INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT, AKURDI, PUNE

“ EasyHome ”
Property Rental System

PG-DAC August 2024

Submitted By:

Group No: 86

Roll No	**Name of Student**
248180	Prathamesh Chavan
248168	Shubham Nigave

Mr. Vaibhav Verulkar
Project Guide

Mr. Rohit Puranik
Centre Coordinator

ABSTRACT

The "Property Rental System" is a comprehensive platform designed to integrate property rental management with a seamless user experience. It enables tenants to easily browse and book rental properties while allowing landlords to manage and list their available units with ease. The platform incorporates advanced technologies to streamline property leasing, ensuring an efficient process for both tenants and property owners.

Built with React for a responsive front-end and Spring Boot for a robust backend API, the system provides a smooth and intuitive interface. Security is enhanced through Spring Security for user authentication and authorization, ensuring that sensitive data is protected. Stripe payment integration securely processes tenant payments once their booking requests are approved.

The Property Rental System is designed to address the challenges faced by both renters and property owners by offering an integrated platform that simplifies property searching, booking, management, and payments, making the rental process more efficient and reliable for everyone involved.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to everyone who has been a part of this journey. Our heartfelt thanks go to **Mr. Vaibhav Verulkar**, our esteemed guide, whose invaluable advice and support at crucial stages of the project have been instrumental in steering us in the right direction. We are also grateful to our respected Centre Co-ordinator, **Mr. Rohit Puranik**, for providing us with the necessary resources and facilities. We would like to extend our appreciation to all the faculty members for their assistance throughout this process. Lastly, we must thank our friends and family for their unwavering encouragement and support, which has been a constant source of motivation during the course of this project.

Prathamesh Chavan (240841220126)
Shubham Nigave (240841220107)

Table of Contents

1. INTRODUCTION

EasyHome : **Property Rental System** is a comprehensive platform designed to simplify the process of renting properties while facilitating easy management for both tenants and landlords. The platform allows tenants to easily find and book rental properties, while landlords can efficiently manage their listings, approve/reject bookings, and handle property details. The system integrates secure payment processing through Stripe and provides a seamless user experience for both tenants and landlords.

The platform uses React for the responsive and dynamic front-end, while Spring Boot powers the robust back-end API. Spring Security is used to ensure secure authentication and authorization, while payments are securely processed via Stripe.

The Property Rental System is designed to cater to tenants, landlords, and admins, each of whom has specific responsibilities, permissions, and access to the system's features. This ensures smooth operation and secure management of properties, bookings, payments, and user interactions.

Roles and Responsibilities

1. Admin Role

The Admin role is the highest level of access within the EasyHome - Property Rental System. Admins can do User Management, Property Management, Booking Management, Payment Management, Feedback & Reports.

The role provides comprehensive control over the platform's settings, security policies, and content management.

Responsibilities:

➢ **User Management:**

Create, manage, and delete user accounts (both tenants and landlords).

Assign roles and permissions to users.

Monitor user activities to ensure compliance.

➢ **Property Management:**

View, update, and delete property listings.

Manage property details, including availability and amenities.

➢ **Booking Management:**

View and manage all bookings made by tenants.

Approve or reject booking requests based on landlord feedback.

➢ **Payment Management:**

Oversee all payment transactions, resolve payment issues, and issue refunds when necessary.

➢ **Feedback & Reports:**

Generate and analyze reports on bookings, revenue, and property performance.

Use data to assist in decision-making for business strategy.

➢ **Access Level:**

Full access to all system features.

Ability to manage users, properties, and bookings.

Access to sensitive financial data, transaction history, and user details.

2.<u>Tenant Role</u>

Tenants are the primary users of the Property Rental System, using the platform to browse properties, submit booking requests, and make payments. Their access is focused on property browsing, booking management, and payment processing.

Responsibilities:

- **<u>View Properties:</u>**
  Browse available rental properties, including details on price, location, and amenities.
- **<u>Book Properties:</u>**
  Submit booking requests for properties, providing rental duration and preferences.
- **<u>Profile Management:</u>**
  Update personal information (name, contact details, address, etc.).
- **<u>Payments:</u>**
  Make payments through Stripe upon booking approval.
  View payment history and manage payment details.
- **<u>Support Requests:</u>**
  Raise support tickets for any issues related to bookings or property inquiries.
- **<u>Access Level:</u>**
  Limited access to view properties, submit booking requests, and make payments.
  No access to property management, user roles, or system configurations.

<u>The Role-Based Access Control (RBAC) and Security</u>

The Property Rental System implements Role-Based Access Control (RBAC) using Spring Security along with JWT (JSON Web Tokens) to ensure secure and role-specific access to the platform's resources. RBAC ensures that each user has access only to the functionalities they need based on their assigned role. This approach ensures that the system is secure, with users being able to perform only the actions allowed by their role, and sensitive data is protected.

Roles in the System:

Admin: Has full access to all system functionalities, including user and property management, booking approvals, and generating reports.

Tenant: Can view and book available properties, make payments, and manage their personal information.

Landlord: Can list properties, manage property details, and approve/reject tenant bookings.

RBAC Implementation:

Admins have unrestricted access to all platform resources. They can manage users, properties, bookings, and generate reports. They also have the ability to configure system-wide settings.

Tenants are restricted to viewing properties, submitting booking requests, and managing their own profiles.

Landlords can add, update, and delete their properties, and approve/reject booking requests. They cannot access the functionalities available to admins or tenants.

Security Considerations

The Property Rental System employs robust security mechanisms to ensure the safety and integrity of user data and actions. Security features include JWT Token Authentication, Spring Security for authorization, and secure session management. These mechanisms ensure that each user is only able to access the resources they are permitted to interact with, depending on their role.

Key Security Features:

Authentication with JWT Tokens:

When a user logs in (whether a tenant, landlord, or admin), they receive a JWT Token which they must include in the request headers for subsequent interactions with the system.

This token is used for authenticating the user's identity and ensures secure communication between the client and the server.

JWT tokens are signed and contain user role information, allowing the back-end to verify both the identity and role of the user.

Authorization:

The system checks the user's role (extracted from the JWT token) before granting access to certain resources or actions.

Admin users can perform all operations, while Landlords and Tenants have access limited to their

respective areas of the system (e.g., Landlords can manage properties, but cannot manage users).

Session Management:

JWT tokens are stateless, meaning no session data is stored server-side, reducing the risk of session-based attacks (like session hijacking).

Tokens are set to expire after a predefined period to ensure that expired tokens cannot be used to gain unauthorized access.

The integration of JWT Token Authentication ensures that only authorized users can access the system, and their actions are restricted based on their role. This, in combination with Spring Security, ensures that the platform is secure, and sensitive data (such as payment details or personal information) remains protected.

1.1 Purpose

"EasyHome - Property Rental System," aims to simplify the process of renting properties. We're developing a user-friendly platform where tenants can easily find and apply for rental properties, while landlords and property managers can seamlessly list available units. This web application reduces manual effort, streamlining the rental process for both renters and property owners, making property searching and leasing more efficient for everyone involved.

1.2 Scope

1. Real-time Property Search & Recommendations
2. Booking Management & Notifications
3. Payment Integration & Automation
4. Property Management & Admin Controls
5. Report Generation & Analytics
6. Automation of Booking & Payment Reminders
7. Tenant and Landlord Mobile App Integration
8. Property Availability & Calendar Synchronization
9. Security Enhancements

1.3 Objective Of EasyHome - Property Rental System

The objectives of the Property Rental System project outline the primary goals that the system aims to achieve, providing clear guidance on its design, implementation, and operational focus. These objectives ensure that the system effectively supports the needs of tenants, landlords, and admins, while driving operational efficiency and growth in the property rental business.

1. Efficient Property Management
2. Seamless Booking Process
3. Robust Security and Access Control (including JWT Token Authentication and RBAC)
4. Scalable and Flexible Architecture
5. Comprehensive Reporting and Analytics
6. Enhanced User Experience
7. Payment Integration and Security (using Stripe)
8. Compliance and Audit Readiness

1.4 Functionalities Provided by Property Rental System

User Management:-

➢ **User Registration and Login:**

Users can create accounts, log in, and manage their profiles.

Secure authentication and authorization using JWT Tokens and Spring Security.

<u>Role-Based Access Control:</u>

Different user roles (e.g., Tenant, Landlord, Admin) with specific permissions.

Admins can create and manage roles and assign them to users.

<u>Profile Management:</u>

Users can update their personal information (name, email, contact details).

Password management, including options for changing and resetting passwords.

➢ **Property Management:**

<u>Property Listing:</u>

Landlords can add, update, and delete property listings.

Manage property details like name, address, rent, description, images, and amenities.

Property Availability:

Landlords can set availability for properties and manage booking statuses.

Property Search:

Tenants can search for properties based on location, price, and amenities.

➢ **Booking Management:**

Booking Request:

Tenants can book properties after selecting them and providing required details.

Booking Approval/Decline:

Landlords can approve or reject booking requests.

Payment Integration:

Integration with Stripe for secure online payments.

Tenants can make payments after booking approval.

Booking Status:

Tenants can track the status of their booking (pending, approved, or rejected).

➢ **Admin Management**

Manage Users:

Admins can view, add, and manage user profiles (Tenants, Landlords).

Manage Properties:

Admins can oversee all property listings and manage property approval processes.

Manage Bookings:

Admins can view, approve, or reject bookings and track overall booking status.

Generate Reports:

Admins can generate reports on bookings, payments, and property statistics.

➢ **Search and Filters:**

Tenants can search and filter properties based on criteria like location, price, and amenities.

Responsive Design:

A responsive and user-friendly interface.

Booking History:

Tenants can view their booking history and manage past bookings.

2. SOFTWARE REQUIREMENT SPECIFICATION

EasyHome Property Rental System outlines the functional and non-functional requirements essential for the successful design, implementation, and operation of the platform. These requirements focus on providing a seamless, secure, and user-friendly experience for tenants, landlords, and admins.

2.1 Functional Requirements for EasyHome: Property Rental System

1. User Management

- **User Registration:**

The system shall allow new users (tenants and landlords) to create an account by providing necessary details such as name, email, and password.

- **User Authentication:**

The system shall authenticate users during login using their registered email and password. It will use JWT tokens for secure, sessionless authentication.

- **Role-Based Access Control:**

The system shall support multiple user roles, including **Admin**, **Tenant**, and **Landlord**, each with different levels of access and permissions.

- **Profile Management:**

Tenants can update personal details and rental preferences, while landlords can manage property details and listings.

2. Property Management

- **Property Listings:**

Landlords shall have the ability to add, update, and delete property listings. Each property will have details such as name, location, description, amenities, rent, and images..

- **Property Availability:**

Landlords can manage the availability of their properties by setting available dates and booking statuses. This helps avoid double bookings.

- **Property Search:**

Tenants can search for rental properties by criteria such as location, price range, and amenities.

3. Booking Management

- **Booking Requests:**

Tenants shall be able to submit booking requests after selecting a property.

- **Booking Approval/Decline:**

Landlords can approve or reject booking requests based on availability and personal discretion. Upon approval, tenants are notified, and booking statuses are updated.

- **Payment Integration:**

The system shall integrate with **Stripe** to handle secure online payments. Tenants can complete payments for confirmed bookings, and landlords will be notified of payment status.

- **Booking Status Tracking:**

Tenants shall be able to track the status of their bookings (pending, confirmed, rejected) and view any relevant updates.

