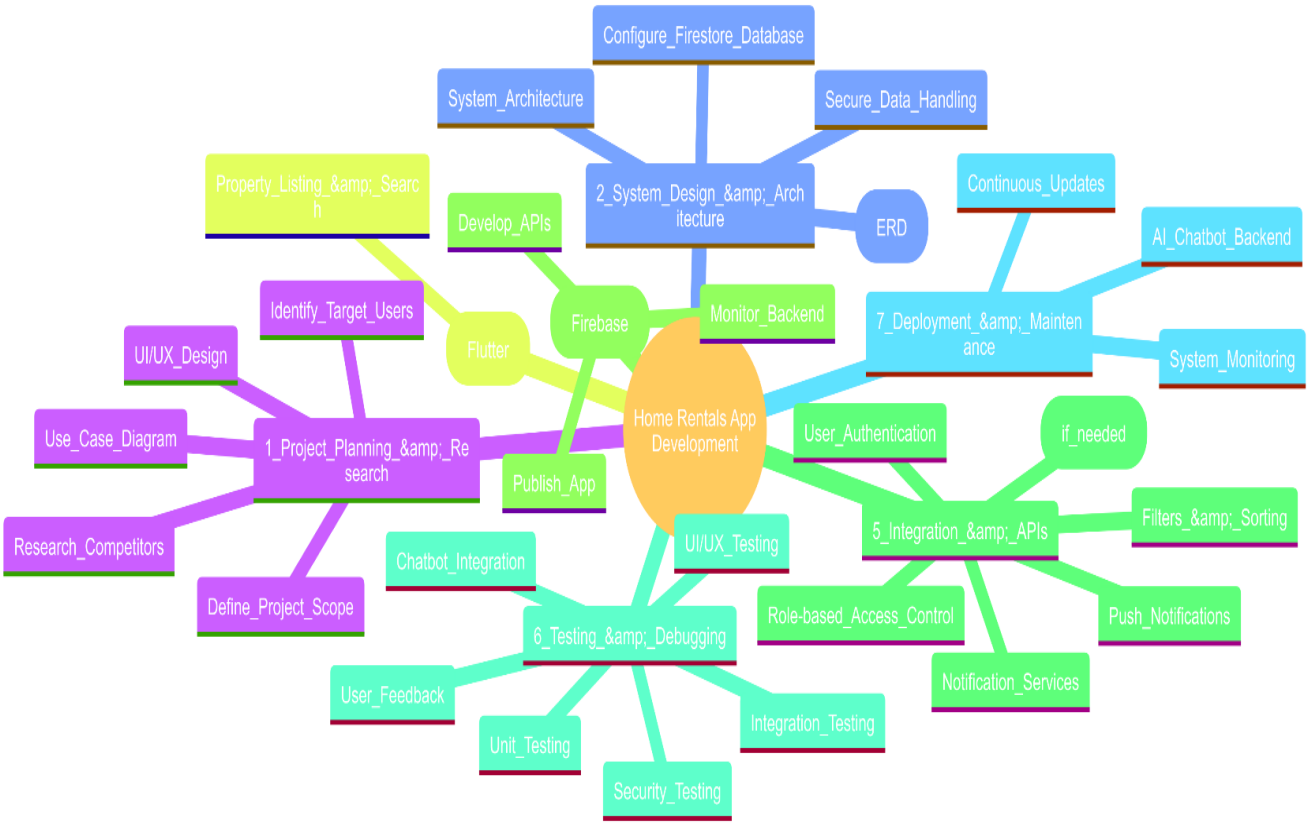



**Figure 1.1 Work Breakdown Structure**

### 1.7.2. Roles & Responsibility Matrix

| Task/Activity | Abdul Haseeb (Developer) | Ayesha Arif (Designer) |
|---|---|---|
| 1. Project Initialization | Responsible, Accountable | |
| 1.1 Define Project Scope | Responsible, Accountable | |
| 1.2 Gather Requirements | Responsible, Accountable | |
| 2. Research and Planning | Responsible, Accountable | Responsible, Accountable |
| 2.1 Conduct Market Analysis | Responsible, Accountable | Responsible, Accountable |
| 2.2 Finalize App Features | Responsible, Accountable | |
| 3. Design Phase | | Responsible, Accountable |
| 3.1 User Interface Design | | Responsible, Accountable |
| 4. Development Phase | | Responsible, Accountable |
| 4.1 Frontend Development | | Responsible, Accountable |
| 4.2 Backend Development | Responsible, Accountable | |
| 5. Testing Phase | Responsible, Accountable | |
| 6. Deployment Phase | Responsible, Accountable | |
| 7. Documentation | | Responsible, Accountable |
| 7.1 Write Project Documentation | | Responsible, Accountable |

**Table 1.1 Roles & Responsibility Matrix**

### 1.7.3. Gantt Chart

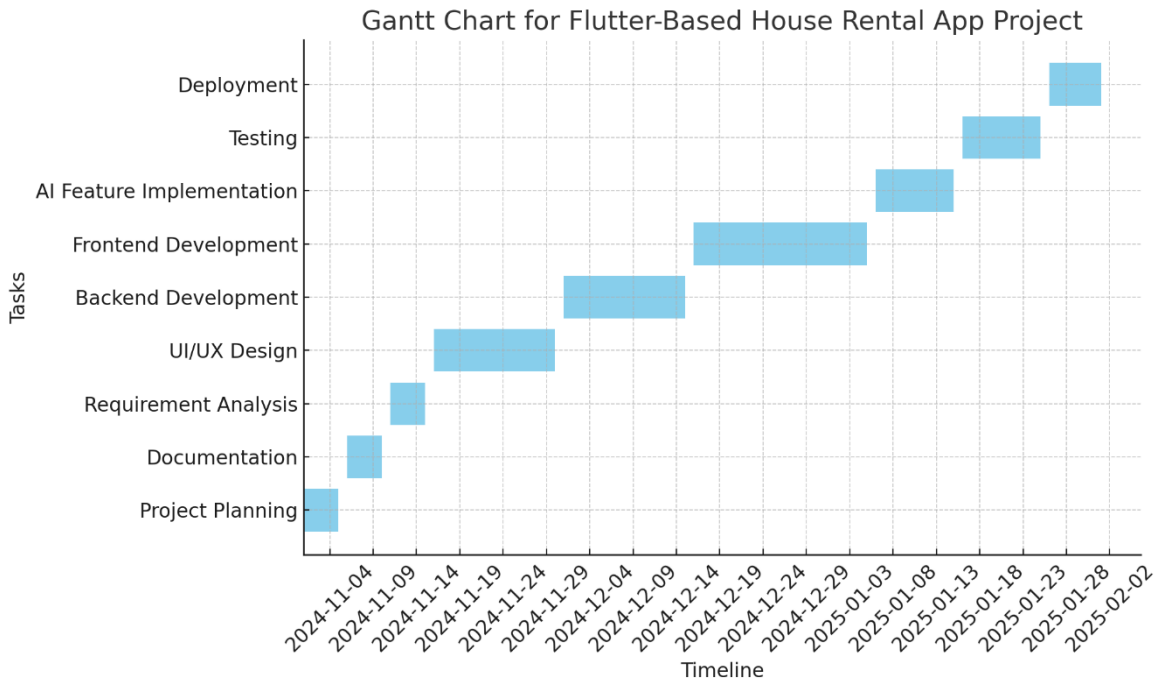



**Figure 1.2 Gant Chart**

## 1.8. Report Outline

1. Introduction
2. Project Description
3. Design of system
4. Implement
5. Testing
6. Conclusion
7. References
8. Appendices

## 1.9. Empathy Map

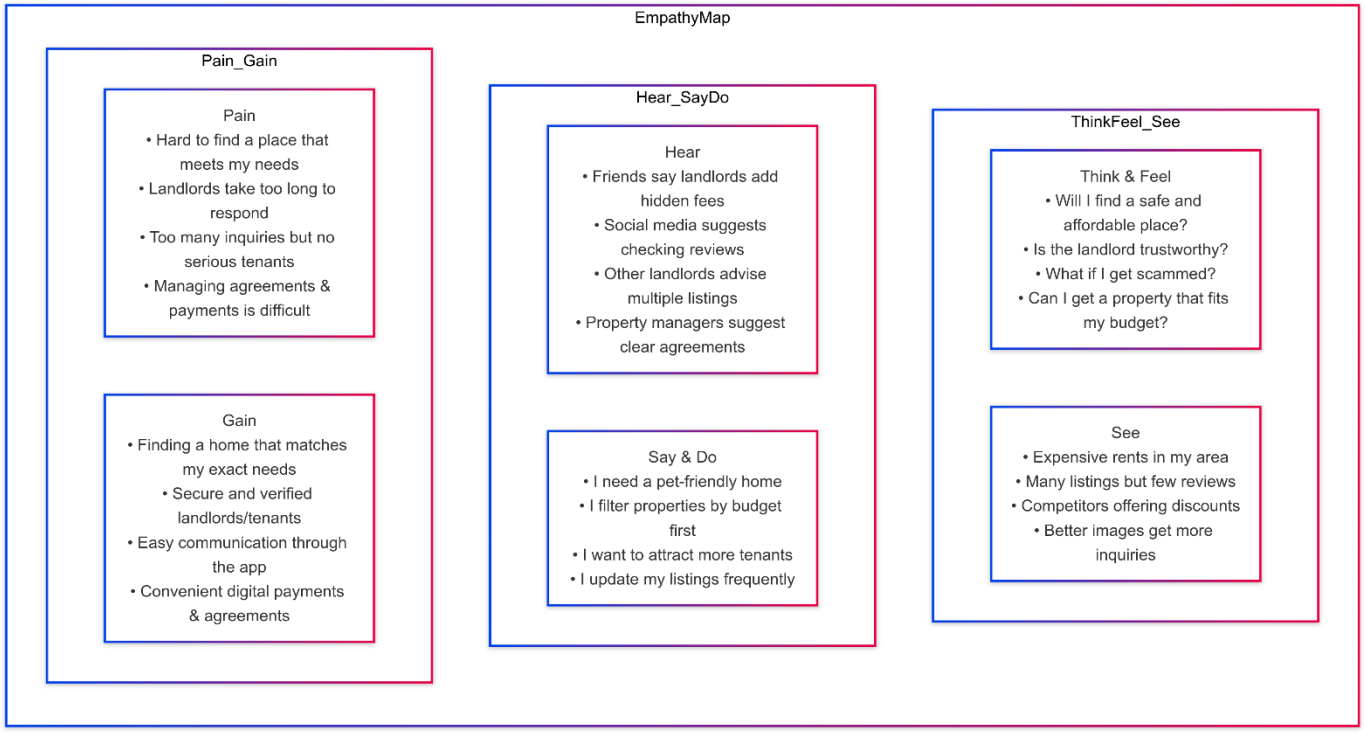



**Figure 1.3 Empathy Map**

# Chapter 2

# Software Requirement Specifications

# Chapter 2: Software Requirement Specifications

## 2.1. Introduction

The application operates through Flutter and serves to improve rental practices between owners and renters. The rental market experiences problems because renters lack sufficient accurate property details and property owners struggle to reach potential tenants which extends their units' vacancy period. The app solves these issues by giving property owners a straightforward system to sign up for registration and make property listings containing essential information about location, rent costs and photographic content. Users can access an intelligent chatbot system which helps property owners by both managing inquiries and providing specific property suggestions. The managed search tool enables renters to narrow down housing options through a location-based approach, a price range selection, and feature-specific search criteria for a quick property search outcome. The app utilizes Firebase real-time management to deliver simplified property rental solutions that serve the needs of landlords and renters.

### 2.1.1. Purpose

The mobile application leverages Flutter framework technology to establish an effective solution that links property owners with prospective renters. This application features a solution for rental market issues by enabling property owners to outline listings with mandatory information and add location data and rental amounts and property images. The AI chatbot system provides owners assistance in property proposals and question response functions. The search functionality within the rental application grants users modern filtering tools to locate homes matching their precise property requirements including price and location. The application depends on Firebase to handle data processing alongside user authentication and provides real-time functionality and secure environments to users. The solution provides an updated rental processing system through its intuitive interface which caters to all property owners and renters.

## 2.1.2. Document Conventions

### Font and Formatting:

The main body of text presents in Calibri font at size 12 points with 1.5 lines between each entry for improved readability. Headings together with subheadings are displayed using bold text to enhance navigation. The essential terms receive italicization for better emphasis on fundamental concepts.

### Requirement Numbering:

Functional requirements use FR-1 as their identifier whereas NFR-1 indicates non-functional requirements through unique identification for easy locating.

### Priority Indicators:

The documentation presents requirements with priority indicators that indicate their relative importance through high medium and low rankings. The assigned priority indicators allow development teams to establish their work sequences during testing and development stages.

### Reference Format:

Every external reference within the document adopts a standard citation method by supplying links to sources or publishing information.

## 2.1.3. Intended Audience and Reading Suggestions

1. **Developers:**

   The application developers will obtain technical guidance, setup documentation, and small sections of code. Before starting to read the document, explore the Technology Stack along with the Installation Instructions.

2. **Project Managers:**

   The Overview, Purpose, and Future Work sections in the document provide project managers with crucial information about objectives along with schedule expectations.

3. **Testers:**

   The Quality assurance testers can use the Features and Usage sections to grasp the expected app behavior and develop the testing tasks.

4. **End Users (Property Owners and Renters):**

   The Introduction along with Usage sections serve as the starting point for both property owners and renters who want to access the app for property searching.

5. **Contributors:**

   Project contributors interested in cooperation should dedicate their attention to both Contributing and Future Work sections for required guidance.

**Reading Suggestions:**

New Users should begin with the Introduction to grasp the application purpose.
For technical direction about both features and technology stack Developers must work with Testers. Users looking for usage instruction should read the usage section of the application.

### 2.1.4. Product Scope

Through its user-friendly interface the Home Rentals application serves property owners together with renters inside the rental process. The key features include:

1. **User Registration:**
   - Both owners and renters possess the ability to establish their accounts along with account management features.
2. **Property Listings:**
   - Through the application users can submit properties by adding essential information such as location together with renting prices and images and property specifications.
3. **Search and Filter**:
   - With the Home Rentals application users can effortlessly locate appropriate properties through mapping features together with specified cost levels and detailed property features.
4. **AI-powered:**
   - A smart AI-powered robot helps property owners manage their queries through a chat interface while making specific property recommendations to them.

### 2.1.5. References

1. **Flutter Documentation:**

   Official Flutter documentation for framework guidelines and best practices. Available at: Flutter Documentation

2. **Firebase Documentation:**

   Guides for using Firebase for backend support in mobile applications. Available at: Firebase Documentation

3. **Research Articles on AI:**

   Local research publications on AI and machine learning applications, available through institutions like the National University of Sciences and Technology (NUST). Visit: NUST Research

4. **Software Development Practices:**

   Guidelines from organizations like the Pakistan Software Export Board (PSEB) on best practices in software development. Visit: PSEB

## 2.2. Overall Description

### 2.2.1. Product Perspective

Through its mobile functionalities the application functions as a crucial connector between property owners and renters. The system allows property owners to add their key property information including site location alongside rental rates together with picture uploads. Excellent property searchability exists for renters because they can locate properties precisely matching their requirements. Owners receive support from AI-powered chatbots which answers inquiries and suggests specific properties to enhance their successful property listing process.