4. Admin Management:

- **User Management:**

  Admins shall be able to view, create, and manage user accounts (both tenants and landlords). Admins can assign roles to users based on their responsibilities.

- **Property Management:**
  Admins can oversee all property listings on the platform and review property approval processes. Admins have the authority to approve or reject new property listings submitted by landlords.

- **Booking Management:**
  Admins can manage bookings at the system level, including reviewing, approving, and rejecting bookings in case of conflicts or issues.

- **Reports and Anayltics:**
  Admins can generate comprehensive reports on bookings to assist with strategic decisions.

2.2 Non-Functional Requirements for EasyHome : Property Rental System

1. Performance

- Response Time:

The system shall respond to user actions, such as property search and booking requests, within minimum time under normal operating conditions.

- Scalability:

The system should be designed to support a growing number of users and properties without performance degradation. It should be able to scale to support thousands of concurrent users.

- Throughput:

The system shall process at least 100 transactions per second during peak usage times.

2. Reliability

- Availability:

The system shall have an uptime of 99.9% over a 12-month period, ensuring high availability for users.

- Fault Tolerance:

In case of hardware or software failure, the system shall continue to function with minimal disruptions, ensuring that users experience minimal downtime.

- Error Handling:

The system shall provide clear error messages and gracefully handle unexpected situations. Users should receive helpful feedback when issues occur.

3. Usability

- User Interface:

The system shall have a user-friendly interface that is easy to navigate, with clear instructions and minimal learning curve.

4. Maintainability

- Modularity:

The system shall be designed in a modular fashion, allowing for easy updates and enhancements to individual components without affecting the entire system.

- Code Quality:

The system shall follow coding best practices, with well-documented, clean, and
maintainable code.

- Testing:

The system shall undergo comprehensive testing, including unit testing, integration testing, and user acceptance testing (UAT) to ensure its quality and stability.

● Other Requirements:

Hardware and Network Interfaces:
Back-end Server Configuration:

- Intel Pentium-IV Processor
- 8 GB RAM
- 500 GB Hard Drive (minimum)
- High-Speed Internet Connection (to handle multiple concurrent users and data transactions)

Software Interfaces:

Software configuration for back-end Services:

- Java EE
- Spring Boot, JPA, Stripe, Spring Authentication
- MySQL 8.0
- Spring Tool Suit 4

Software configuration for front-end Services:

- ReactJS, VS Code
- HTML 5, CSS 3, JS
- Bootstrap 5

Future Scope

The Property Rental System offers real-time property search with personalized recommendations, streamlined booking management with automated notifications, and secure payment integration. It features comprehensive property management, admin controls, and advanced report generation with analytics. The system also automates booking and payment reminders, integrates tenant and landlord mobile apps, synchronizes property availability with calendar updates, and ensures robust security enhancements.

3. DIAGRAMS

3.1 **Entity Relationship Diagram**:

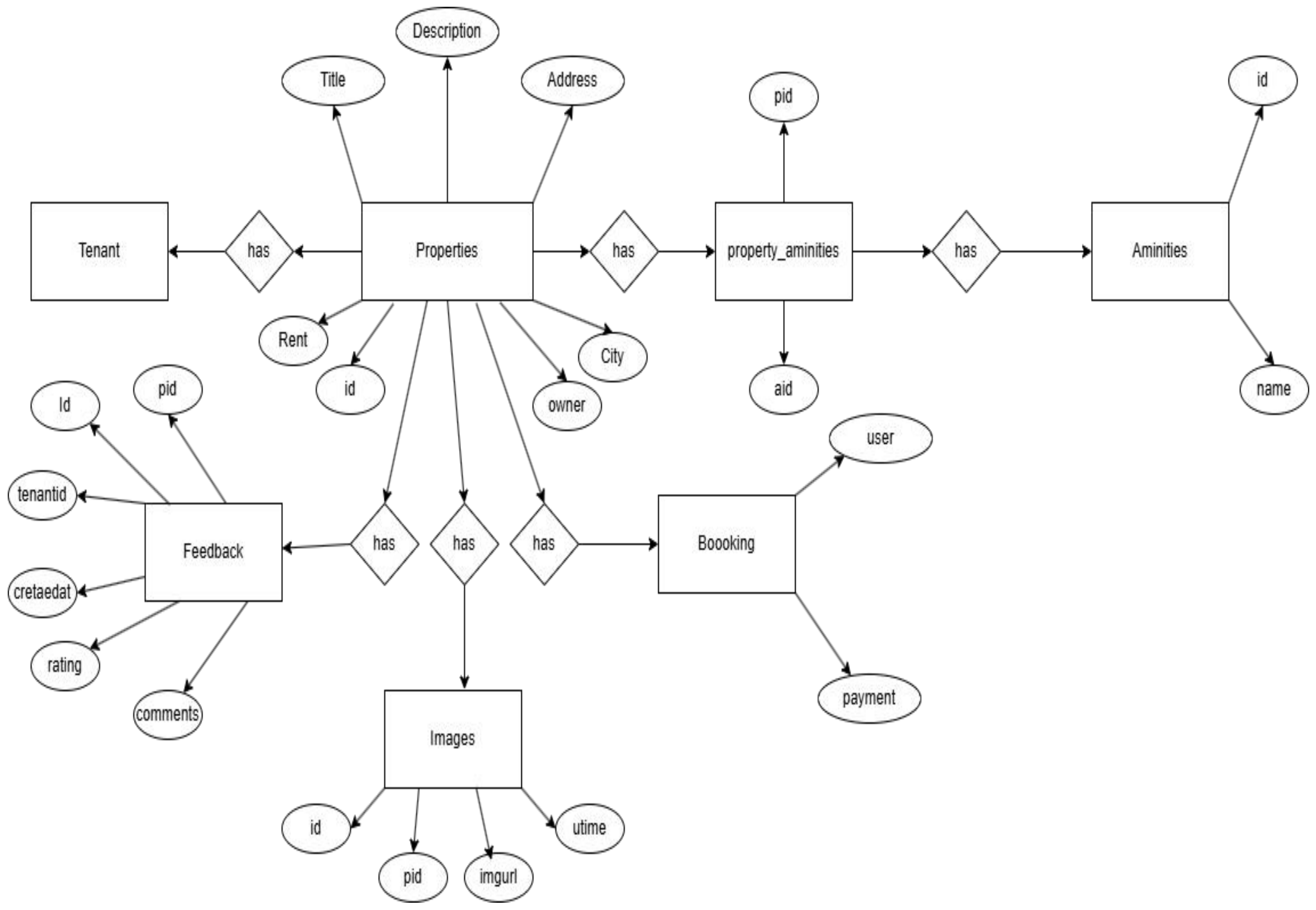


Fig. ER Diagram for EasyHome

3.2 **Use Case Diagram**:

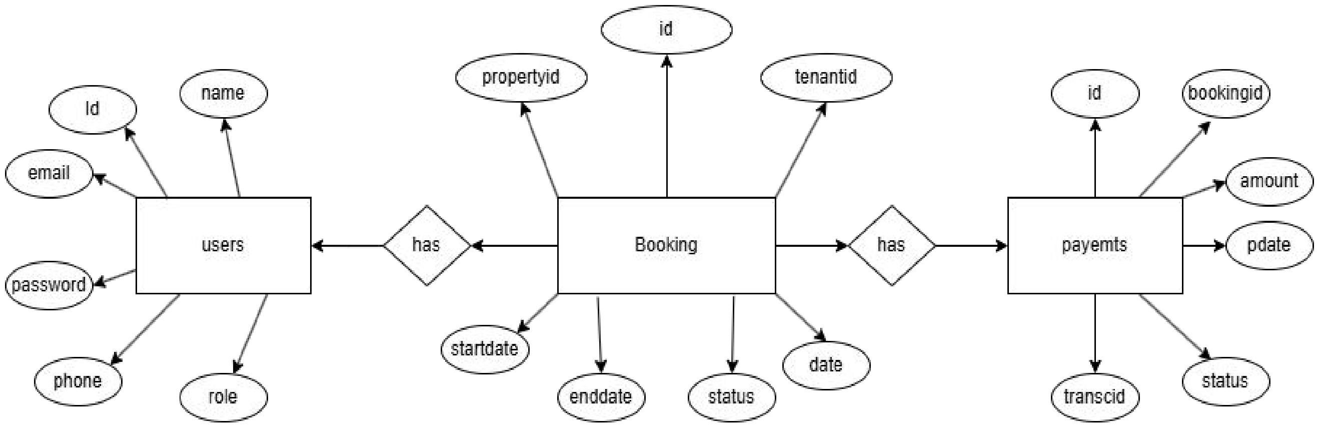


Fig.Booking ER

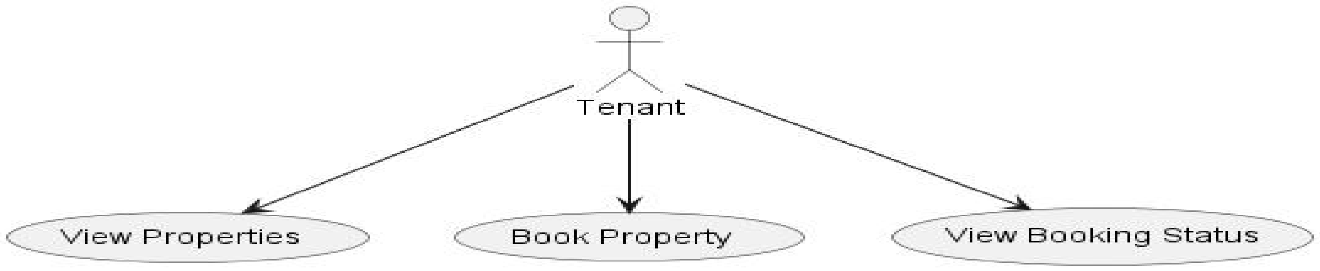


Fig.Admin UseCase

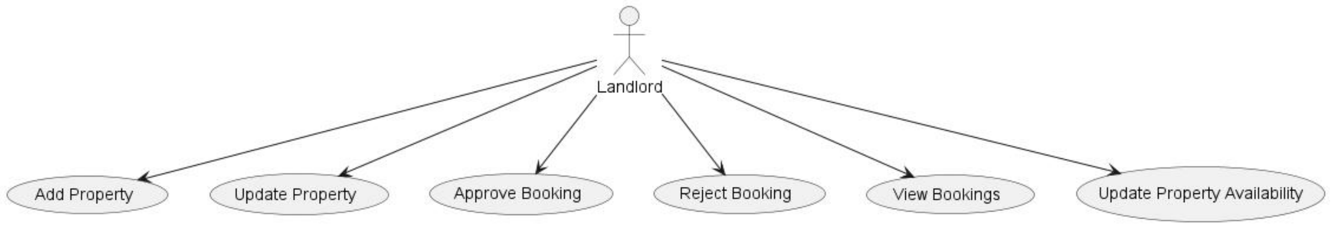


Fig.LandLord UseCase

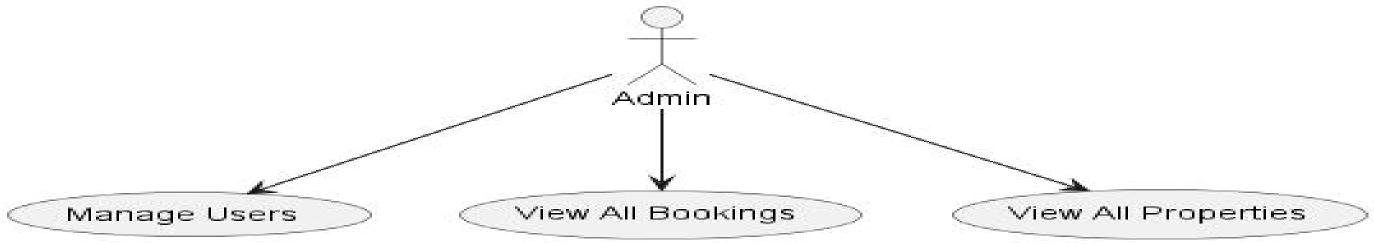
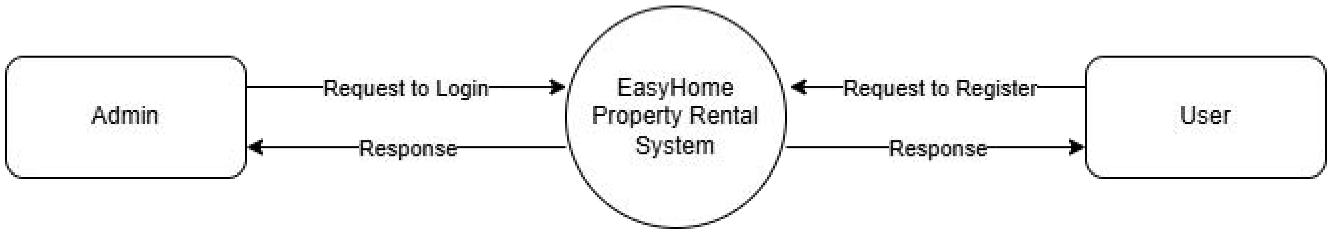