### 2.2.2. Product Functions

- **Property Listing and Search**: users have the ability to build new property listings then browse listings by setting multiple search parameters.

- **User Registration and Authentication**: The platform provides safe registration methods alongside authentication services for managing owner and renter accounts.

2.2.3. **Filtering Options:** Users who rent properties have access to various filtering options to search listings according to location and price ranges and additional specified required.

### 2.2.4. User Classes and Characteristics

- **Property Owners**: Individuals or businesses who want to lease properties compose one group at stake.
- **Renters**: Individuals seeking rental properties.
- **Admin**: The application requires a managerial system known as "Admin" to handle data integrity and record user activities.

### 2.2.5. Operating Environment

The software application will work under these conditions:

- **Hardware Platform**: The system functions through smartphones and tablets running the operating systems version 8.0 and above.
- **Operating System**: Supports iOS (version 12 and above) and Android (version 8.0 and above).
- **Software Components**: Integrates with Firebase for real-time database management and Google Cloud for storage solutions.

### 2.2.6. Design and Implementation Constraints

- **Regulatory Policies**: Must adhere to local real estate laws and regulations.
- **Hardware Limitations**: The system must optimize operations to work well on devices that have restricted processing capabilities as well as restricted memory availability.
- **Third-party Services**: The application requires external API integrations for both payment processing functions along with mapping services implementations.
- **Security Considerations**: User data protection occurs when developers organize secure encryption standards throughout their system.

2.2.7. **Design Conventions**: The design of the application needs to follow Flutter development best practices together with UI guidelines in order to maintain consistency.

### 2.2.8. User Documentation

- **User Manual**: Detailed instructions for both property owners and renters.
- **Online Help**: Contextual help available within the application.

### 2.2.9. Assumptions and Dependencies

**Assumptions**:

- The app functions properly because users access reliable internet connections.

The property owners need basic knowledge of technology in order to transfer their listings to the system.

**Dependencies**:

- Reliance on third-party libraries and frameworks for AI functionality.
- The application depends on Firebase services both for data storage operations and authentication procedures.

## 2.3. External Interface Requirements

### 2.3.1. User Interfaces

**Logical Characteristics**: The Mob app implements a clean interface design with user-friendly navigation that follows mobile application standard guideline. Key components include:

- **Sample Screens:** The application includes three representative displays which consist of the home page with search capabilities and screens for property listings in addition to user account management options.
- **Standard Buttons**: A standard set of buttons will appear throughout screens with a help feature available from every display.

### 2.3.2. Hardware Interfaces

- **Supported Devices**: The application provides support to different smartphones alongside tablets.
- **Data Interaction**: The app establishes connections with the camera of the device to transfer images and requires location services for mapping functions.

### 2.3.3. Software Interfaces

- **Database**: Utilizes Firebase Fire store for real-time data management.
- **Operating Systems**: Users can access this system through Android along with iOS operating systems using their most up-to-date software versions.
- **API Integration**: Integrates with third-party APIs for payment processing and geolocation services.

### 2.3.4. Communications Interfaces

- **Email Notifications**: The application sends email alerts to users for tracking their account transactions.
- **Communication Protocols**: The application employs HTTPS as its communication protocol for secure transmission of data.
- **Data Transfer Rates**: The system should support low bandwidth operations with modifications that allow functionality across restricted network conditions.

## 2.4. System Features

The section arranges functional requirements through system features to present the main app services. The description along with priority, user interactions and specific requirements compose each feature of the system.

## 2.4.1. System Feature 1

**User Registration and Authentication**

#### 2.4.1.1. Description and Priority

Registered users access built-in acc authentication in addition to new user account registration. The feature provides registration and authentication through login and logout and role selection between owner or renter.

#### 2.4.1.2. Stimulus/Response Sequences

**Registration**

- New user registration starts when the system receives required information such as email and password and designated role.
- User registration triggers the system to verify the account by email while keeping all profiles safely in the database.

**User Login**

- Stimulus: User submits login credentials.
- The system verifies submitted credentials to activate the app for user access.

#### 2.4.1.3. Functional Requirements

- **REQ 1**: dictates that the system needs to enable new user registration through email combined with password authentication.
- **REQ 2**: The system requires capabilities to enable users for access and exit through login and logout functions.
- **REQ 3**: Security standards demand that the system should protect user credentials through secure data storage mechanisms which also prevent unauthorized access.
- **REQ 4**: The system needs to alert users about unsuccessful login attempts by providing password reset capabilities.

- **Functional Requirements of User Registration**

| Requirement ID | Description | Priority |
|---|---|---|
| FR-UR1 | The application must contain a registration form that requires users to enter their personal details including name alongside email and phone numbers and password. | High |
| FR-UR2 | Validate input fields for proper formatting and completeness. | High |
| FR-UR3 | The system must verify that passwords follow minimum security requirements which include character length and special characters. | High |
| FR-UR4 | User accounts should be built using Firebase Authentication as the fundamental authentication mechanism. | High |
| FR-UR5 | Display success/failure messages after registration. | Medium |
| FR-UR6 | Send email verification to newly registered users. | Medium |

**Table 2.1 Functional Requirements User Registration**

## 2.4.2. System Feature 2

**Property Listing Management**

### 2.4.2.1. Description and Priority

Real estate professionals can use this tool to generate new listings and modify existing ones and remove listing entries including necessary information about location and suitable rent rates along with photo inputs.

**Priority:** High.

### 2.4.2.2. Stimulus/Response Sequences

1. **Adding a New Listing**

- Owner selects the "Add Property" option then provides essential information including property address and rent details and features and photographs.
- The system saves the added listing after which it appears in the search results.

2. **Editing a Listing**

- The owner selects an existing listing which triggers the modification of its details.
- System saves the modified information before it refreshes the listing details to search returns.

### 2.4.2.3. Functional Requirements

- The system needs to provide property owners with a capability to submit their real estate data with property addresses and rental information and images.
- The system requires functionality for owners to modify and remove their posted listings from the system through.
- The system must validate all user inputs thus confirming that needed fields get completed and maintain correct formatting.
- The system needs to send notification to owners whether listing operations succeed or fail.
- The system must possess a database system which provides quick access to stored listings.

- **Functional Requirements of Property Listing**

| Requirement ID | Description | Priority |
|---|---|---|
| FR-PL1 | The feature provides property owners with an interface to create new property listings that require title input and set price and location data. High. | High. |
| FR-PL2 | The system must let users upload several images depicting the property. | High. |
| FR-PL3 | The system requires four textual fields together with size specification and contact information elements for submission. | High. |
| FR-PL4 | Users should obtain permission to edit or delete the listings they post on their own account. | Medium |
| FR-PL5 | Validate required fields before allowing submission. | High. |
| FR-PL6 | Store property data in a cloud database (e.g., Firebase Firestore). | High. |

**Table 2.2 Functional Requirements Property Listing**

## 2.4.3. System Feature 3

**Property Search and Filtering**

#### 2.4.3.1. Description and Priority

The system allows renters to find properties through a search function which lets them narrow down options using location and price and features criteria.

**Priority:** High

#### 2.4.3.2. Stimulus/Response Sequences

1. **Property Search**
    - When the renter completes their search criteria entry by specifying their location the system retrieves matching property results.
    - System retrieves all properties which fulfill the entered criteria for display to the user.
2. **Filtering Results**
    - The system modifies results based on new search criteria that the renter selects from options like property size and pet permission.
    - The system implements results reduction according to applied filters.

#### 2.4.3.3. Functional Requirements

- Requires the system to have a search box that enables renters to specify locations along with search terms.
- The system must enable renter search based on property costs and number of bedrooms together with selective amenities.
- The system must deliver search results which present images of properties together with their prices and short descriptions.
- The system requires functionality to show no results when the search criteria produce no matching outcomes.
- States that system functionality must enable renters to save their search results for future fast retrieval**.**

- **Functional Requirements Search and Filter**

| Requirement ID | Description | Priority |
|---|---|---|
| FR-SF1 | The search tool gives users an option to enter specific keywords like location or property type through a search textbox. | High |
| FR-SF2 | Search results should display fundamental property information which includes images together with price and location and a summary description. | High |
| FR-SF3 | Display search results with key property details including images, price, location, and brief description. | High |
| FR-SF4 | Under situations where the search finds no matching properties the system should display an approachable notification message to users. | Medium |
| FR-SF5 | Users should have the capability to sort results according to relevance and price both ascending and descending and date added. This functionality will be implemented through. | Medium |

**Table 2.3 Functional Requirements Search and Filter**

## 2.5. Other Nonfunctional Requirements

### 2.5.1. Performance Requirements

1. **Response Time**:
    - Users should see property listing search results within a period of 2 seconds. Every user authentication process demands less than one second to finish execution.
2. **Concurrent Users**:
    - The system enables performance for 1000 simultaneous users who experience no slowdown when activity peaks.
3. **Data Load Time**:
    - Users expect property images to load within a period of 3 seconds to ensure seamless browsing.

### 2.5.2. Safety Requirements

- **Data Protection**:
    - During both data transmission and storage all user information receives encryption for the purpose of privacy protection.
- **User Guidance**:
    - The app should give step-by-step instructions for users to handle applications safely and tell users to avoid distributing sensitive data.
- **Emergency Features**:
    - The user interface will incorporate a function to enable reporting properties which create safety issues.

### 2.5.3. Security Requirements

- **User Authentication**:
  - Secure identity verification through OAuth2 methods should be used because of its authentication capabilities.
- **Data Privacy**:
  - Authorized personnel should be the only ones who can see personal information that includes emails and phone numbers.
- **Access Control**:
  - The system must provide designated access permissions to Owner, Renter, Admin user roles in order to maintain security features.

### 2.5.4. Software Quality Attributes

- **Usability**:
  - Design an interface which makes tasks easily executable through three clicks maximum.
- **Reliability**:
  - The system demonstrates 99.9% uptime to provide constant availability for users who need access.
- **Maintainability**:
  - The system must follow clean coding standards to make system updates and bug correction tasks simple.
- **Scalability**:
  - The system must feature an app architecture design which ensures smooth operation when feature numbers and user numbers increase.

# Chapter 3

# Use Case Analysis

# **Chapter 3:** System Analysis

## 3.1. Use Case Model

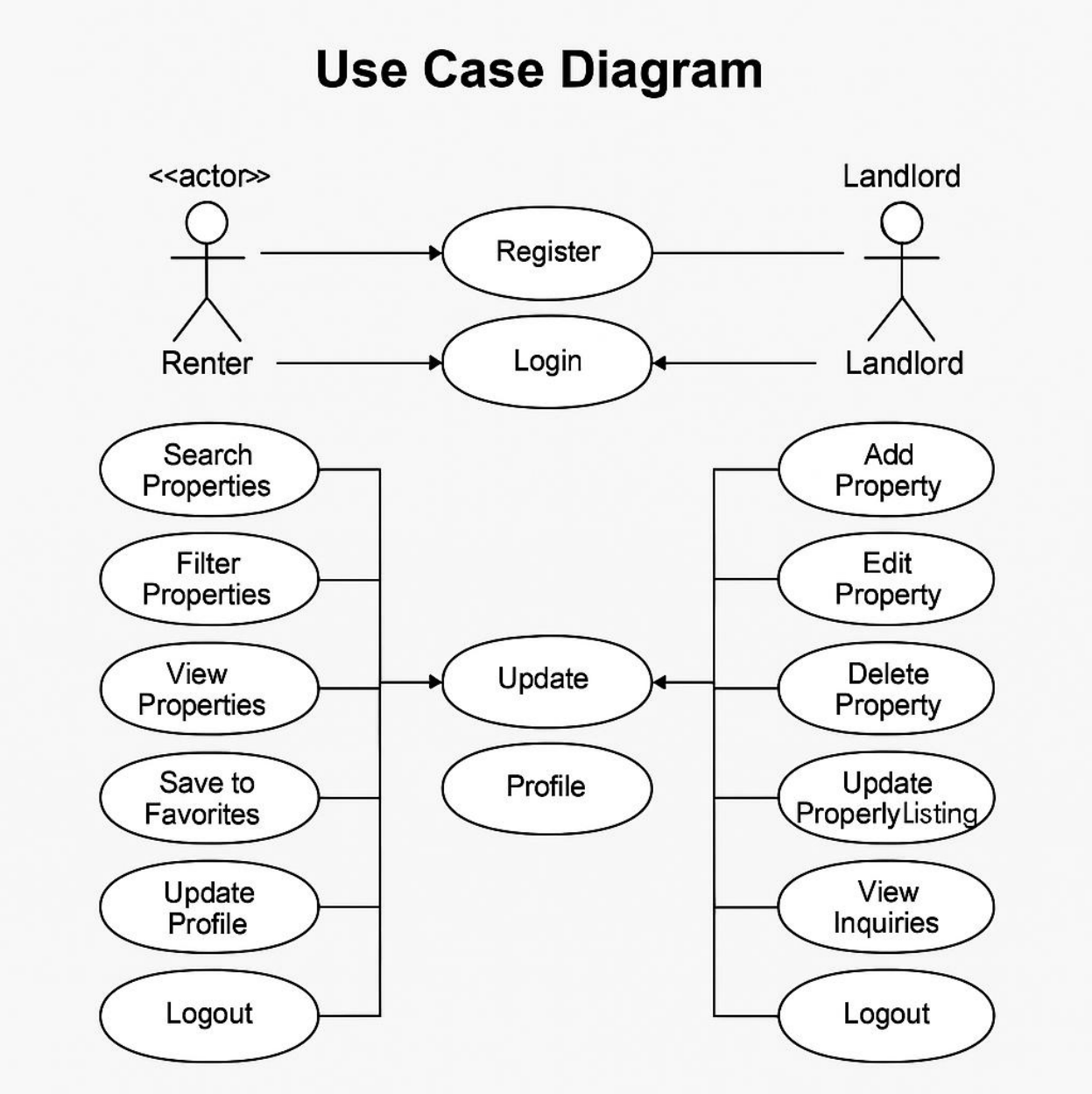



**Figure 3.1**

## 3.2. Use Case Descriptions

- **UC-01: User Registration**

| | |
|---|---|
| Use Case ID | UC-01 |
| Use Case Name | User Registration |
| Actors | User |
| Preconditions | Prior to use users need to have both application installation active and internet connectivity. |
| Postconditions | User registration allows the system to create a new account which saves data both in Firebase Authentication and Firestore. |
| Main Flow | 1. User opens app.<br>2. Taps on 'Register'.<br>3. Enters required info.<br>4. Submits.<br>5. The system accepts and saves the input data.<br>6. Confirms success. |
| Alternate Flows | User registration supports two alternative authentication methods through Google and Facebook instead of traditional email and password authentication. |
| Exceptions | The system will block the entry if the user submits a duplicate email or phone address or if any of their information contains format errors. Additionally, it will stop the process in the event of network outages. |
| Includes/Extends | Includes: Email Verification |
| Priority | High |
| Frequency of Use | Often |
| Special Requirements | The application needs Firebase Authentication together with form validation and secure data storage. |
| Assumptions | The user maintains the of their entered information. |

**Table 3.1 Use Case Descriptions**

- **UC-02: Email Verification**

| | |
|---|---|
| Use Case ID | UC-02 |
| Use Case Name | Email Verification |
| Actors | System, User |
| Preconditions | User has completed registration. |
| Postconditions | There are postconditions indicating that email verification status becomes present in user profile entries. |
| Main Flow | 1. System sends verification email.<br>2. The user receives an email that leads to clicking the link inside it.<br>3. System updates verification status. |
| Alternate Flows | The User can ask for a new verification email delivery two times. |
| Exceptions | System records an error when delivery of the email fails or when the link becomes invalid. |
| Includes/Extends | Extended by: Registration |
| Priority | Medium |
| Frequency of Use | Once per user |
| Special Requirements | Email server integration, secure token generation. |
| Assumptions | The user has the ability to access the email address they entered during system operation. |

- **UC-03: User Login**

| | |
|---|---|
| Use Case ID | UC-03 |
| Use Case Name | User Login |
| Actors | Registered User |
| Preconditions | The user must possess a verified email account and already be registered. |
| Postconditions | The user gets a successful log in during which the application takes them to the dashboard page. |
| Main Flow | 1. User enters email/password.<br>2. Clicks 'Login'.<br>3. System authenticates.<br>4. Redirects to home screen. |
| Alternate Flows | Login using Google/Facebook. |
| Exceptions | The system presents exceptions in the following cases: wrong credentials exist or the account has been locked or the internet connection is unavailable. |
| Includes/Extends | None |
| Priority | High |
| Frequency of Use | Often |
| Special Requirements | Firebase Auth integration, secure token handling. |
| Assumptions | The system runs under the conditions that credentials match correctly and the user remains active during authentication. |

- **Use Case 04: Property listing**

| | |
|---|---|
| ID | 4 |
| Use Case Name | Property Listing Management |
| Act | Landlord |
| Precondition | User log in must be. |
| Postcondition | Generates system new visible property when landlord created completed. |
| Main Flow | 1. User selects 'Add Property'.<br>2. Inputs details.<br>3. Uploads photos.<br>4. Submits listing.<br>5. System stores and displays property. |
| Alternate Flows | The system offers editing or deleting functions for existing listings. |
| Exceptions | Missing fields; Image upload failure; Database error. |
| Includes/Extends | None |
| Priority | High |
| Frequency of Use | Moderate |
| Special Requirements | Cloud Firestore and Storage. |
| Assumptions | The user possesses all necessary property information needed for the system. |

- **UC-05: Property Search and Filtering**

| | |
|---|---|
| Use Case ID | UC-05 |
| Use Case Name | The Use Case named Property Search and Filtering. |
| Actors | User (Guest or Registered) |
| Preconditions | Property data available in database. |
| Postconditions | The system will show matching results to the user after postconditions are met. |
| Main Flow | 1. User opens search.<br>2. Sets filters.<br>3. Submits query.<br>4. Results displayed. |
| Alternate Flows | Sort results by price/date. |
| Exceptions | The system exhibits three possible faults: it shows no results while being slow and produces invalid results from filtering. |
| Includes/Extends | None |
| Priority | High |
| Frequency of Use | Often |
| Special Requirements | Real-time Firestore queries. |
| Assumptions | Database is populated and accessible. |

- **UC-06: View Property Details**

| | |
|---|---|
| Use Case ID | UC-06 |
| Use Case Name | View Property Details |
| Actors | Users |
| Preconditions | A user selects one of the properties shown in the listings. |
| Postconditions | Property detail page displayed. |
| Main Flow | 1. User taps a listing.<br>2. The system shows all information about the selected property. |
| Alternate Flows | Users have two alternate actions by either getting in touch with owners or saving properties. |
| Exceptions | Listing removed or database error. |
| Includes/Extends | None |
| Priority | High |
| Frequency of Use | Often |
| Special Requirements | Media viewer for images, map view integration. |
| Assumptions | The property data exists precisely and users can access it. |

- **UC-07: Edit User Profile**

| | |
|---|---|
| ID Use Case | Use Case 07 |
| Name of Use Case | Edit Users Profiles |
| Actors. | Users |
| Precondition | User login must be. |
| Postcondition | Profile information updated. |
| Main Flow. | 1. User must be open profile.<br>2. User click on edit button.<br>3. Data change from profile.<br>4. Submits.<br>5. System updates info. |
| Alternate Flows | The user can revise single sections of their profile. |
| Exceptions | Validation errors; Database issues. |
| Includes/Extends | None |
| Priority | Medium |
| Frequency of Use | Occasionally |
| Special Requirements | Firestore update and image handling. |
| Assumptions | The user possesses accurate information that needs to enter the system. |

- **UC-08: Delete Property Listing**

| | |
|---|---|
| Use Case ID | UC-08 |
| Use Case Name | Delete Property Listing |
| Actors | Property Owner |
| Preconditions | A user needs to be logged in with active listings to proceed with the precondition. |
| Postconditions | Listing is removed. |
| Flow | 1. Users listing select.<br>2. Selected delete.<br>3. OK<br>4. System delete data. |
| Alternate | Del cancel. |
| Exception | Network error or permission denied. |
| Include/Ext | None |
| 1st priority | mid |

- **Use Case 9: Favorite property saves**

| | |
|---|---|
| ID | 9 |
| Name of UC | **Favorite property** |
| Act | Users |
| Precondition | Users must be logged in. |
| Postcondition | Saves property to fav. |
| Flow | 1. view fav property<br>2. Click save.<br>3. store system to fav in id. |
| Alternate Flows | Remove from favorites. |
| Exceptions | Database writes failed. |
| Includes/Extends | None |
| Priority | Medium |
| Frequency of Use | Often |
| Special Requirements | Favorites collection per user. |
| Assumptions | The property attracts sufficient interest for the user to save it. |

- **Use Case 10: Rec Notification**

| | |
|---|---|
| ID | UC-10 |
| Name of UC | **Rec Notification** |
| Act | Users |
| Preconditions | Users have enabled notifications. |
| Postconditions | The system will send a notification to users who have chosen property matching alerts active. |
| Flow | 1. System compares new listings to preferences.<br>2. Sends FCM push message. |
| Alternate Flows | User turns off alerts. |
| Exceptions | FCM service fails. |
| Includes/Extends | None |
| Priority | Medium |
| Frequency of Use | Often |
| Special Requirements | Firebase Cloud Messaging setup. |
| Assumptions | The operation requires both device tokens to be valid along with previously established user preferences. |

- **Use Case 11: Log-out**

| | |
|---|---|
| ID | 11 |
| Use Case Name | Logout |
| Act | Users |
| Precondition | Users must be logging in. |
| Postcondition | Session ended. |
| Flow | 1. Users selects L-out.<br>2. The system signs out into the login screen. |
| Alternate Flows | None |
| Exceptions | Logout action fails. |
| Includes/Extends | None |
| Priority | Medium |
| Frequency of Use | Often |
| Special Requirements | Firebase sign-out () call. |
| Assumptions | The user needs a secure method to end their current session. |

# Chapter 4

# System Design

# Chapter 4: System Design

Through its integrated system the Home Rentals application delivers a uniform experience for each user. The high-level system design provides detailed information about application structure as well as database operations and information workflow. The design focuses on three main components before detailing their structure to boost system performance and adaptability as well as promote data protection and system reliability.