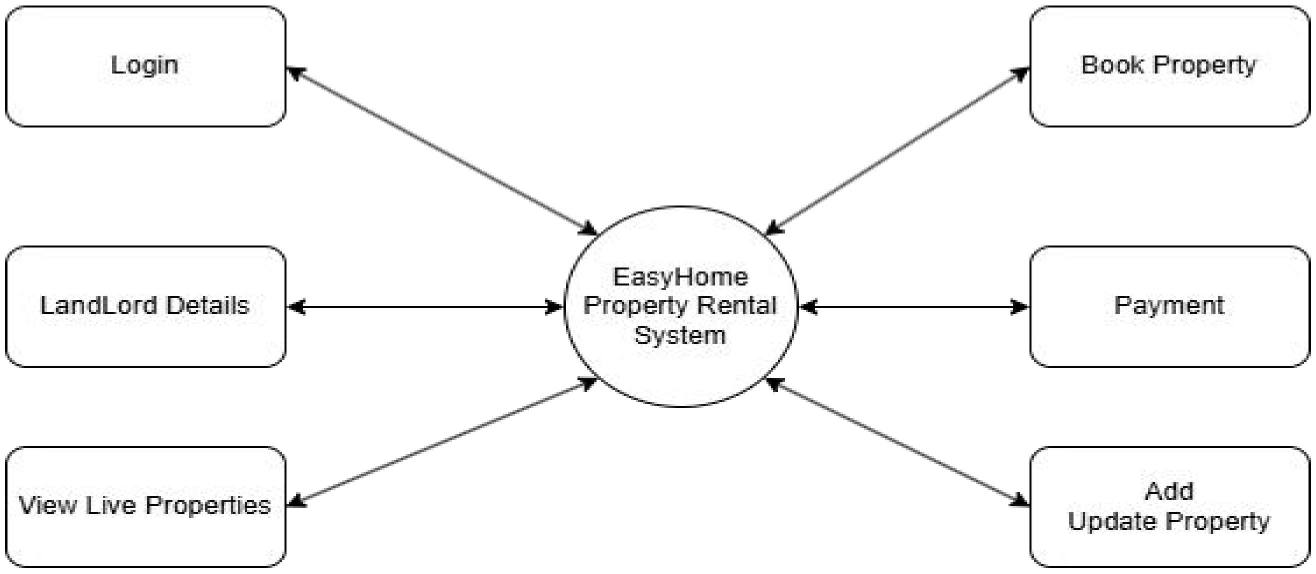
Fig.Tenant UseCase

3.3 <u>Data Flow Diagram</u>

DFD Level 0:

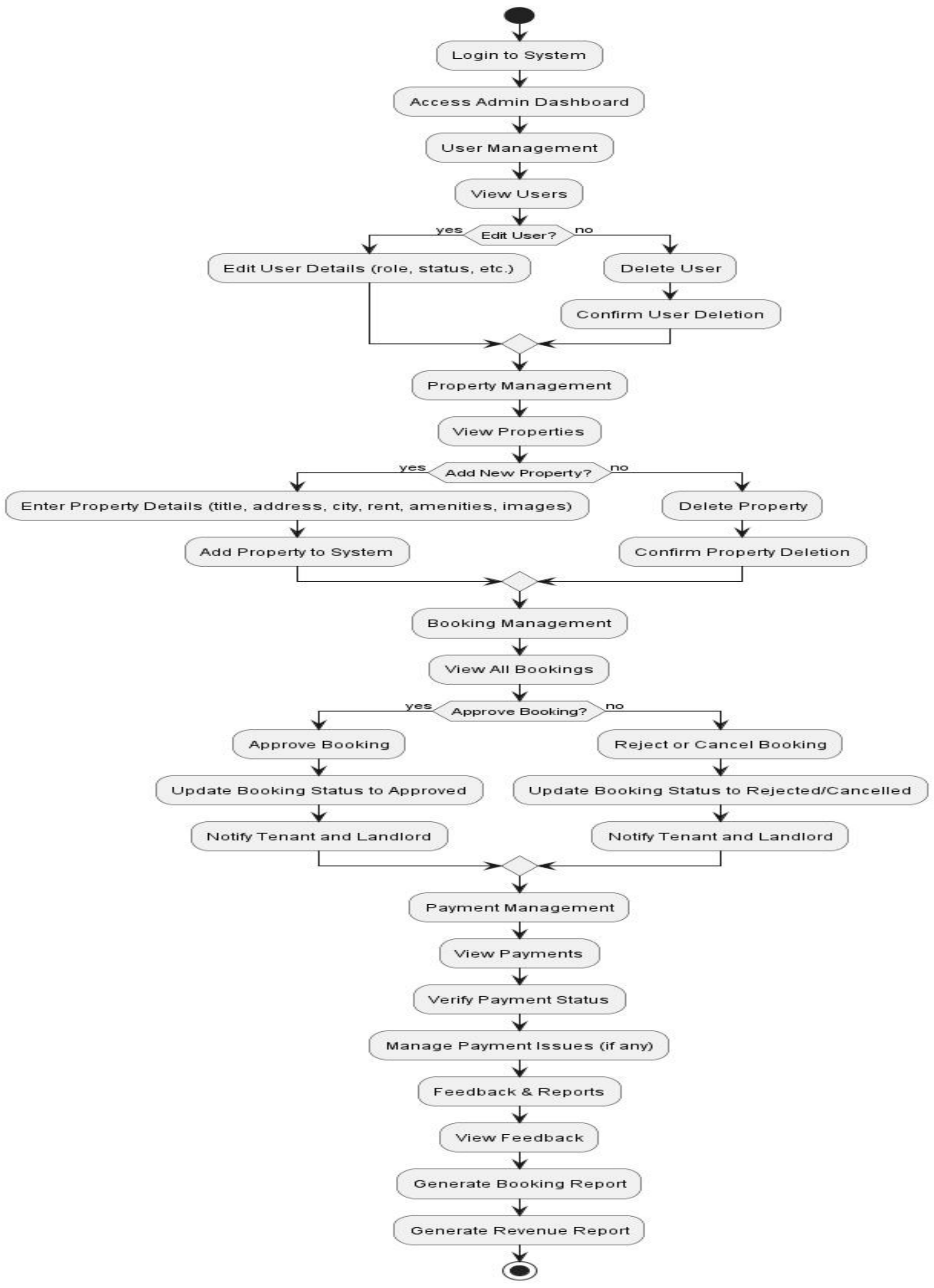


DFD level 1 :