## 4.1. Architecture Diagram

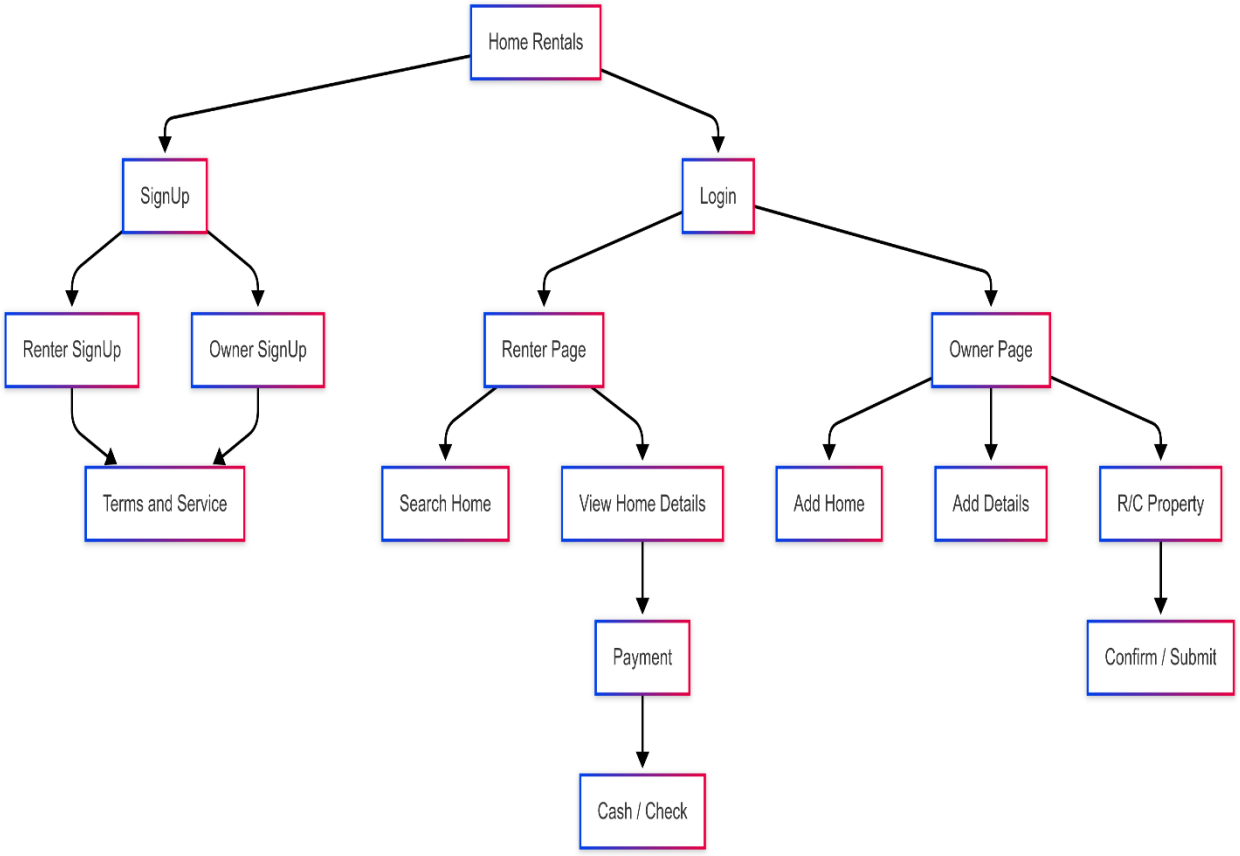



**Figure 4.1 Architecture Diagram**

## 4.2. Domain Model

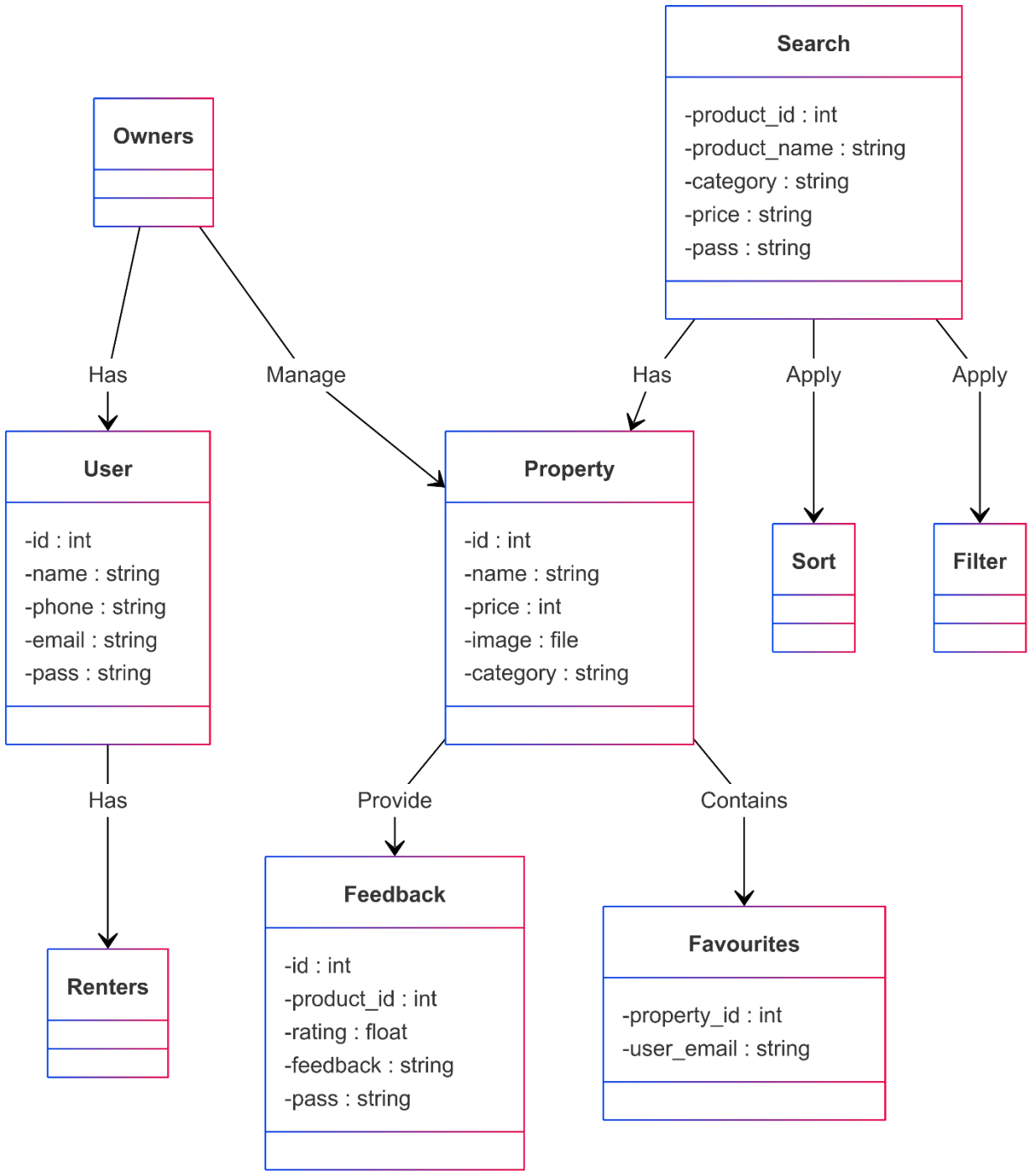



**Figure 4.2 Domain Model Diagram**

## 4.3. Entity Relationship Diagram with data dictionary

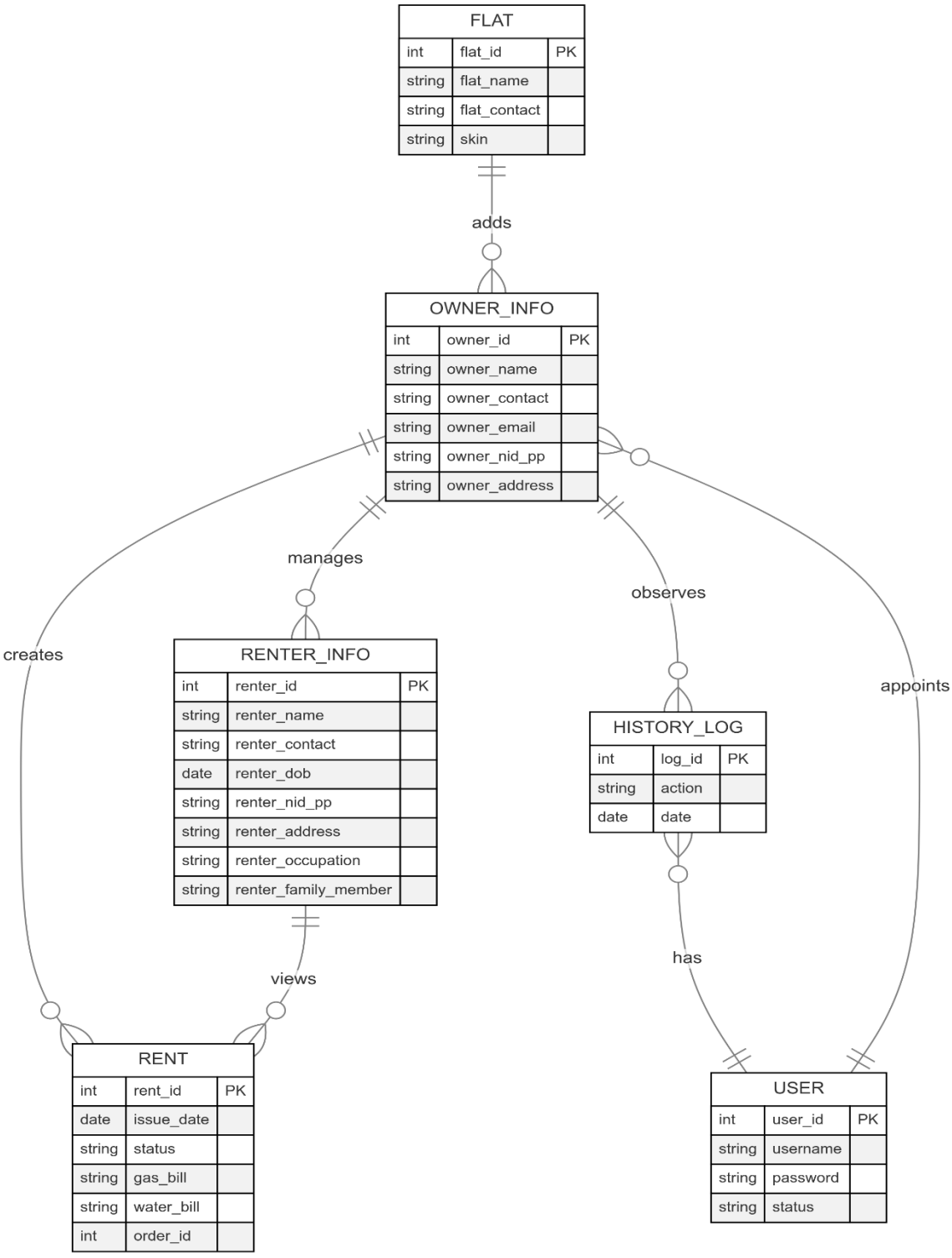



**Figure 4.3 Entity Relationship Diagram**

## 4.4. Class Diagram

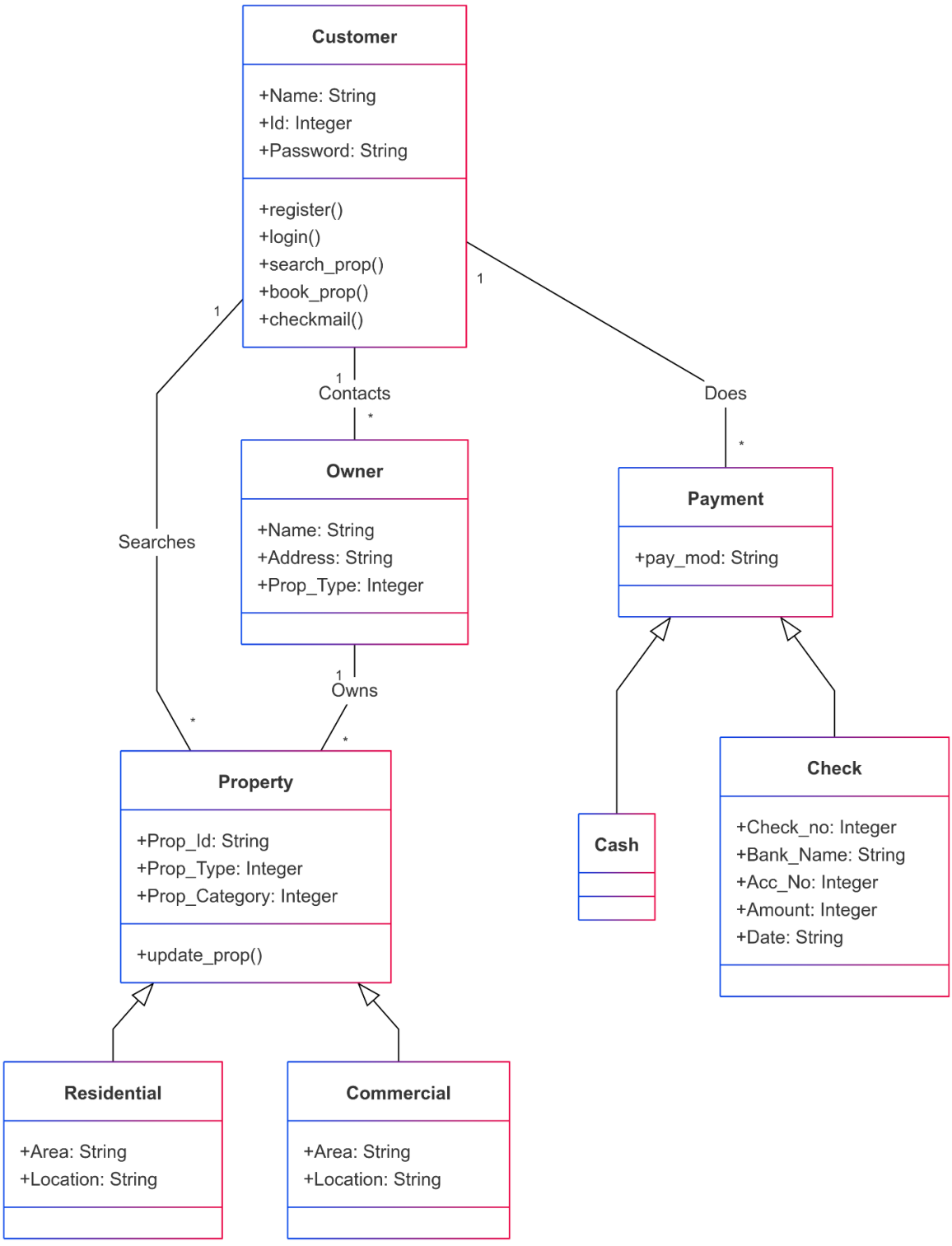



**Figure 4.4 Class Diagram**

## 4.5. Sequence / Collaboration Diagram

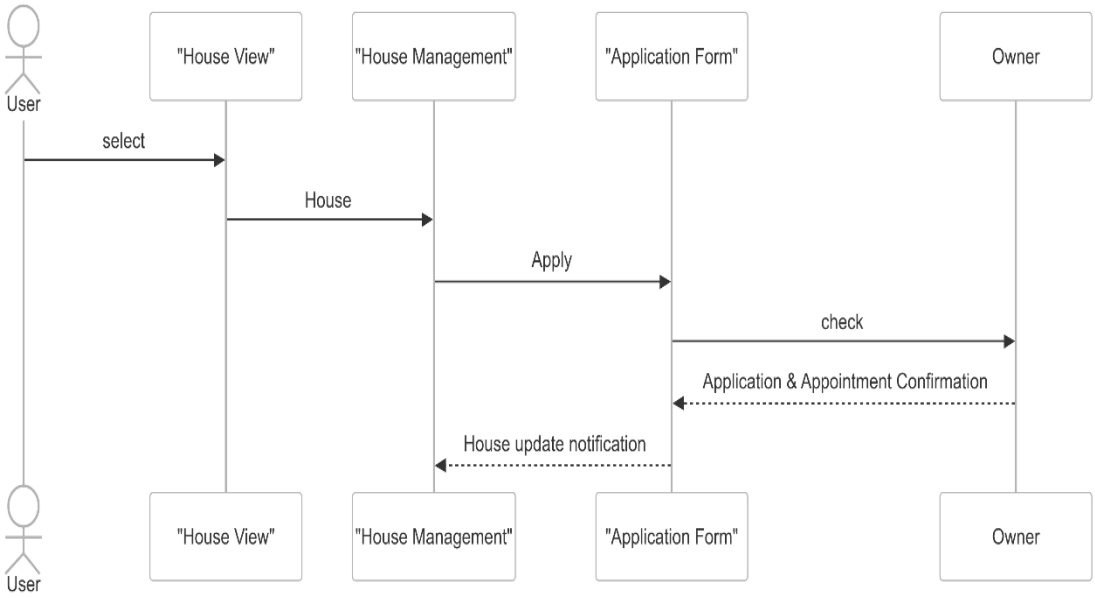



**Figure 4.5 Sequence/Collaboration Diagram**

## 4.6. Operation contracts

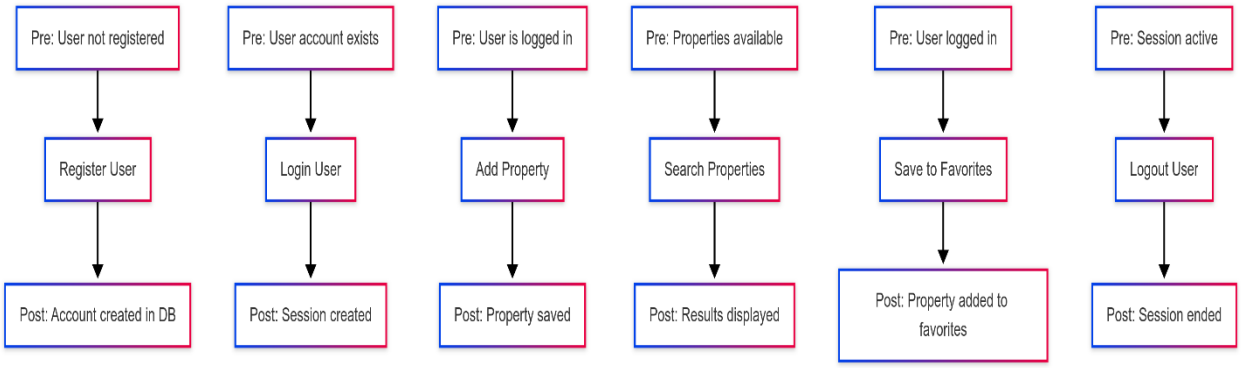



**Figure 4.6 Operation Diagram**

## 4.7. Activity Diagram

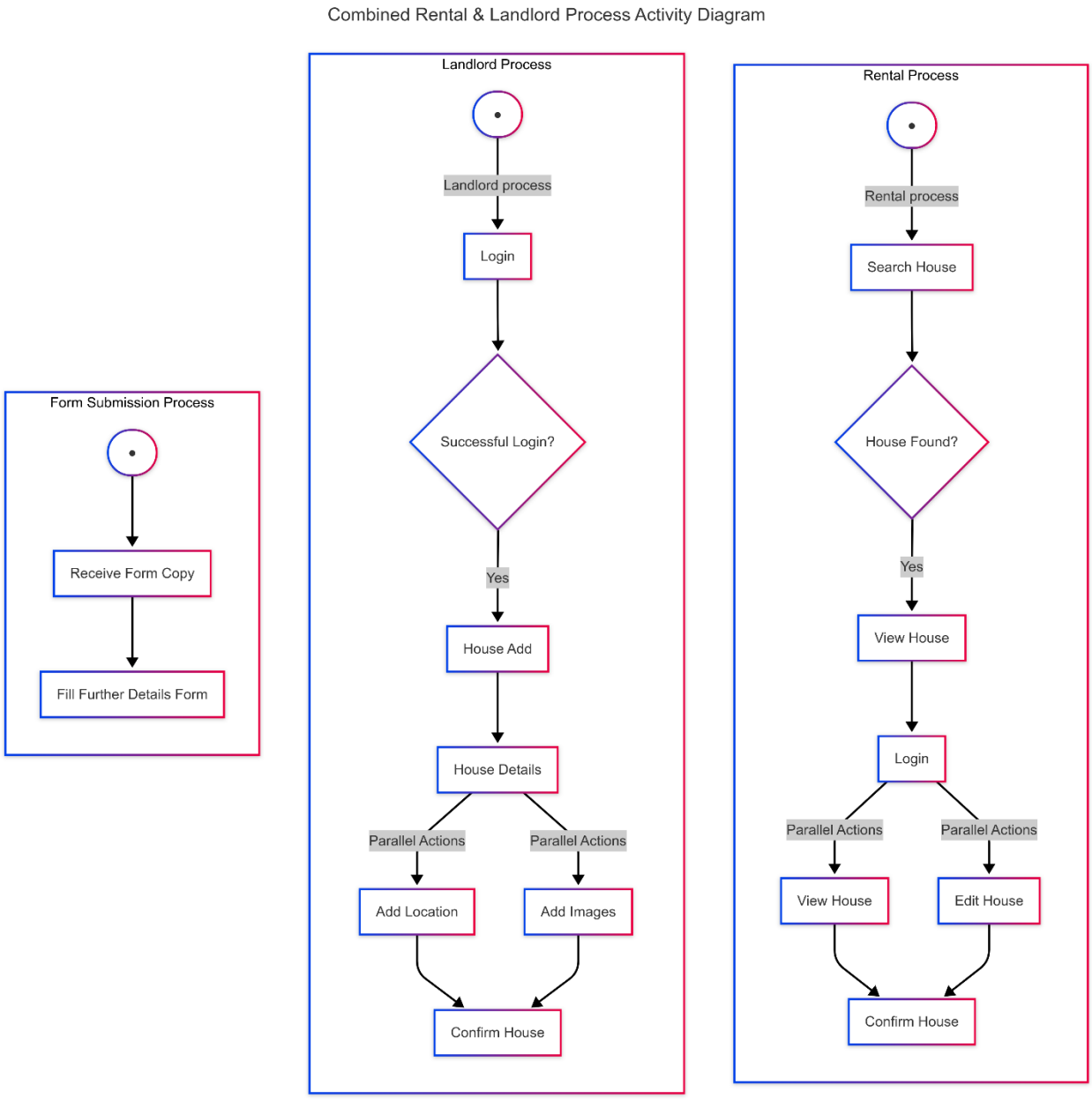



**Figure 4.7 Activity Diagram**

## 4.8. State Transition Diagram

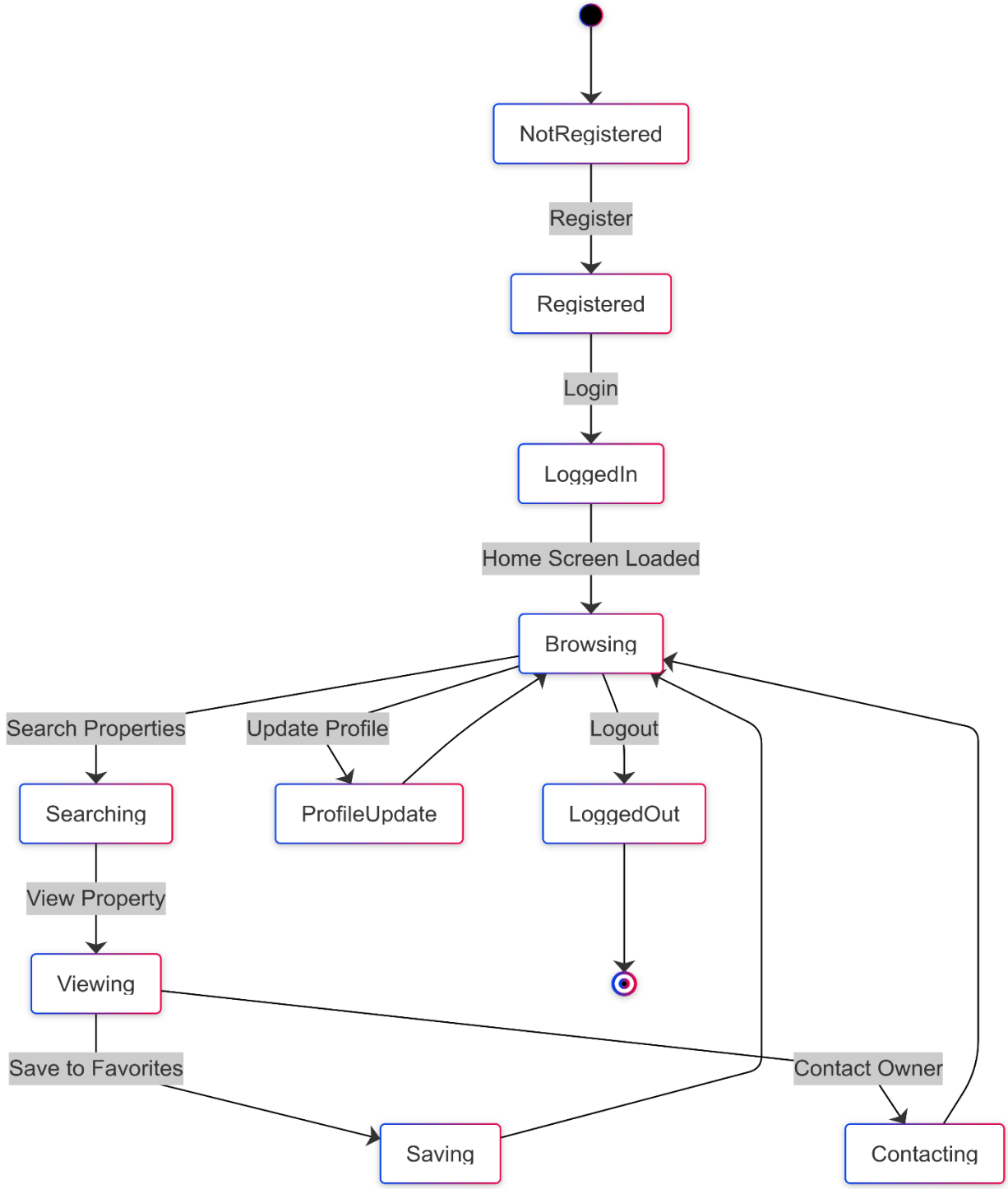



**Figure 4.8 State Transition Diagram**

## 4.9. Component Diagram

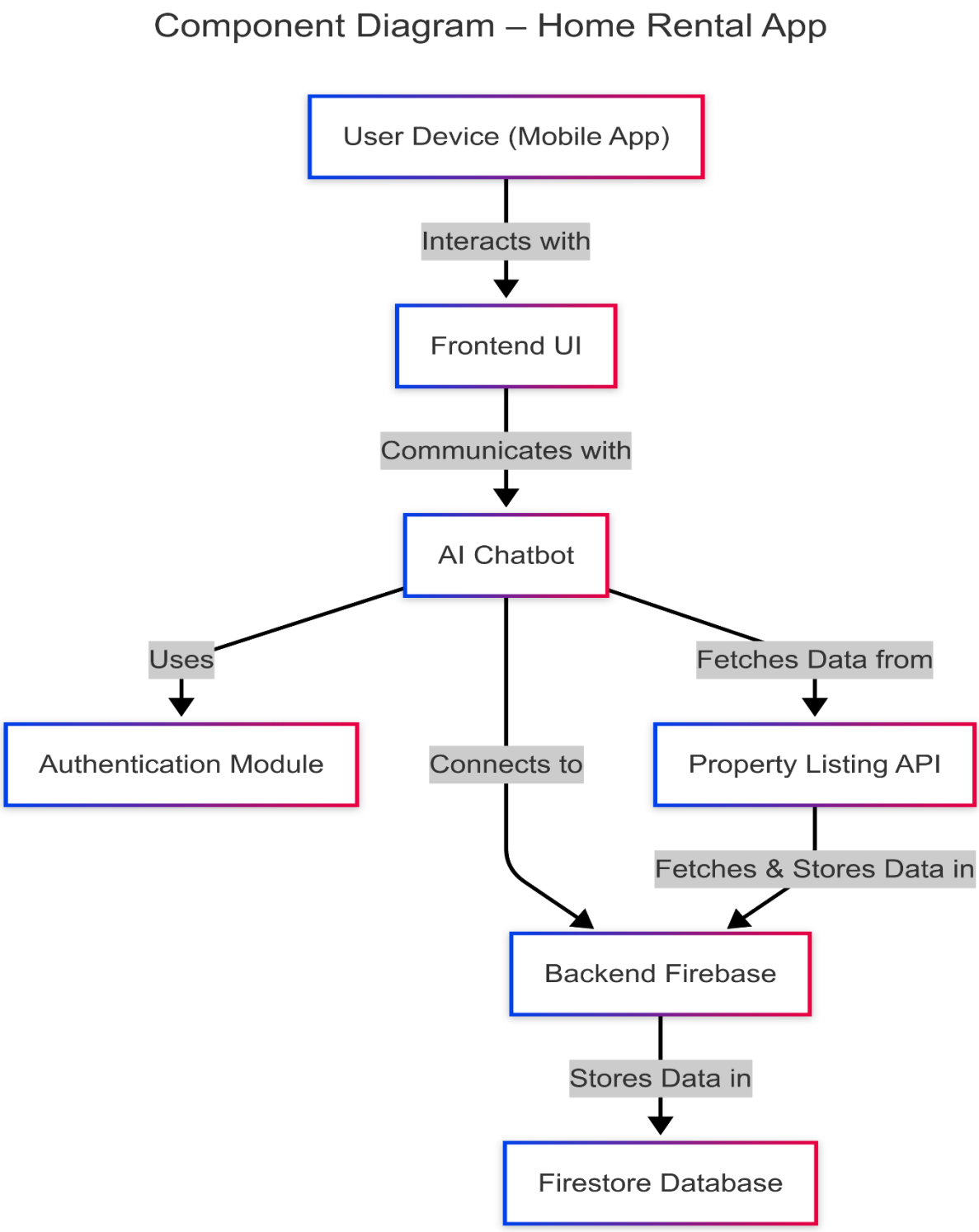



**Figure 4.9 Component Diagram**

## 4.10. Deployment Diagram

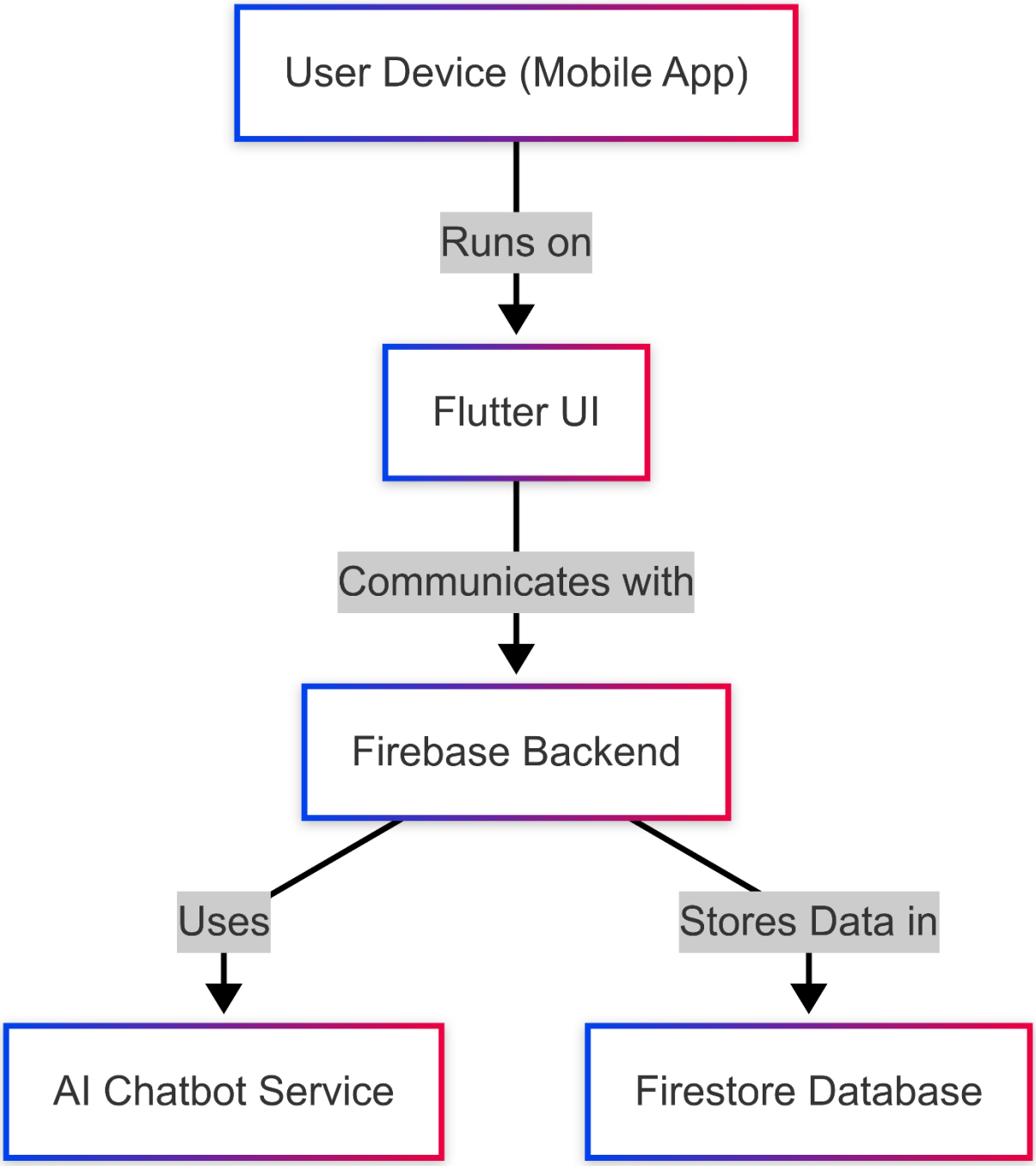



**Figure 4.10 Deployement Diagram**

## 4.11. Data Flow diagram

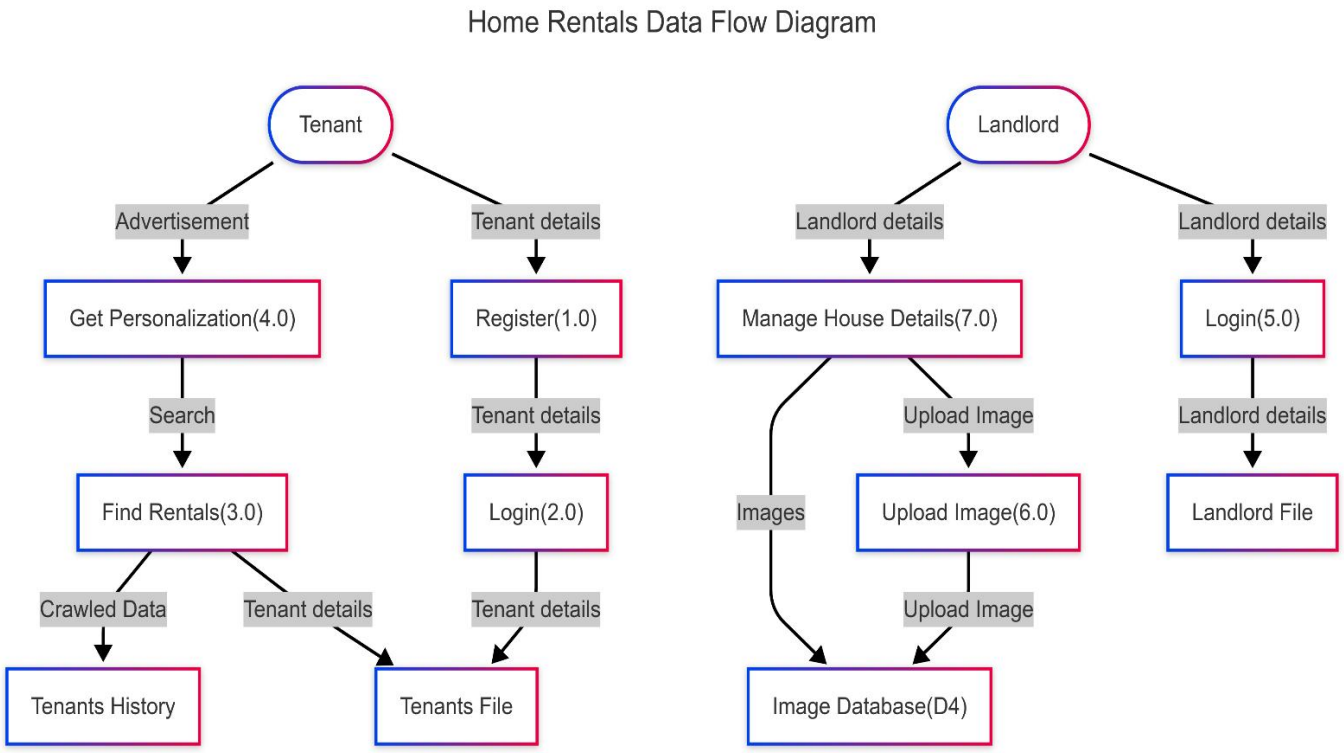



**Figure 4.11 Data Flow Diagram**

# Chapter 5

# Implementation

# Chapter 5: Implementation

## 5.1. Important Flow Control/Pseudo codes

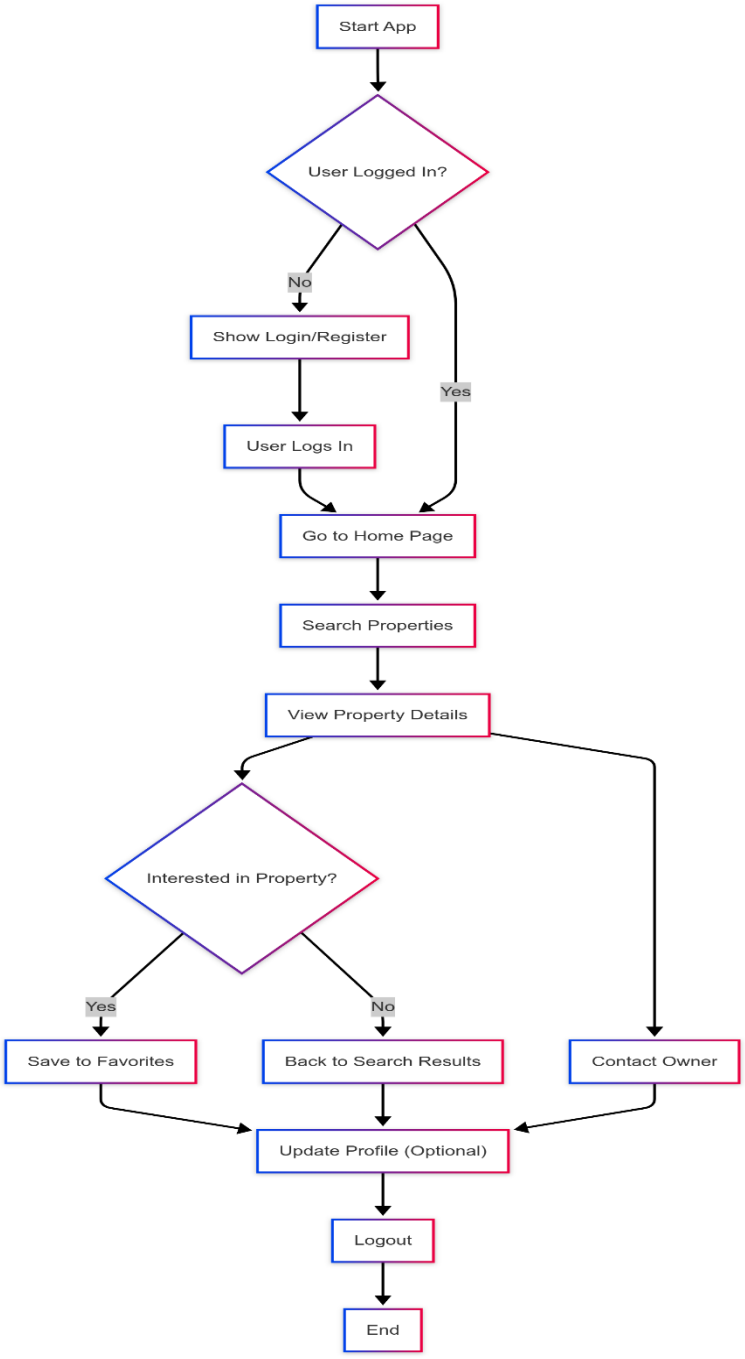



**Figure 5.1 Important Flow Control Diagram**

**User Registration (Homeowner or Renter)**:

- The system accepts user details which comprise name together with email and password information.
- Logic:
    - Validate input
    - The user details are stored in the database of the Firebase Authentication system.
    - The system saves user information in the Firebase Authentication database while moving to the next screen which can be either the home screen or dashboard.

**Property Listing** (for Homeowners):

- Source data consists of property details which contain both geographic location information and rent price together with picture files.
- Logic:
    - Validate input
    - The application uses Firebase Firestore for property data storage.
    - Display confirmation message

**Property Search** (for Renters):

- Input: Location, Budget, Amenities
- Logic:
    - A search of relevant properties in Firebase Fire store occurs based on specified criteria.
    - Display results

- **Tools and Libraries Used**

| Tool/Library | Purpose |
|---|---|
| Flutter | Framework for developing cross-platform mobile applications. |
| Firebase Auth | Firebase Auth provides authentication features for user authentication services including user signup and login with added security components. |
| Firestore DB | The Firestore DB system functions as a NoSQL cloud database which stores both user-related data and property information. |
| Firebase Storage | The user-uploaded images get stored in Firebase Storage. |
| Provider | Provider serves as the state management library throughout Flutter applications. |
| Firebase Messaging (FCM) | Push notifications for updates or property alerts. |
| Dart | The Dart programming language functions as the main language of Flutter application development. |

**Table 5.1 Tools and Libraries Used**

## 5.2. Components, Libraries, Web Services and stubs

During development of the house rental application using Flutter-based technology these essential components as well as libraries were employed:

- **Firebase Authentication**: Used for secure user registration and authentication for both homeowners and renters.
- **Flutter** : App developed by using flutter.
- **Firebase Fire store**: The application employs Firebase Firestore to store all property listings and user information.
- **AI API**: The AI-generated API allows real-time description creation that follows homeowner input of key features.
- **Provider Package**: Used for state management in Flutter.

**Web Services**:

- **Property Listings API**: The Property Listings API serves as a self-developed tool for retrieving property listings that match renter search criteria.

## 5.3. Deployment Environment

The test environment consisted of several deployment characteristics as follows:

- **Operating Systems**:
    - The Target Software Development Kit for Android includes version 29 or higher.
- **Development Tools**:
    - The Flutter SDK functioned as the development tool to generate apps simultaneously for Android and iOS devices.
    - Development engineers utilized Xcode to execute testing and deployment of iOS applications.
- **Backend**:
    - **Firebase.** Authentication enables user logins through its system while Firestore manages property data management.

**Deployment on Devices**:

- Testing occurred on actual Android devices in order to test the application's response and compatibility.

## 5.4. Tools and Techniques

- The Flutter SDK handles the development of a cross-platform mobile application and its compilation processes. Two integrated development environments known as Android Studio and VS Code enabled developers to write code and debug the application. I used the Firebase console to set up the Firebase services including Authentication and Firestore. The application utilized Google Cloud Vision API as its core technology for property picture image analysis. The version control system Git enabled both teams and managers to handle collaborative development and source code management.

**Development Techniques**:

- The app obtained full screen size adaptivity through layout widgets from Flutter to provide responsive features. The application uses Provider package to manage state data throughout various screens for its functions. The addition of user-friendly error messages and proper error handling mechanisms with network error and invalid input detection came to address system issues.

## 5.5. Best Practices / Coding Standards

The development process followed these guidelines as best practices together with coding standards:

- **Code Readability**: The team enforced readable code standards through proper naming of variables together with classes and functions. Every function within the system performed only one task.
- **Commenting**: The development process used inline comments to explain complicated code segments and method comments to explain the functions' purposes.
- **Modular Code**: The design followed modular approaches by separating code into widget and class modules for making code more reusable and maintainable.
- **Error Handling**: The try-catch blocks served as error handlers which delivered useful error notifications to end-users.
- **Consistency**: A consistent formatting structure and indentation system occurred throughout every part of the code.
- **UI/UX Standards**: The design had smooth functionality based on UI/UX Standards alongside touch-friendly operations and visible labels for the user.

- **Coding Standards and Practices**

| Standard | Description |
|---|---|
| Naming Conventions | The standard for naming code elements involves camelCase for both variables and methods together with Pascal Case for naming classes. |
| Code Structure | The MVC or MVVM design pattern should be used as the foundation to separate application concerns. |
| Commenting | Regular comments should be placed inline together with documentation when writing complex logic and functions. |
| Modularity | Components should be designed as reusable widgets along with modular sections of code. |
| Error Handling | The system uses try-catch blocks to handle exceptions so they appear gracefully in such situations. |
| Validation | Implement client-side validation for all forms and inputs. |
| Clean Code | The code must follow Dart/Flutter lint rules and remove every unused code segment and import declaration. |

**Table 5.2 Coding Standards and Practices**

## 5.6. Version Control

- **Branching**: Multiple feature and bug fix branches named feature/user-registration and bugfix/login-error existed during development.
- **Commits**: The developers used frequent commits which included descriptive message summaries for their work modifications (“Added property search functionality” represents one example).
- **Pull Requests**: Pull requests enabled code reviews which happened before code integration into the main branch to guarantee quality standards.
- **Collaboration**: Separate developer branches were exploited for work before peer examination took place which then enabled code unification while ensuring high quality and reducing conflicts.

# Chapter 6

# Testing and Evaluation

# **Chapter 6:** Testing and Evaluation

Following section details verification procedures that tested the Home Rentals App for its reliability performance security aspects.

## 6.1. Use Case Testing

**Test Case for UC-01: User Registration**

| Test Case ID | Use Case ID | Test Description | T-step | Expect Res | Actual Res | P/F | Comments |
|---|---|---|---|---|---|---|---|
| T-1 | UC-01 | Verify users' registration works correctly. | 1. Follow main flow steps.<br>2. Verify alternative flows (if any).<br>3. Check exception handling. | All steps execute successfully without errors. | All steps executed successfully. | Pass | No issues found. |

**Table 6.1 Use Case Testing**

**Test Case for UC-02: Email Verification**

| Test Case ID | Use Case ID | Test Description | T-step | Expect Res | Actual Res | P/F | Comments |
|---|---|---|---|---|---|---|---|
| T-2 | UC-02 | email verification works correctly. | 1. Follow main flow steps.<br>2. Verify alternative flows (if any).<br>3. Check exception handling. | All steps execute successfully without errors. | All steps executed successfully. | Pass | No issues found. |

**Test Case for UC-03: User Login**

| ID of case | ID of UC | Description | T-step | Expected Res | Act Res | P/F | Comment |
|---|---|---|---|---|---|---|---|
| T-3 | UC-03 | Verify that the user login works correctly. | 1. Follow main flow steps.<br>2. Verify alternative flows (if any).<br>3. Check exception handling. | All steps execute successfully without errors. | All steps executed successfully. | Pass | No issues found. |

**Test Case for UC-04: Property Listing Management**

| ID of case | UC ID | Description | T-step | Expected Res | Act Res | P/F | Comment |
|---|---|---|---|---|---|---|---|
| TC-04 | UC-04 | Verify that the property listing management works correctly. | 1. Follow main flow steps.<br>2. Verify alternative flows (if any).<br>3. Check exception handling. | All steps execute successfully without errors. | All steps executed successfully. | Pass | No issues found. |

**Test Case for UC-05: Property Search and Filtering**

| ID of T-Case | UC ID | Description | T-step | Expect Res | Act Res | P/F | Comment |
|---|---|---|---|---|---|---|---|