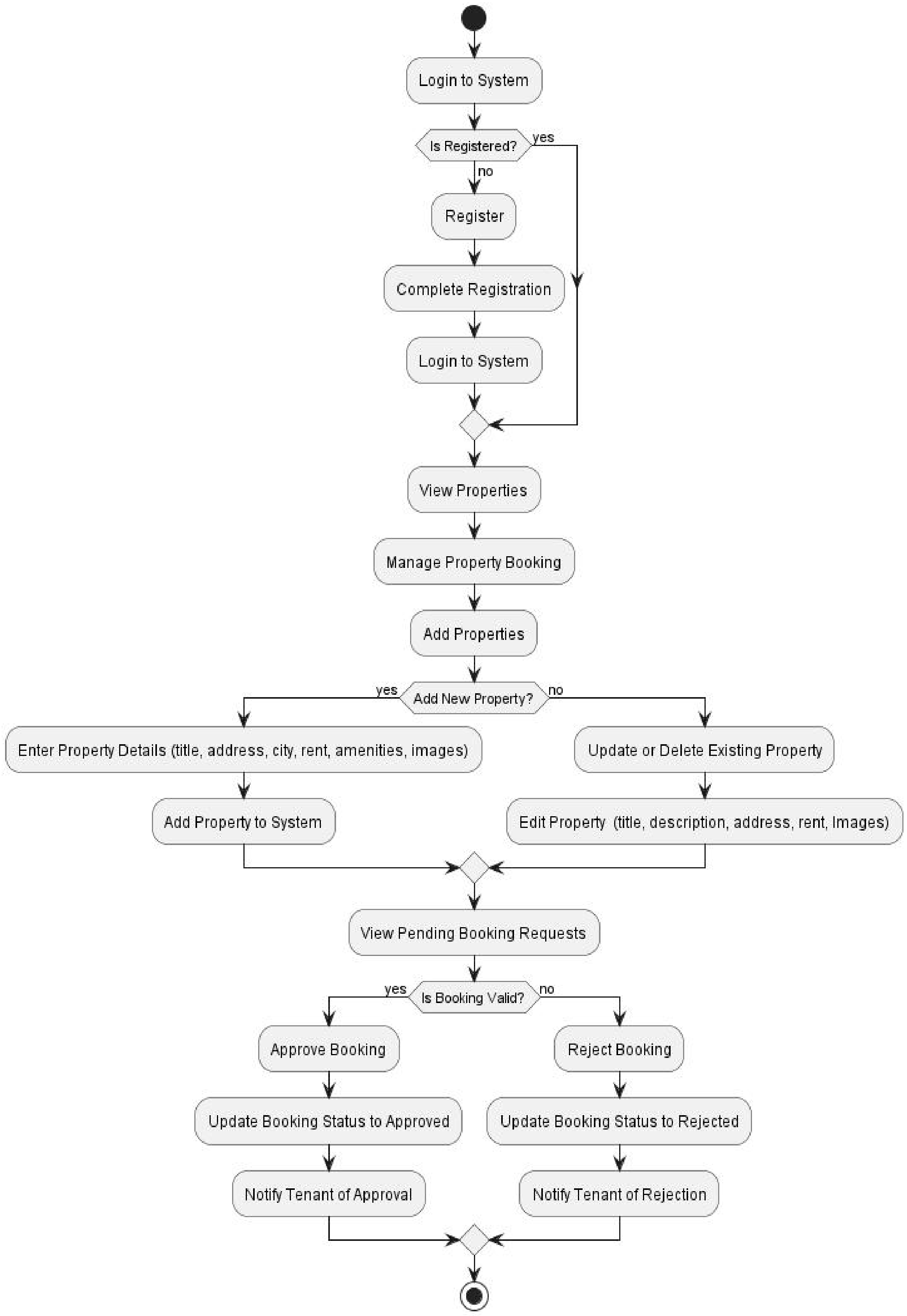


3.4 Activity Diagram

1. **Admin Activity Diagram**

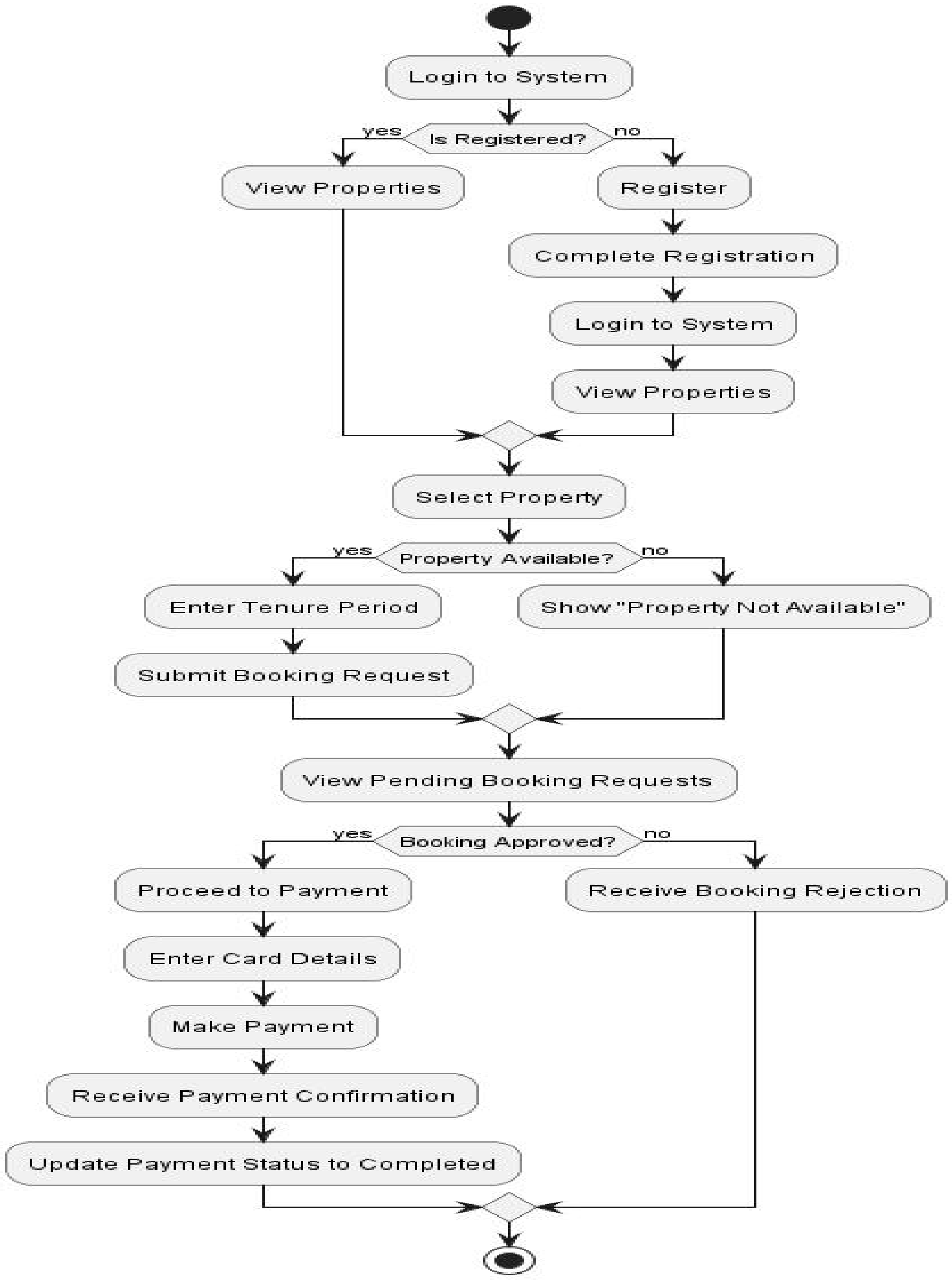


2. Landlord Activity

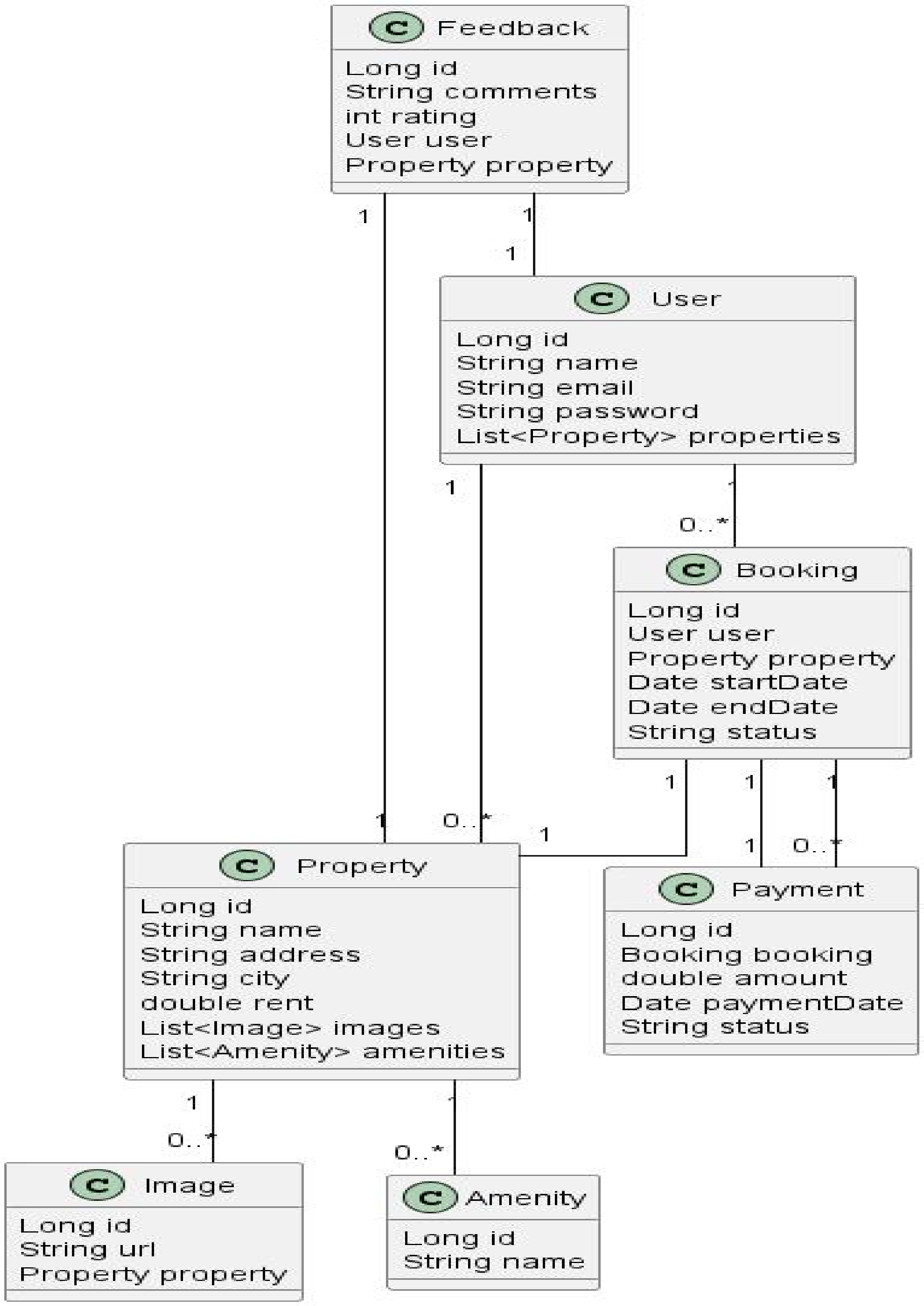

Login to System
Is Registered?
yes
no
Register
Complete Registration
Login to System
View Properties
Manage Property Booking
Add Properties
Add New Property?
yes
no
Enter Property Details (title, address, city, rent, amenities, images)
Update or Delete Existing Property
Add Property to System
Edit Property (title, description, address, rent, Images)
View Pending Booking Requests
Is Booking Valid?
yes
no
Approve Booking
Reject Booking
Update Booking Status to Approved
Update Booking Status to Rejected
Notify Tenant of Approval
Notify Tenant of Rejection

:

3. **Tenant Activity Diagram**

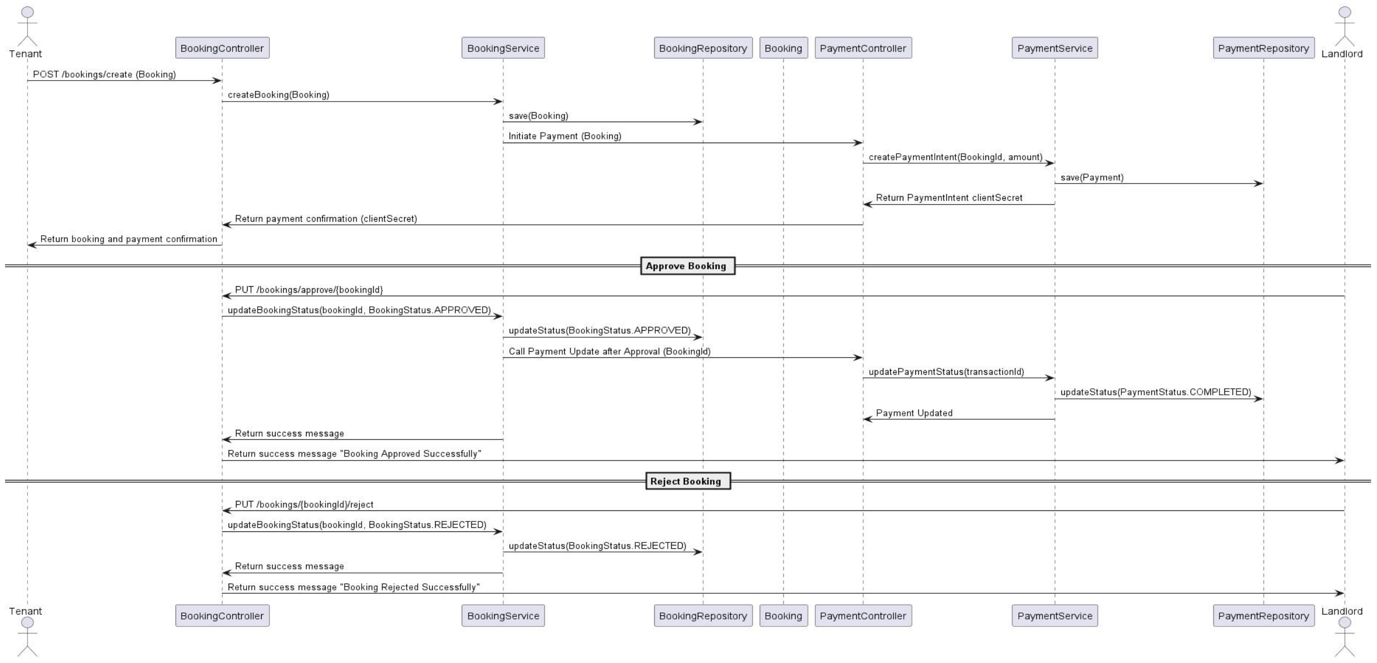


3.5 Class Diagram



3.6 Sequence Diagram

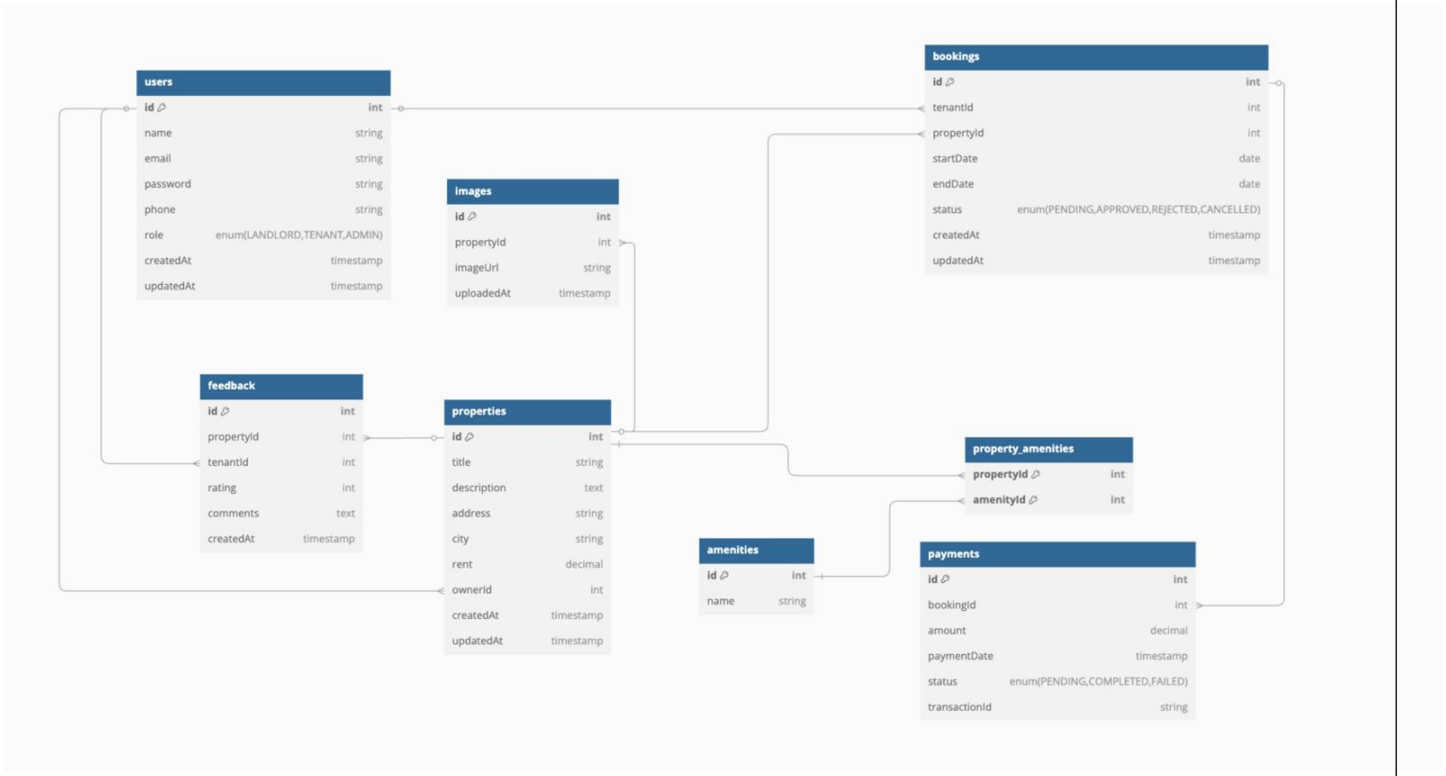
Booking Sequence:-


Tenant
BookingController
BookingService
BookingRepository
Booking
PaymentController
PaymentService
PaymentRepository
Landlord
POST /bookings/create (Booking)
createBooking(Booking)
save(Booking)
Initiate Payment (Booking)
createPaymentIntent(BookingId, amount)
save(Payment)
Return PaymentIntent clientSecret
Return payment confirmation (clientSecret)
Return booking and payment confirmation
Approve Booking
PUT /bookings/approve/{bookingId}
updateBookingStatus(bookingId, BookingStatus.APPROVED)
updateStatus(BookingStatus.APPROVED)
Call Payment Update after Approval (BookingId)
updatePaymentStatus(transactionId)
updateStatus(PaymentStatus.COMPLETED)
Payment Updated
Return success message
Return success message "Booking Approved Successfully"
Reject Booking
PUT /bookings/{bookingId}/reject
updateBookingStatus(bookingId, BookingStatus.REJECTED)
updateStatus(BookingStatus.REJECTED)
Return success message
Return success message "Booking Rejected Successfully"

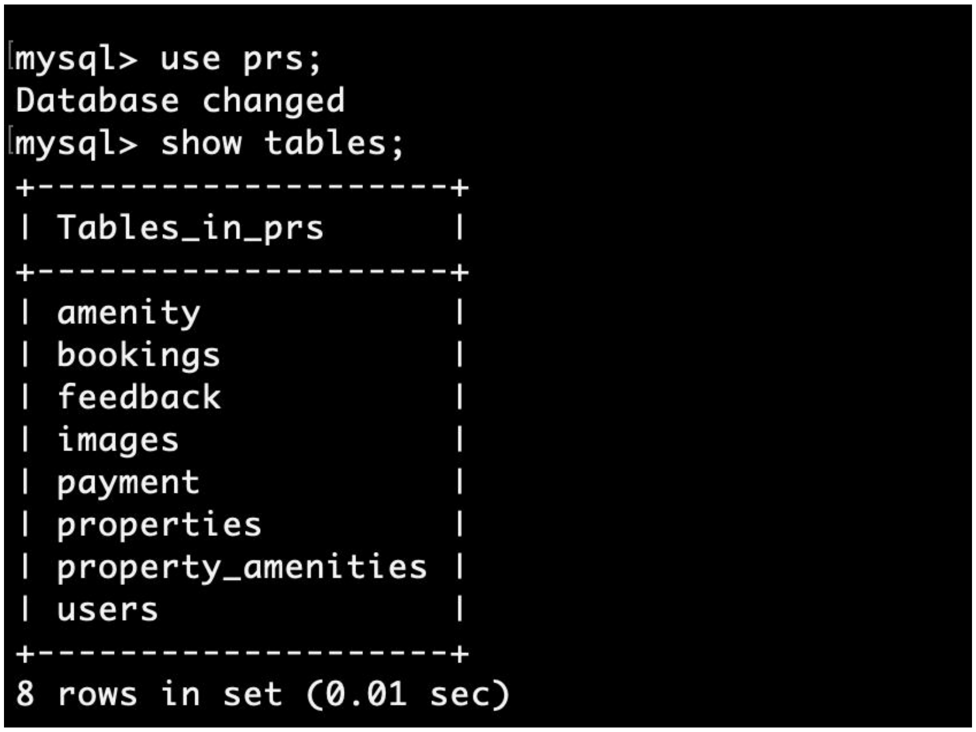
4. DATABASE DESIGN

Tables :

Design :-



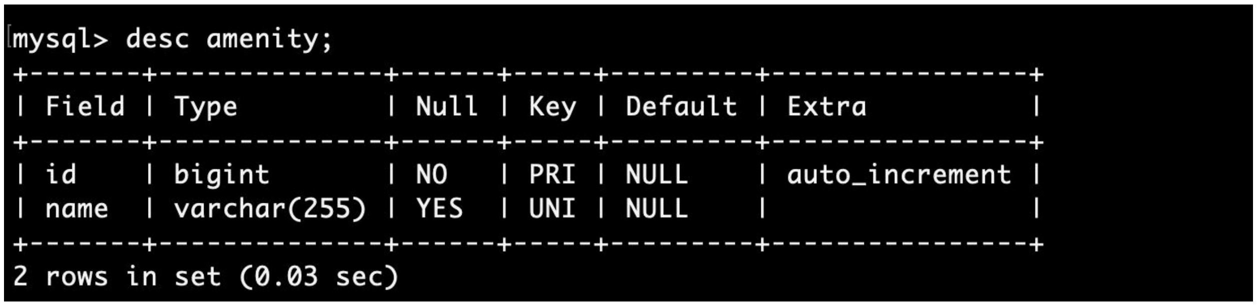
Database :-



```
mysql> use prs;
Database changed
mysql> show tables;
+--------------------+
| Tables_in_prs      |
+--------------------+
| amenity            |
| bookings           |
| feedback           |
| images             |
| payment            |
| properties         |
| property_amenities |
| users              |
+--------------------+
8 rows in set (0.01 sec)
```

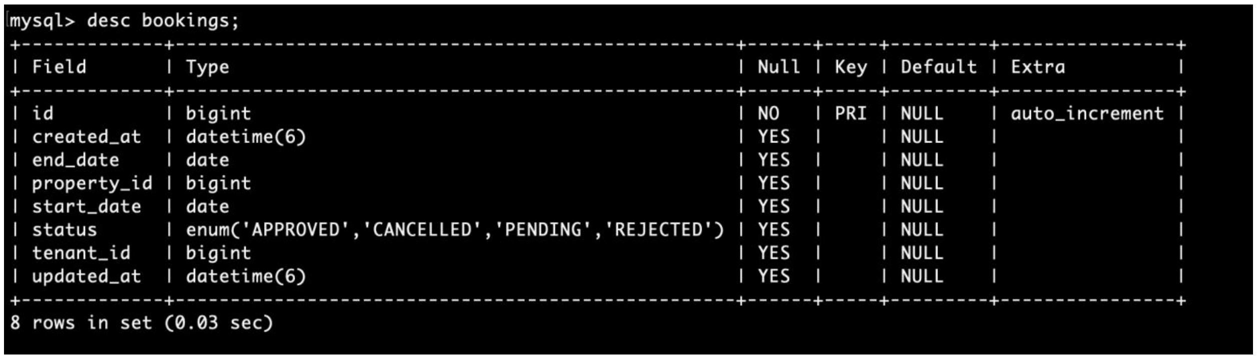
All Tables

1. **Table Amenity:-**



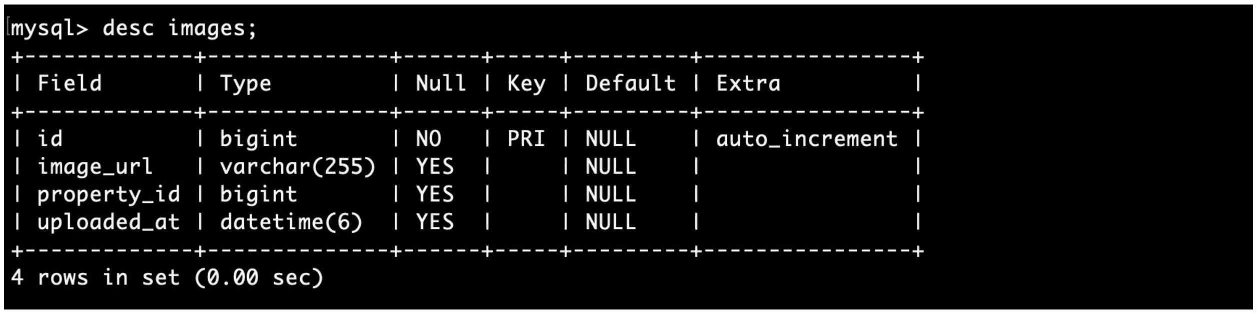
```
mysql> desc amenity;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | bigint       | NO   | PRI | NULL    | auto_increment |
| name  | varchar(255) | YES  | UNI | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
2 rows in set (0.03 sec)
```