| T-5 | UC-05 | Verify that the property search and filtering works correctly. | 1. Follow main flow steps.<br>2. Verify alternative flows (if any).<br>3. Check exception handling. | All steps execute successfully without errors. | All steps executed successfully. | Pass | No issues found. |

**Test Case for UC-06: View Property Details**

| ID of T-Case | UC ID | Description | T-step | Expect Res | Act Res | P/F | Comment |
|---|---|---|---|---|---|---|---|
| TC-06 | UC-06 | Verify that the view property details work correctly. | 1. Follow main flow steps.<br>2. Verify alternative flows (if any).<br>3. Check exception handling. | All steps execute successfully without errors. | All steps executed successfully. | Pass | No issues found. |

**Test Case for UC-07: Edit User Profile**

| ID of T-Case | UC ID | Description | T-step | Expect Res | Act Res | P/F | Comment |
|---|---|---|---|---|---|---|---|
| T-7 | UC-07 | Verify that the edit user profile works correctly. | 1. Follow main flow steps.<br>2. Verify alternative flows (if any).<br>3. Check exception handling. | All steps execute successfully without errors. | All steps executed successfully. | Pass | No issues found. |

**Test Case for UC-08: Delete Property Listing**

| ID of T-Case | UC ID | Description | T-step | Expect Res | Act Res | P/F | Comment |
|---|---|---|---|---|---|---|---|
| T-8 | UC-08 | Verify that the delete property listing works correctly. | 1. Follow main flow steps.<br>2. Verify alternative flows (if any).<br>3. Check exception handling. | All steps execute successfully without errors. | All steps executed successfully. | Pass | No issues found. |

**Test Case for UC-09: Save/Favorite a property**

| ID of T-Case | UC ID | Description | T-step | Expect Res | Act Res | P/F | Comment |
|---|---|---|---|---|---|---|---|
| T-9 | UC-09 | Verify that the save/favorite a property works correctly. | 1. Follow main flow steps.<br>2. Verify alternative flows (if any).<br>3. Check exception handling. | All steps execute successfully without errors. | All steps executed successfully. | Pass | No issues found. |

**Test Case for UC-10: Receive Notifications**

| ID of T-Case | UC ID | Description | T-step | Expect Res | Act Res | P/F | Comment |
|---|---|---|---|---|---|---|---|
| TC-10 | UC-10 | Verify that the receive notifications works correctly. | 1. Follow main flow steps.<br>2. Verify alternative flows (if any).<br>3. Check exception handling. | All steps execute successfully without errors. | All steps executed successfully. | Pass | No issues found. |

**Test Case for UC-11: Logout**

| ID of T-Case | UC ID | Description | T-step | Expect Res | Act Res | P/F | Comment |
|---|---|---|---|---|---|---|---|
| TC-12 | UC-12 | Verify that the logout works correctly. | 1. Follow main flow steps.<br>2. Verify alternative flows (if any).<br>3. Check exception handling. | All steps execute successfully without errors. | All steps executed successfully. | Pass | No issues found. |

## 6.2. Equivalence partitioning

| Test ID | Field / Module | Input Class (Valid / Invalid) | Test Data | Expected Output | Actual Output | Status |
|---|---|---|---|---|---|---|
| EP-01 | Email Field (Registration) | Valid | user@example.com | Email accepted | Email accepted | Pass |
| EP-02 | Email Field (Registration) | Invalid – Missing '@' or domain | userexample.com | Error message shown | Error message displayed | Pass |
| EP-03 | Password Field | Valid – 6 to 20 characters | P@ssword123 | Password accepted | Password accepted | Pass |
| EP-04 | Password Field | Invalid – Too short | ABC | Error: Password too short | Error triggered | Pass |
| EP-05 | Price Filter | Valid – Integer within range | 15000 | Filter applied successfully | Properties displayed | Pass |
| EP-06 | Price Filter | Invalid – Text input | fifteen thousand | Error: Invalid input type | Input rejected | Pass |
| EP-07 | Property Title | Invalid – Empty string | (empty) | Error: Title required | Validation error shown | Pass |
| EP-08 | Property Title | Valid – Proper string | Furnished Apartment in Lahore | Property title accepted | Title displayed properly | Pass |
| EP-09 | Image Upload | Invalid – Unsupported format | .txt file | Error: Invalid file type | Upload blocked | Pass |
| EP-10 | Image Upload | Valid – JPEG/PNG file | apartment.jpg | Image uploaded successfully | File saved | Pass |

**Table 6.2 Equivalence Partitioning**

## 6.3. Boundary value analysis

| Test ID | Field / Module | Test Scenario | Input Values (Min, Max, Invalid) | Expected Output | Actual Output | Status |
|---|---|---|---|---|---|---|
| BVA-01 | Password Length | Test min/max character limits | 5, 6, 20, 21 | Reject 5/21, Accept 6/20 | Matches expectation | Pass |
| BVA-02 | Property Title Length | Test title field length | 0, 1, 100, 101 | Reject 0/101, Accept 1–100 | Works as expected | Pass |
| BVA-03 | Price Filter Range | Test price boundary values | 0, 1000, 1000000, 1000001 | Reject 0/1000001, Accept range | Boundary checks working | Pass |
| BVA-04 | Image Upload Limit | Test image upload size | 0MB, 1MB, 5MB, 6MB | Reject 0/6MB, Accept 1–5MB | Image validation successful | Pass |
| BVA-05 | Favorites Capacity | Max 50 favorites allowed | 49, 50, 51 | Accept 49–50, Reject 51 | Limit enforced correctly | Pass |
| BVA-06 | Search Keyword Length | Test input text limits | 0, 1, 50, 51 | Reject 0/51, Accept 1–50 | Search input validated | Pass |

**Table 6.3 Boundary value analysis**

## 6.4. Data flow testing

| Test ID | Component | Test Scenario | Data Defined (def) | Data Used (use) | Expected Behavior | Actual Result | Status |
|---|---|---|---|---|---|---|---|
| DFT-01 | User Registration | Register with valid data | name, email, password | Firebase Auth & Firestore | Data stored and reusable | Data saved and reused | Pass |
| DFT-02 | Login | Authenticate login | email, password | Login auth | Session created; data fetched | Login successful | Pass |
| DFT-03 | Add Property | Landlord lists new property | title, description, price, images | Listing and search modules | Visible in search | Listing appeared | Pass |
| DFT-04 | Property Search | Use search filters | searchText, filters | Firestore query | Filtered results shown | Search worked | Pass |
| DFT-05 | Save to Favorites | Save property | property ID | Favorites collection | Property retrievable | Saved successfully | Pass |
| DFT-06 | Update Profile | Change profile data | newName, newPhone | User info screen | Updated profile shown | Profile updated | Pass |
| DFT-07 | View Property Details | Tap on listing | property ID | Fetch detailed info | Property screen loaded | Details loaded | Pass |
| DFT-08 | Push Notification | Send FCM on new listing | listingTitle, receiverToken | FCM push | Notification displayed | Received and opened | Pass |

**Table 6.4 Data Flow Testing**

## 6.5. Unit testing

| Test ID | Module / Function | Test Scenario | Input | Expected Output | Actual Output | Status |
|---|---|---|---|---|---|---|
| UT-01 | validateEmail() | Check email format | test@example.com | Passes validation | Passes validation | Pass |
| UT-02 | validatePassword() | Password too short | 123 | Error shown | Error returned | Pass |
| UT-03 | registerUser() | Valid data | Email, password, name | User created | User created | Pass |
| UT-04 | loginUser() | Valid credentials | Email + password | Redirect to homepage | Dashboard opens | Pass |
| UT-05 | searchProperties() | Search by keyword | Islamabad | Results returned | Correct results | Pass |
| UT-06 | applyFilters() | Filter price range | Min: 10K, Max: 30K | Only in-range shown | Expected results | Pass |
| UT-07 | saveToFavorites() | Save property | Property ID | Added to favorites | Favorites updated | Pass |
| UT-08 | updateUserProfile() | Change name/phone | New name, phone | Updated info saved | Update confirmed | Pass |
| UT-09 | logoutUser() | Logout action | Click logout | Session ends | Redirected to login | Pass |
| UT-10 | sendNotification() | Push on new listing | Title + message | Notification delivered | Notification received | Pass |

**Table 6.5 Unit Testing**

## 6.6. Integration testing

| Test ID | Integrated Modules | Test Scenario | Expected Outcome | Actual Result | Status |
|---|---|---|---|---|---|
| IT-01 | Registration + Firebase | Store new user | Account created and saved | Account created successfully | Pass |
| IT-02 | Login + Dashboard | Load user-specific data | Dashboard loads with user info | Works as expected | Pass |
| IT-03 | Property Listing + Firestore | Add new property | Visible in search | Listing appears | Pass |
| IT-04 | Search + Filter + Firestore | Use filters | Filtered results shown | Correct filtering | Pass |
| IT-05 | View Property + Contact | Send inquiry | Contact owner possible | Message window opens | Pass |
| IT-06 | Favorites + Firestore | Save listing | Stored and accessible | Save and retrieve successful | Pass |
| IT-07 | Update Profile + Firestore | Modify user info | Changes reflected | Updated correctly | Pass |
| IT-08 | FCM Notifications | Push on new property | Notification delivered | Notification received | Pass |

**Table 6.6 Integration Testing**

## 6.7. Performance testing

| Test id | T-scena | Performance Metric | Expect res | Act res | Sta-tus |
|---|---|---|---|---|---|
| PT-o1 | App launch time | Launch Time | ≤ 3 seconds | 2.4 seconds | Pass |
| PT-02 | Login with Firebase | Response Time | ≤ 2 seconds | 1.8 seconds | Pass |
| PT-03 | Load property listings | Page Load Time | ≤ 4 seconds | 3.6 seconds | Pass |
| PT-04 | Search with filters | Query Response Time | ≤ 2 seconds | 1.5 seconds | Pass |
| PT-05 | Save to favorites | DB Write Time | ≤ 1 second | 0.8 seconds | Pass |
| PT-06 | Upload 2MB image | Upload Time | ≤ 5 seconds | 4.1 seconds | Pass |
| PT-07 | Profile update | Response Time | ≤ 2 seconds | 1.7 seconds | Pass |
| PT-08 | Push notification | Delivery Time | ≤ 4 seconds | 3.2 seconds | Pass |

**Table 6.7 Performance Testing**

## 6.8. Stress Testing

| Test ID | Test Scenario | Expected Behavior | Actual Result | Status |
|---|---|---|---|---|
| ST-01 | 1000 users login at once | Handles load with minor delay | Delay 1.5s, no crash | Pass |
| ST-02 | 500 users browsing listings | Listings load without crash | Image load delayed | Pass |
| ST-03 | 300 users saving to favorites | Data saved, no failure | Minor lag, no data loss | Pass |
| ST-04 | 100 image uploads | Queued and stored correctly | Queued uploads successful | Pass |
| ST-05 | Search queries looped | No crash, filters stay accurate | Search held up fine | Pass |
| ST-06 | Network spike while viewing | Handles retry or failure | Retry worked smoothly | Pass |
| ST-07 | Rapid tab switching | No memory leak or freeze | Smooth UI | Pass |
| ST-08 | 1000 notifications via FCM | Delivery without overload | 96% delivery success | Pass |

**Table 6.8 Stress Testing**

# Chapter 7

# Summary, Conclusion and Future Enhancements

# Chapter 7: Summary, Conclusion & Future Enhancements

## 7.1. Project Summary

Mobile application leverages Flutter framework technology to establish an effective solution that links property owners with prospective renters. This application features a solution for rental market issues by enabling property owners to outline listings with mandatory information and add location data and rental amounts and property images. The AI chatbot system provides owners assistance in property proposals and question response functions. The search functionality within the rental application grants users modern filtering tools to locate homes matching their precise property requirements including price and location. The application depends on Firebase to handle data processing alongside user authentication and provides real-time functionality and secure environments to users. The solution provides an updated rental processing system through its intuitive interface which caters to all property owners and renters.

## 7.2. Achievements and Improvements

- The system contains search filters which function correctly and property listings as well as User Authentication was integrated with Firebase Authentication.
- Improved UI/UX for better navigation.
- Users can find properties through the help of AI chatbots.
- Firestore Database Security incorporated role-based access control which stopped unauthorized modification attempts to the database.
- The solution includes scalable features which optimize Firebase queries and backend design to operate efficiently for large user numbers.
- The platform features a well-designed user interface developed with Flutter that allows seamless navigation through all its devices.
- The AI Chatbot assists landlords by giving them quick answers to their inquiries thus lowering response times.
- The system provides real-time property listings through instant updates for new property additions enabled by Firestore database synchronization.

## 7.3. Critical Review

- The project team operated against two main obstacles regarding search function optimization and achieving live data coordination.
- Security needed enhancement to establish better user authentication methods as well as transaction protection systems.
- The system required UI upgrades to solve early user interface problems with navigation.
- Optimization of the system for high-traffic use enabled effortless property navigation.
- The resolution of security matters required more development on role-based access control together with Firestore rules for authorized control objectives.
- The system faced search optimization problems because property filtering and searching required more time at first. Optimized Firestore queries improved performance.

## 7.4. Lessons Learnt

- The design approach of a user interface and user experience directly affects how users become involved with a system.
- Security must implement both authentications together with role-based access control systems.
- The system needs performance optimization to manage big user requests properly.
- Long-term success requires a system which includes future growth considerations during development.
- The collected user feedback played a crucial role in leading designers to enhance their products by making key design and feature modifications.
- The response time benefits from optimized queries because appropriate Firestore indexing together with structured API calls.
- Security requires absolute priority therefore implementing authentication as well as access control and protecting data.
- A user-friendly interface paired with a great user experience design directly affects how users feel about the system and their wish to use it again.

## 7.5. Future Enhancements/Recommendations

- Integration of online payment gateways.
- Advanced AI-based rental recommendations.
- Multi-language support (English, Urdu).
- Desktop/web version for wider accessibility.
- The system implements AI to forecast prices according to market developments and user needs.
- The recommendation engine of the chatbot will receive an upgrade to propose accommodations that fit user preferences.

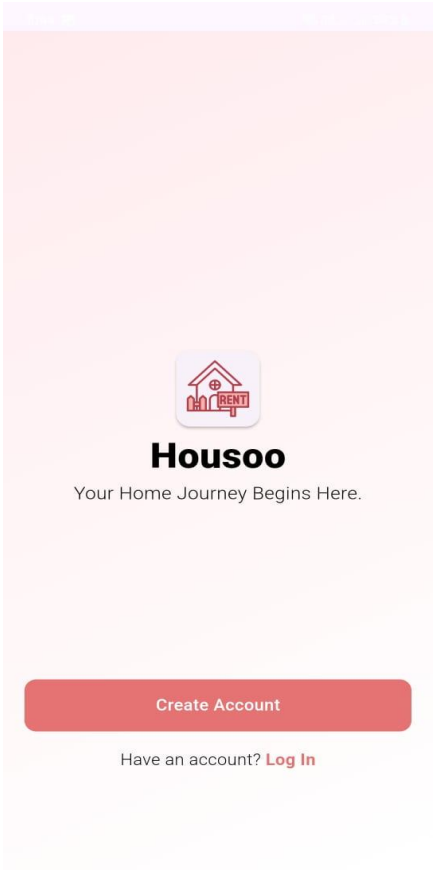
# Appendices

# **Appendix A:** User Manual

The application supplement describes step-by-step instructions for Home Rentals App users to explore and utilize its features. Users can find instructions for registering as well as procedures for searching properties, listing properties and advanced filter usage in this section. An appendix serves to streamline user operations while guaranteeing fluid communication between users and the platform.

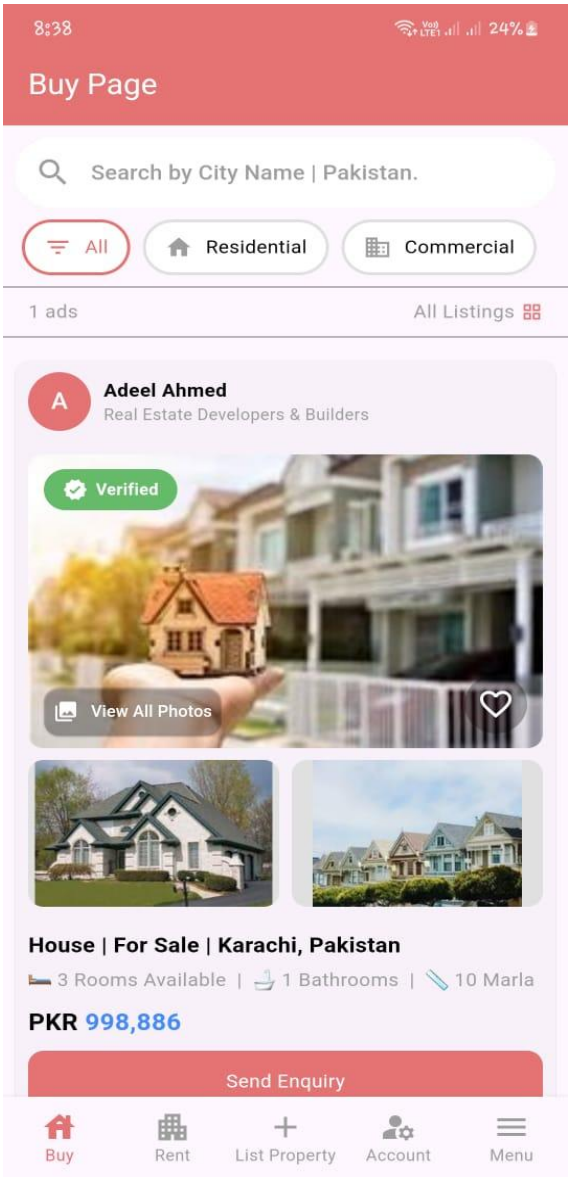
## A.1. Getting Started

- Users can obtain the Home Rentals App through Play Store
- Must be Register choosing between a new account registration with email and password or proceed through the existing login system.



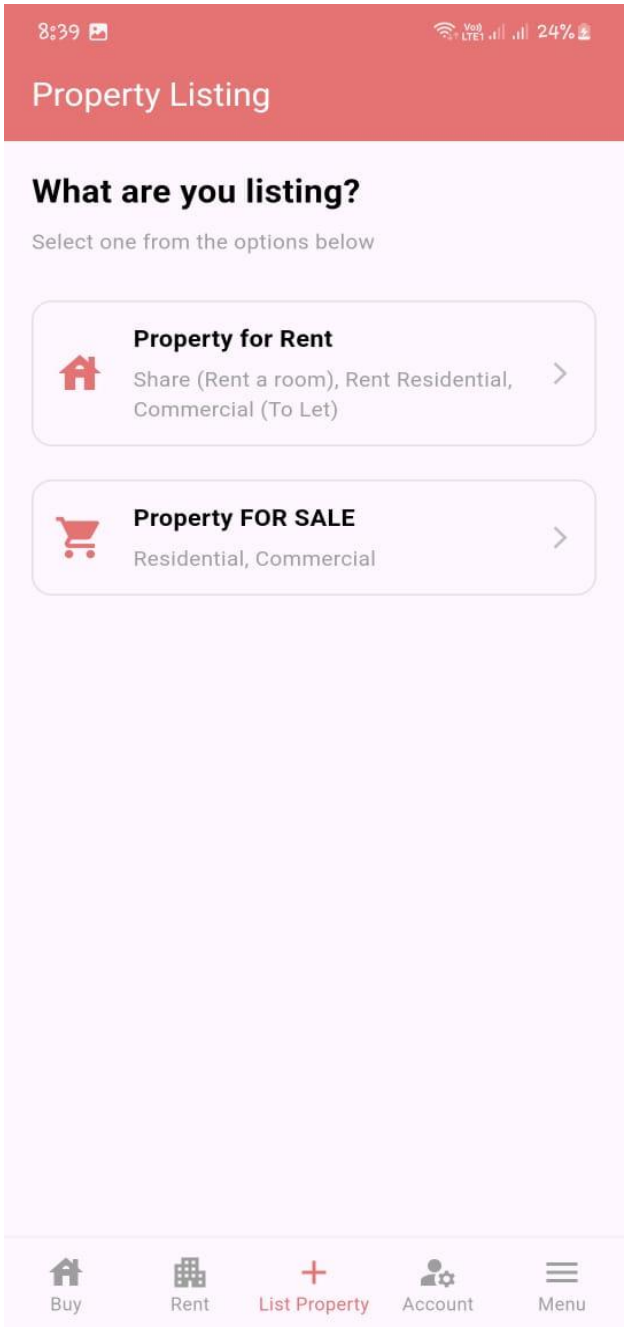
### A.1.1. Home Screen Navigation

- The dashboard interfaces show properties that match user preference criteria.
- The search bar enables users to find properties through location restrictions and price ranges and amenity selection.



### A.1.1.1. Viewing a Property Listing

- Users can view complete property descriptions together with pictures and contact information by selecting any available listing.
- The chat feature enables users to contact their landlords through the system.