2. **Table Bookings:-**



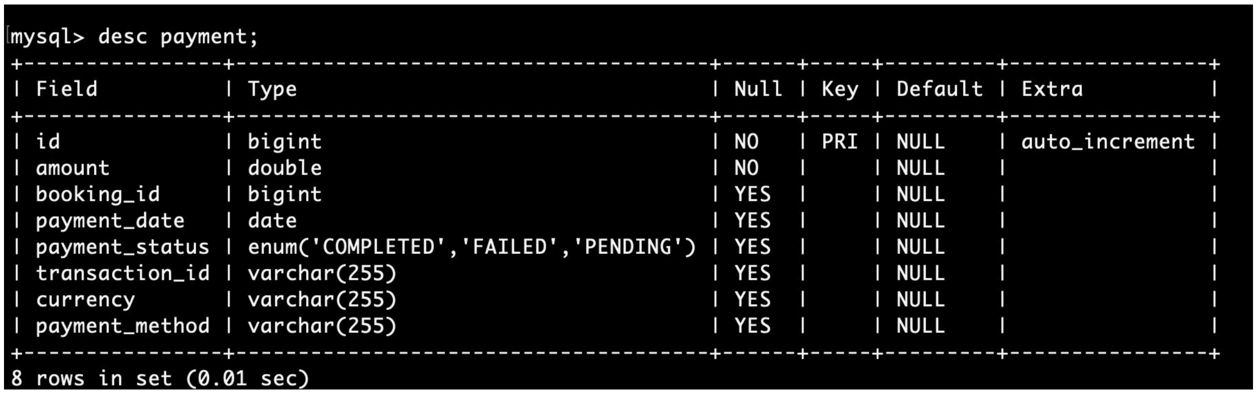
```
mysql> desc bookings;
+-------------+---------------------------------------------------+------+-----+---------+----------------+
| Field       | Type                                              | Null | Key | Default | Extra          |
+-------------+---------------------------------------------------+------+-----+---------+----------------+
| id          | bigint                                            | NO   | PRI | NULL    | auto_increment |
| created_at  | datetime(6)                                       | YES  |     | NULL    |                |
| end_date    | date                                              | YES  |     | NULL    |                |
| property_id | bigint                                            | YES  |     | NULL    |                |
| start_date  | date                                              | YES  |     | NULL    |                |
| status      | enum('APPROVED','CANCELLED','PENDING','REJECTED') | YES  |     | NULL    |                |
| tenant_id   | bigint                                            | YES  |     | NULL    |                |
| updated_at  | datetime(6)                                       | YES  |     | NULL    |                |
+-------------+---------------------------------------------------+------+-----+---------+----------------+
8 rows in set (0.03 sec)
```

3. **Table Images :-**



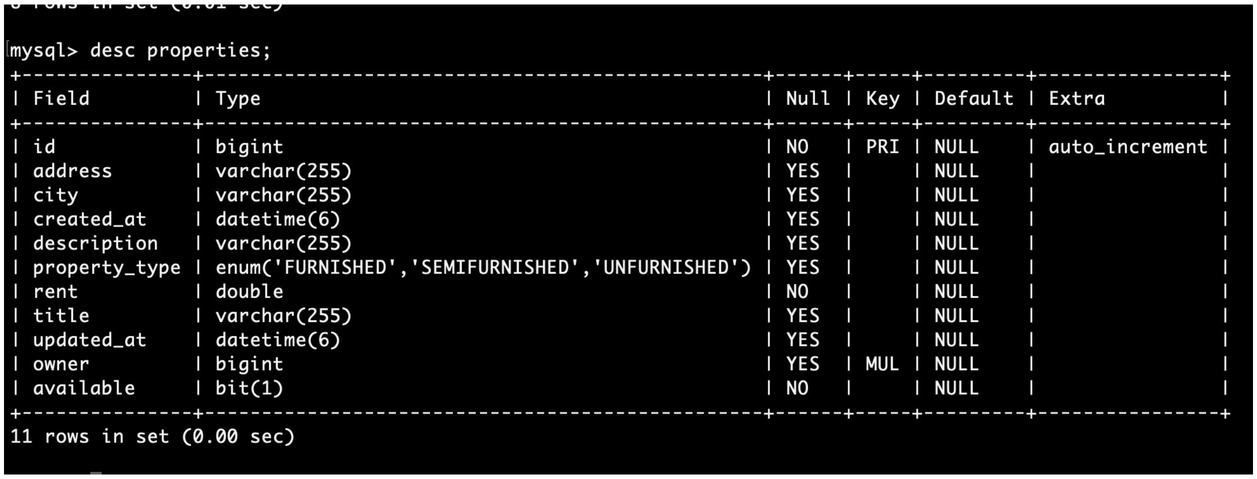
```
mysql> desc images;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | bigint       | NO   | PRI | NULL    | auto_increment |
| image_url   | varchar(255) | YES  |     | NULL    |                |
| property_id | bigint       | YES  |     | NULL    |                |
| uploaded_at | datetime(6)  | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

4. **Table Payments :-**



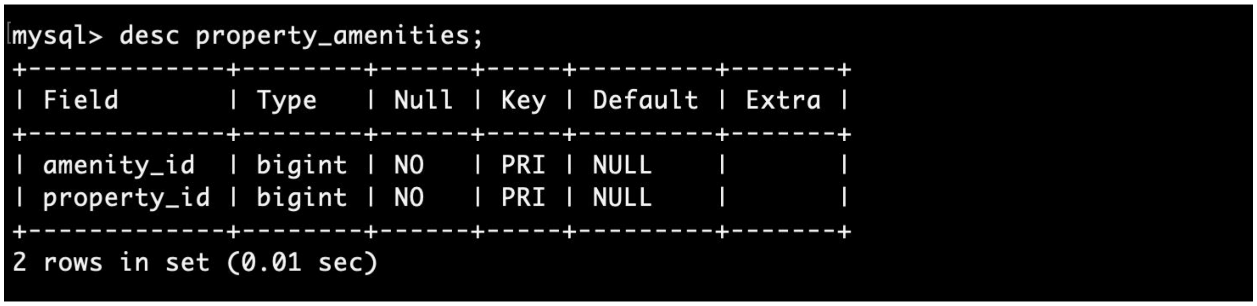
```
mysql> desc payment;
+----------------+-------------------------------------+------+-----+---------+----------------+
| Field          | Type                                | Null | Key | Default | Extra          |
+----------------+-------------------------------------+------+-----+---------+----------------+
| id             | bigint                              | NO   | PRI | NULL    | auto_increment |
| amount         | double                              | NO   |     | NULL    |                |
| booking_id     | bigint                              | YES  |     | NULL    |                |
| payment_date   | date                                | YES  |     | NULL    |                |
| payment_status | enum('COMPLETED','FAILED','PENDING') | YES  |     | NULL    |                |
| transaction_id | varchar(255)                        | YES  |     | NULL    |                |
| currency       | varchar(255)                        | YES  |     | NULL    |                |
| payment_method | varchar(255)                        | YES  |     | NULL    |                |
+----------------+-------------------------------------+------+-----+---------+----------------+
8 rows in set (0.01 sec)
```

5. Table Properties :-



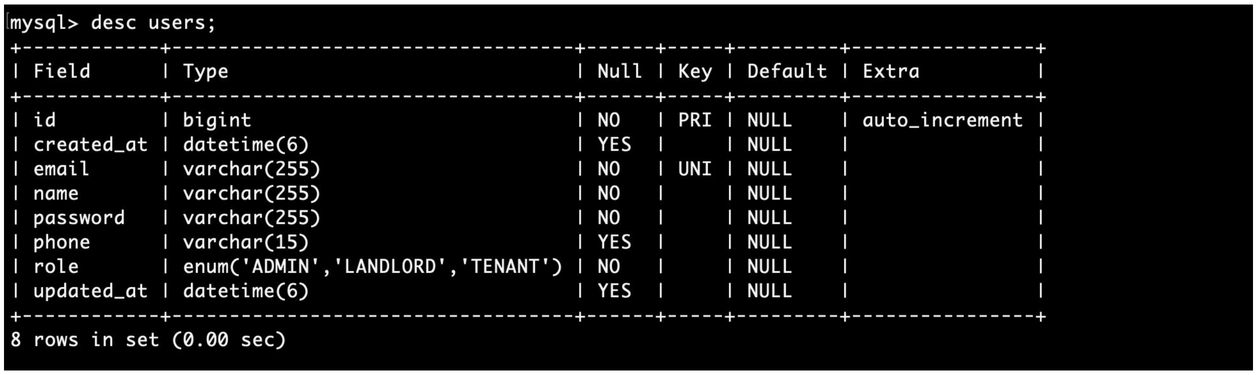
```
8 rows in set (0.01 sec)

mysql> desc properties;
+---------------+-----------------------------------------------+------+-----+---------+----------------+
| Field         | Type                                          | Null | Key | Default | Extra          |
+---------------+-----------------------------------------------+------+-----+---------+----------------+
| id            | bigint                                        | NO   | PRI | NULL    | auto_increment |
| address       | varchar(255)                                  | YES  |     | NULL    |                |
| city          | varchar(255)                                  | YES  |     | NULL    |                |
| created_at    | datetime(6)                                   | YES  |     | NULL    |                |
| description   | varchar(255)                                  | YES  |     | NULL    |                |
| property_type | enum('FURNISHED','SEMIFURNISHED','UNFURNISHED') | YES  |     | NULL    |                |
| rent          | double                                        | NO   |     | NULL    |                |
| title         | varchar(255)                                  | YES  |     | NULL    |                |
| updated_at    | datetime(6)                                   | YES  |     | NULL    |                |
| owner         | bigint                                        | YES  | MUL | NULL    |                |
| available     | bit(1)                                        | NO   |     | NULL    |                |
+---------------+-----------------------------------------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)
```

6. Table Property_Aminities :-



```
mysql> desc property_amenities;
+-------------+--------+------+-----+---------+-------+
| Field       | Type   | Null | Key | Default | Extra |
+-------------+--------+------+-----+---------+-------+
| amenity_id  | bigint | NO   | PRI | NULL    |       |
| property_id | bigint | NO   | PRI | NULL    |       |
+-------------+--------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

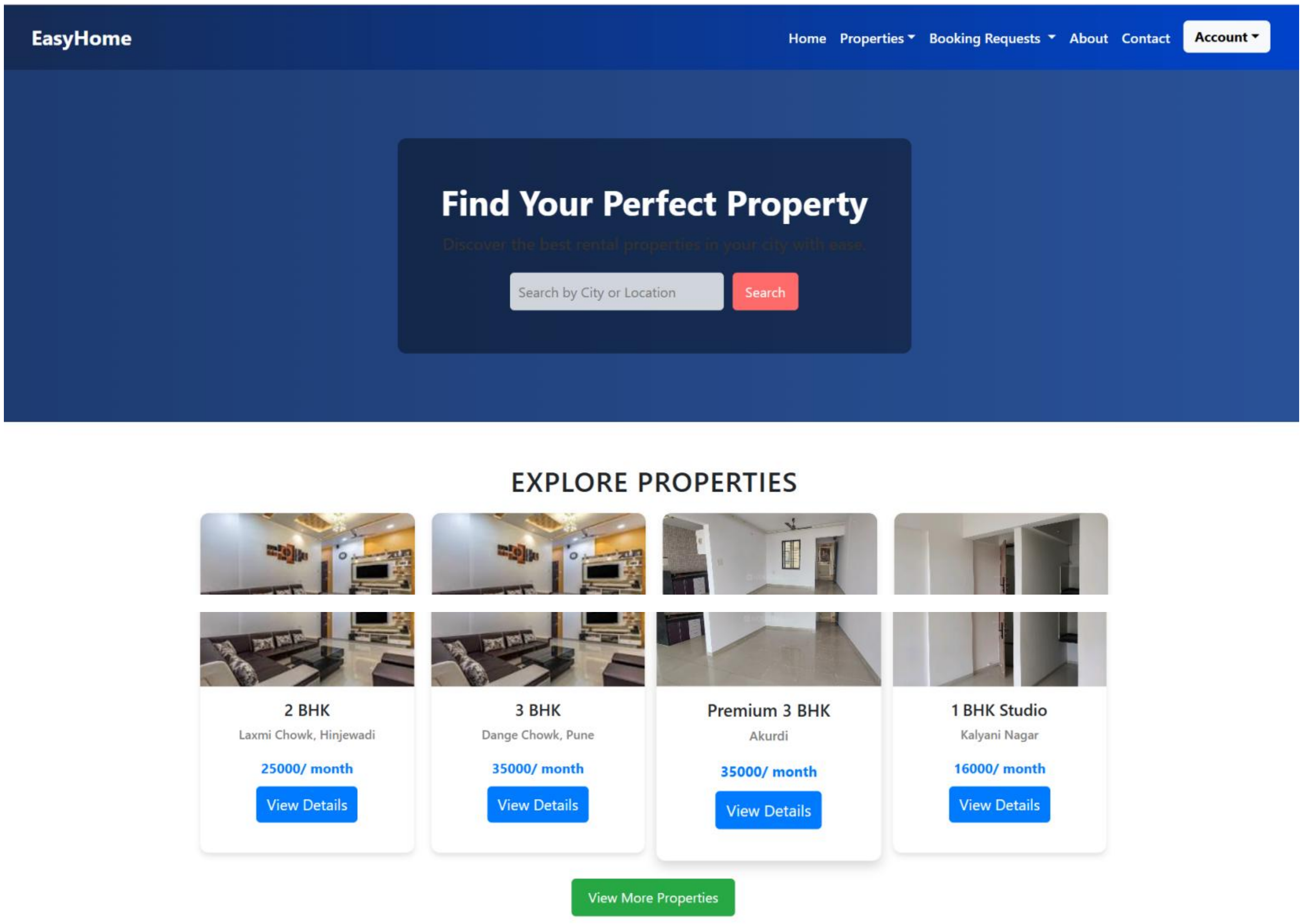
7. Table Users :-



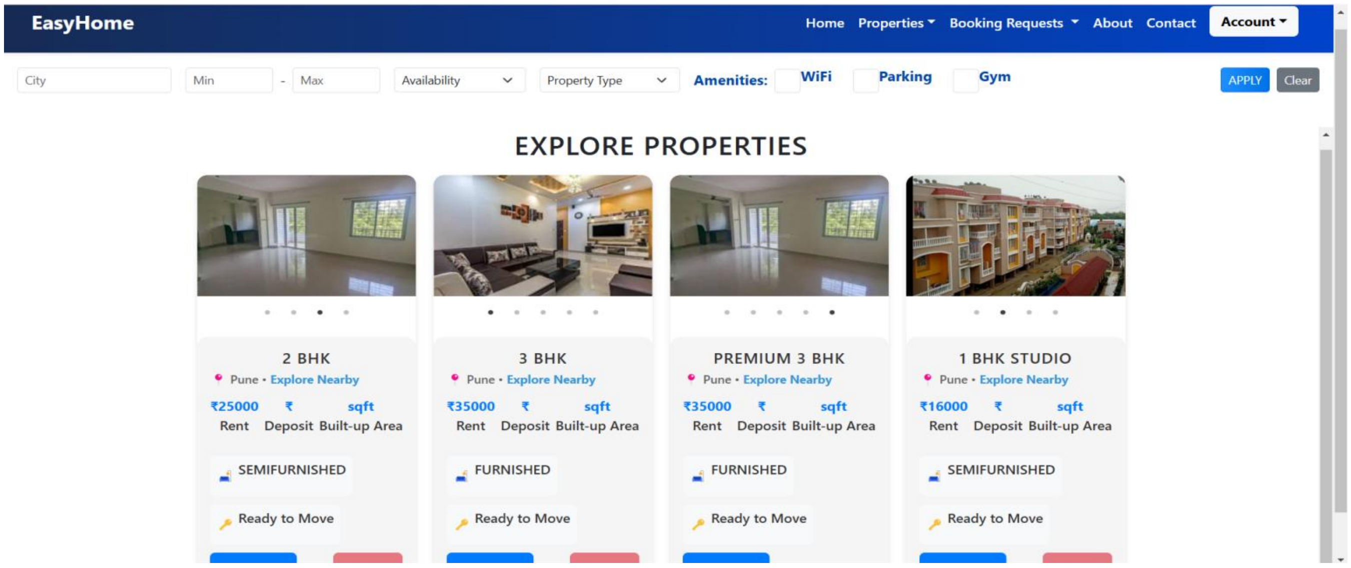
```
mysql> desc users;
+------------+-----------------------------------+------+-----+---------+----------------+
| Field      | Type                              | Null | Key | Default | Extra          |
+------------+-----------------------------------+------+-----+---------+----------------+
| id         | bigint                            | NO   | PRI | NULL    | auto_increment |
| created_at | datetime(6)                       | YES  |     | NULL    |                |
| email      | varchar(255)                      | NO   | UNI | NULL    |                |
| name       | varchar(255)                      | NO   |     | NULL    |                |
| password   | varchar(255)                      | NO   |     | NULL    |                |
| phone      | varchar(15)                       | YES  |     | NULL    |                |
| role       | enum('ADMIN','LANDLORD','TENANT') | NO   |     | NULL    |                |