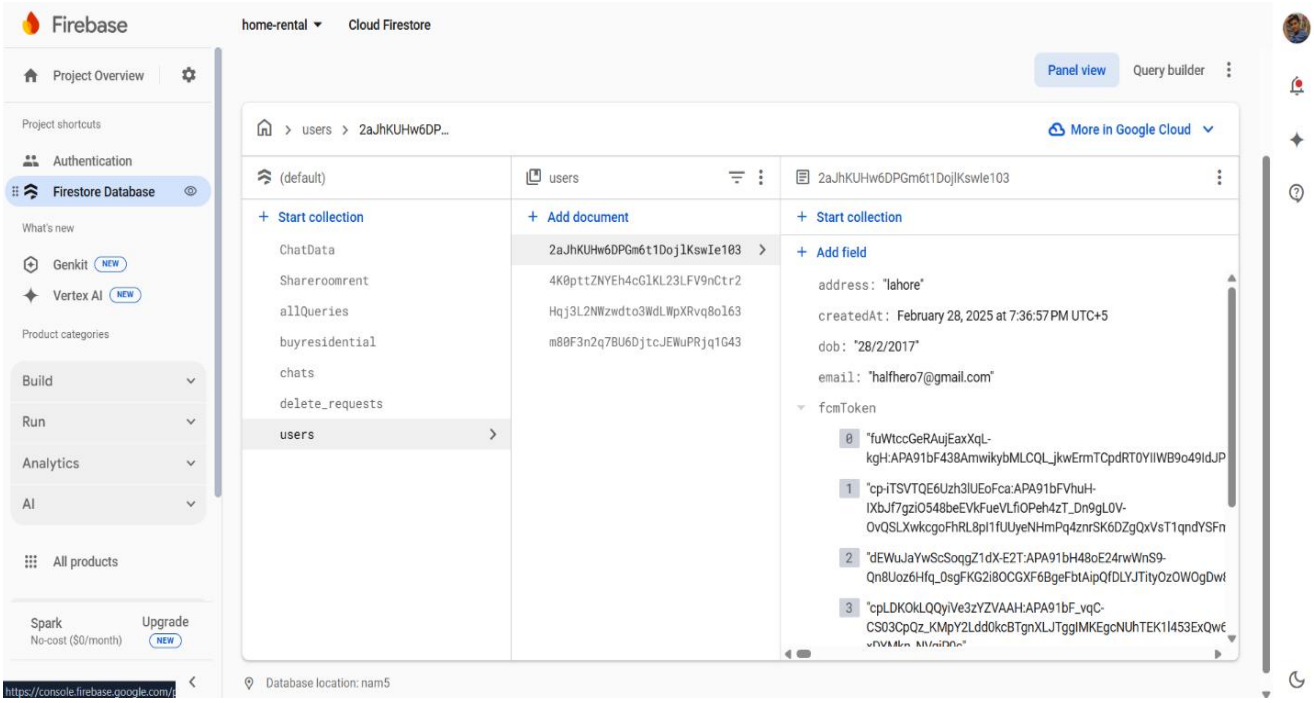


# **Appendix B:** Administrator Manual

The appendix serves as a reference for platform administration control. This appendix contains guidelines that cover user role controls separate from property approval steps as well as database protection strategies.

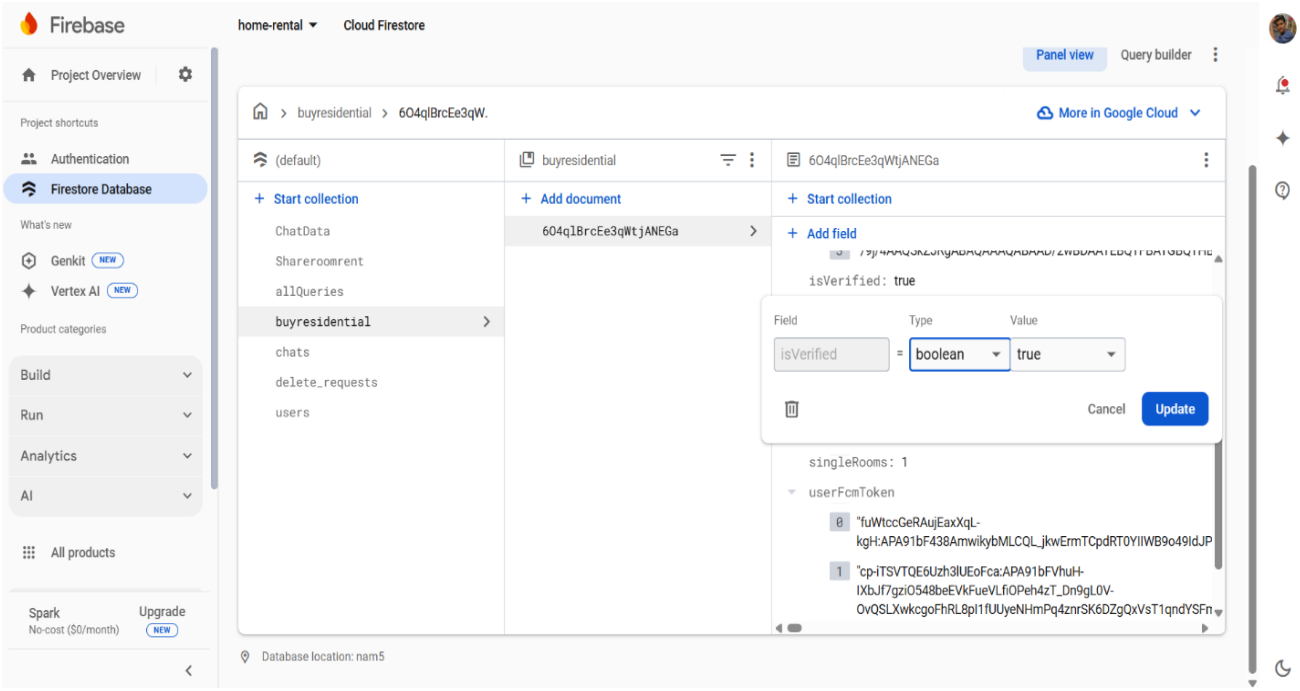
## B.1. Managing Users

- High-level administrators can access user profiles together with activity logs and flagged accounts through the system.
- The administrator can authorize or deny user access through compliance policy parameters.



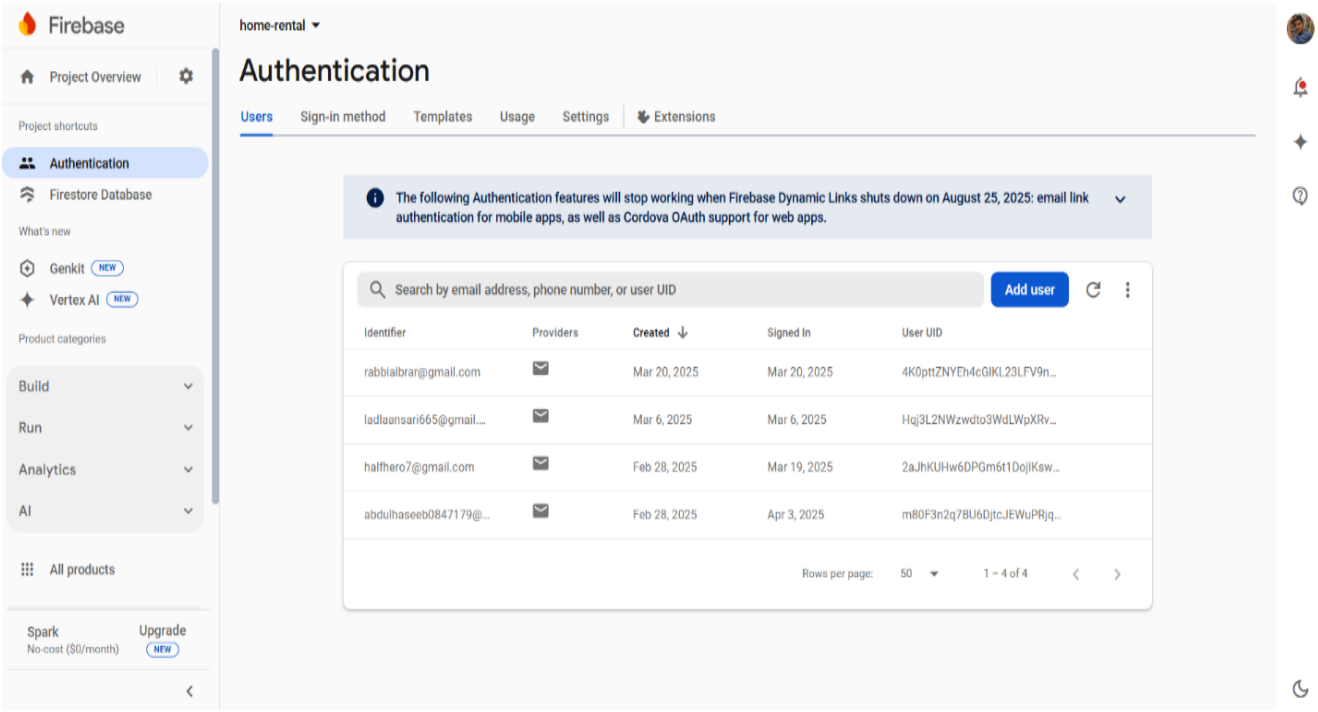
### B.1.1. Property Approval Process

- The collier must inspect new listings then validate property specifications before any information goes public.



### B.1.1.1. Handling Reported Authentications

- Users reporting fake accounts or suspicious logins
- The admin at Seedily uses their authority to handle flagged users who commit fraud or send spam.
- Tracking login history and identifying authentication anomalies should be included as a surveillance practice.
- The organization denies entry for user accounts being actively investigated.



# Reference and Bibliography

# Reference and Bibliography

## Research Papers & Academic References

[1] K. Smith, A. Jones (2023). "The Impact of Mobile Applications on the Real Estate Industry," International Journal of Digital Transformation in Real Estate, Vol. 22, Issue 3, pp. 120-135.

[2] Massi, M., Piancatelli, C., & Vocino, A. (2023). "Authentic Omnichannel: Providing Consumers with a Seamless Brand Experience through Authenticity," Psychology & Marketing, 40(7), pp. 1263–1445.

🔗 Click here

[3] Gusti Noorlitaria Achmad et al. (2023). "The Impact of Social Media on Online Shopping Behavior of Gen Z Consumers In Time of Covid-19 Pandemic," WSEAS Transactions on Business and Economics, 21, pp. 266–279.

🔗 Click here

[4] Lisun, Y. (2024). "The Role of Social Networks in Shaping Consumer Trends and Developing the Advertising Industry," Economic Affairs, 69(1), pp. 10–10.

🔗 Click here

[5] Samuel Hudson Mrisha & Xixiang, S. (2024). "The Power of Influence: How Social Media Influencers Are Shaping Consumer Decision Making in the Digital Age," Journal of Consumer Behavior, 23(4).

🔗 Click here

[6] Sharma, A. (2023). "Mobile Payments and Digital Wallets: Changing Consumer Behavior," International Journal of Social Science & Economic Research, 8(9), pp. 2891–2895.

🔗 Click here

[7] Suvattanadilok, M. (2020). "Factors Influencing Consumer Behavior's via Web Personalization and Information Content on Social Media," African Journal of Hospitality, Tourism and Leisure, 9(1).

🔗 Click here

[8] Singh, P. et al. (2024). "How Information Technology (IT) Is Shaping Consumer Behavior in the Digital Age: A Systematic Review and Future Research Directions," Sustainability, 16(4), p. 1556.

🔗 Click here

[9] T. Brown, M. Green (2022). "AI-Powered Chatbots for Real Estate Transactions: A Case Study," Journal of Artificial Intelligence & Consumer Technology, Vol. 17, No. 2, pp. 90-112.

🔗 Click here

[10] M. Ahmed, L. Zhao (2023). "Cloud-Based Database Management for Scalable Real Estate Applications," IEEE Transactions on Cloud Computing, Vol. 15, No. 4, pp. 210-225.

🔗 Click here

[11] S. Williams, P. Oliver (2023). "Enhancing User Experience in Rental Apps Using Augmented Reality," International Conference on Human-Computer Interaction.

🔗 Click here

[12] **M. Ahmed, L. Zhao**, "Cloud-Based Database Management for Scalable Real Estate Applications," *IEEE Transactions on Cloud Computing*, Vol. 15, No. 4, pp. 210-225, 2023. LinkIEEE Computer Society

[13] **S. Williams, P. Oliver**, "Enhancing User Experience in Rental Apps Using Augmented Reality," *International Conference on Human-Computer Interaction*, 2023. LinkResearchGate

[14] **S. H. Mrisha, S. Xixiang**, "The Power of Influence: Social Media Influencers and Consumer Decision Making," *Journal of Consumer Behavior*, Vol. 23, No. 4, 2024. LinkWiley Online Library+1Wiley Online Library+1

[15] **A. Sharma**, "Mobile Payments and Digital Wallets: Changing Consumer Behavior," *International Journal of Social Science & Economic Research*, Vol. 8, No. 9, pp. 2891–2895, 2023. Linkijsser.org

[16] **P. Singh et al.**, "How Information Technology Is Shaping Consumer Behavior," *Sustainability*, Vol. 16, No. 4, p. 1556, 2024. Link

[17] **M. Suvattanadilok**, "Factors Influencing Consumer Behavior's via Web Personalization," *African Journal of Hospitality, Tourism and Leisure*, Vol. 9, No. 1, 2020. Link

## Technical Documentation & Online Resources


1. **Google Developers** (2024). *Firebase Authentication & Firestore Database: Secure Cloud-Based Solutions*, Google Firebase Documentation.
   Available at: https://firebase.google.com/docs
2. **Flutter Official Documentation** (2024). *Building Scalable Mobile Apps with Flutter*, Flutter.dev.
   Available at: https://flutter.dev/docs
3. **OpenAI** (2024). *AI-Powered Chatbot Integration in Mobile Applications*, OpenAI Developer Guide.
   Available at: https://openai.com/docs
4. **Firebase Cloud Messaging (FCM)** (2024). Users can access the documentation about implementing Push Notifications in Mobile Applications through Firebase Documentation.
   Available at: https://firebase.google.com/docs/cloud-messaging
5. **Google Developers** (2024). The documentation in Firebase Security contains information about securing mobile applications through their security rules.
   Available at: https://firebase.google.com/docs/rules
6. **Flutter Fire Documentation** (2024). Flutter Fire provides directions to integrate Firebase services into Flutter development projects.
   Available at: https://firebase.flutter.dev
7. **Google Developers** (2024). *Firebase Performance Monitoring: Enhancing App Performance*, Firebase Documentation.
   Available at: https://firebase.google.com/docs/perf-mon
8. **OpenAI Platform** (2024). *API Reference: ChatGPT Integration*, OpenAI API Documentation.
   Available at: https://platform.openai.com/docs/api-reference

# Index

# Index

**[A]**

- AI Chatbot – User assistance feature in the app.
- User login security is achieved through the implementation of Firebase Auth authentication system.
- A virtual home tour system is planned for implementation as Augmented Reality under future development (Future Enhancement).

**[B]**

- Firebase operates the Backend functionality to handle user information and listing data together with chat communications.
- Brochure – Marketing material for app promotion.
- The service allows tenants to use a booking system for rental requests.

**[C]**

- Property details are stored within the firebase database part of cloud storage.
- Chatbot – AI-powered assistant for user queries.
- A help hub operates as the customer support platform which assists both tenants and landlords.