| updated_at | datetime(6)                       | YES  |     | NULL    |                |
+------------+-----------------------------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

5. SNAPSHOTS

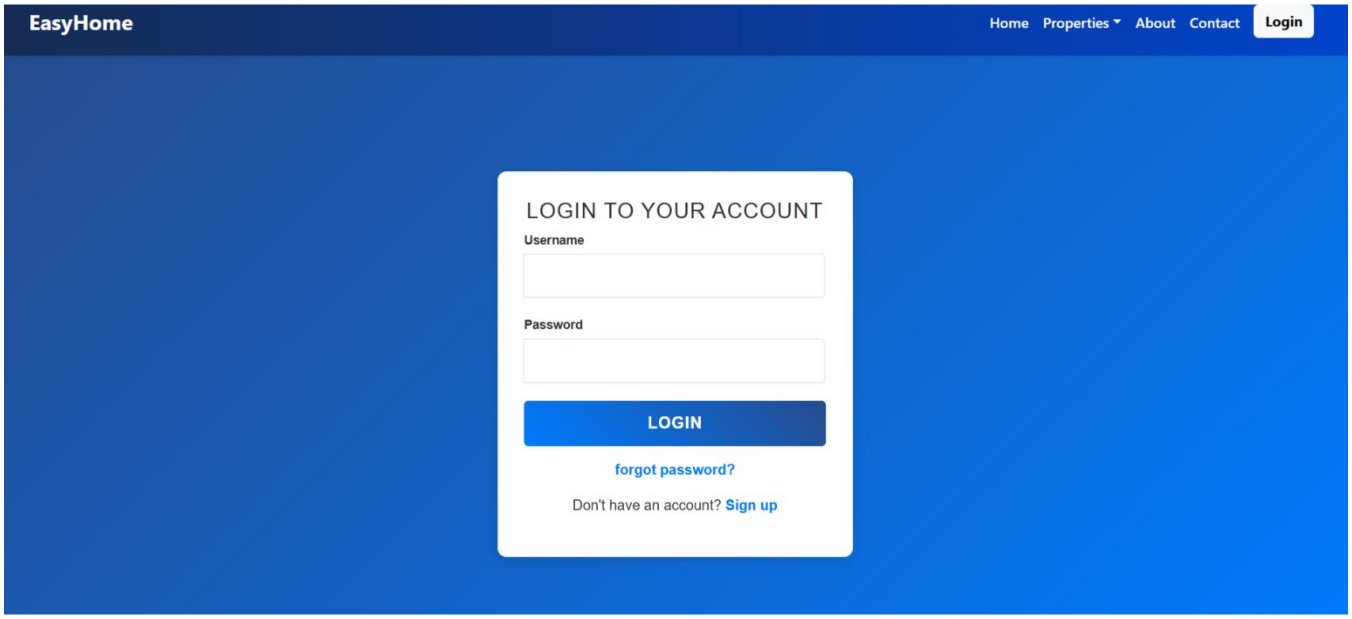
Home Page:



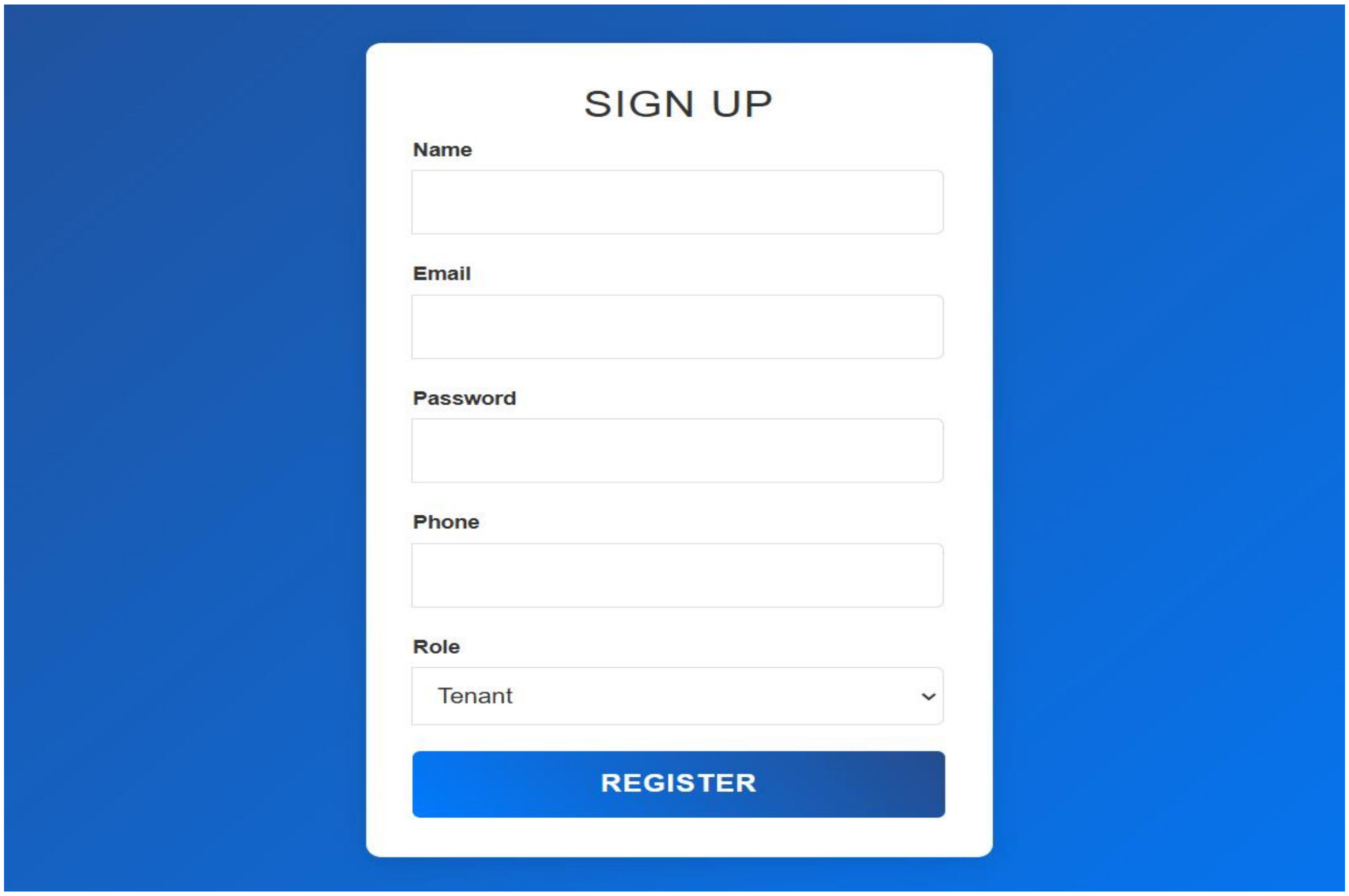
List of Properties:-



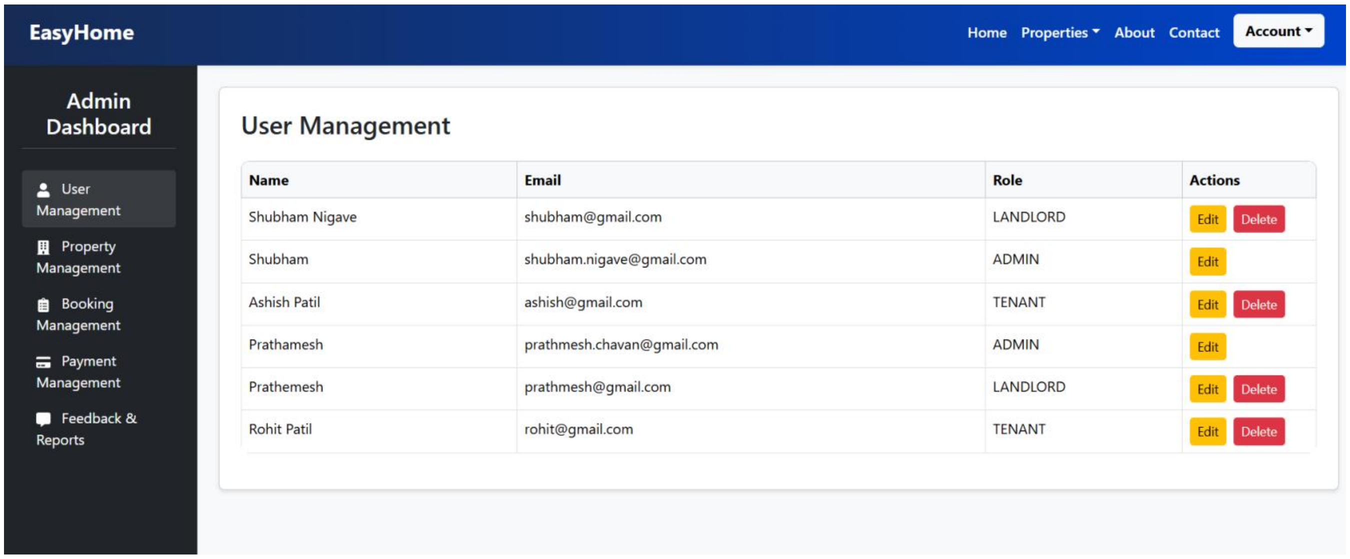
Login:



SignUp:-

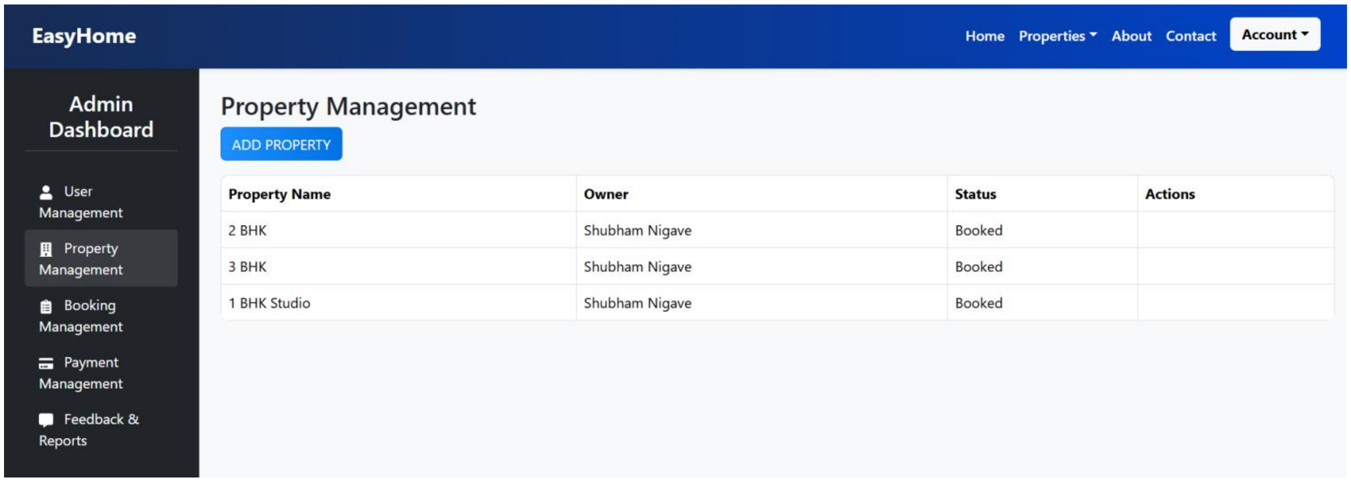


Admin Panel :-



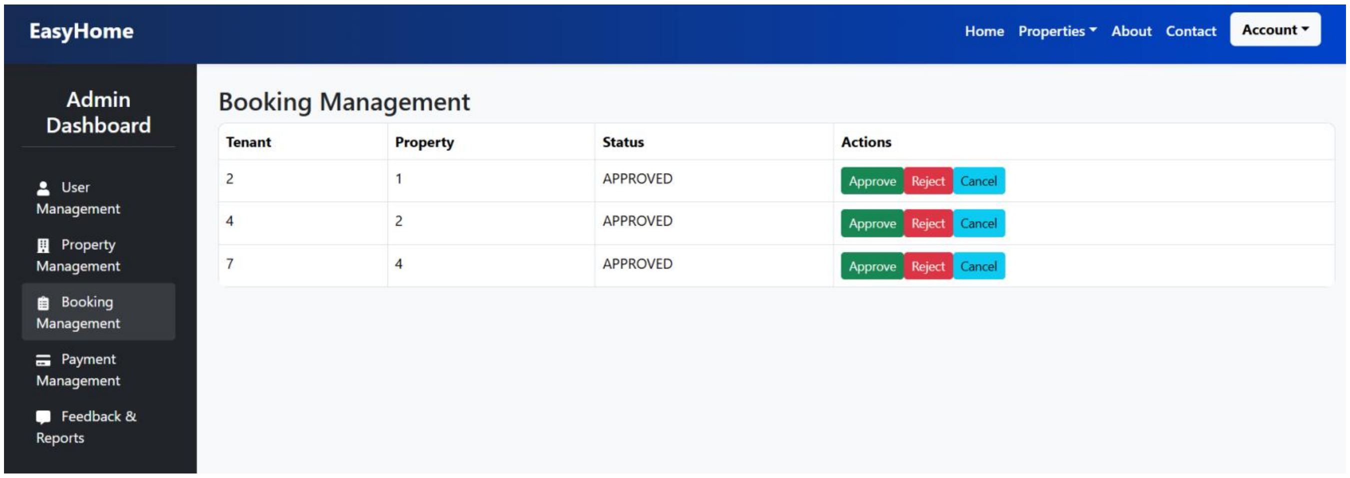
User Management

Name	Email	Role	Actions
Shubham Nigave	shubham@gmail.com	LANDLORD	Edit Delete
Shubham	shubham.nigave@gmail.com	ADMIN	Edit
Ashish Patil	ashish@gmail.com	TENANT	Edit Delete
Prathamesh	prathmesh.chavan@gmail.com	ADMIN	Edit
Prathemesh	prathmesh@gmail.com	LANDLORD	Edit Delete
Rohit Patil	rohit@gmail.com	TENANT	Edit Delete



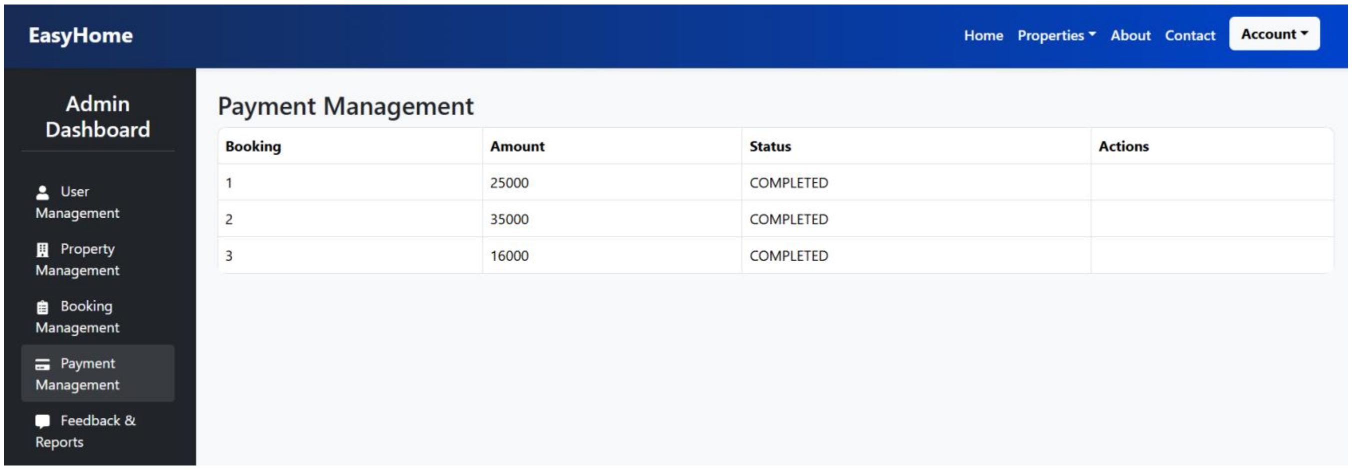
Property Management

Property Name	Owner	Status	Actions
2 BHK	Shubham Nigave	Booked	
3 BHK	Shubham Nigave	Booked	
1 BHK Studio	Shubham Nigave	Booked	



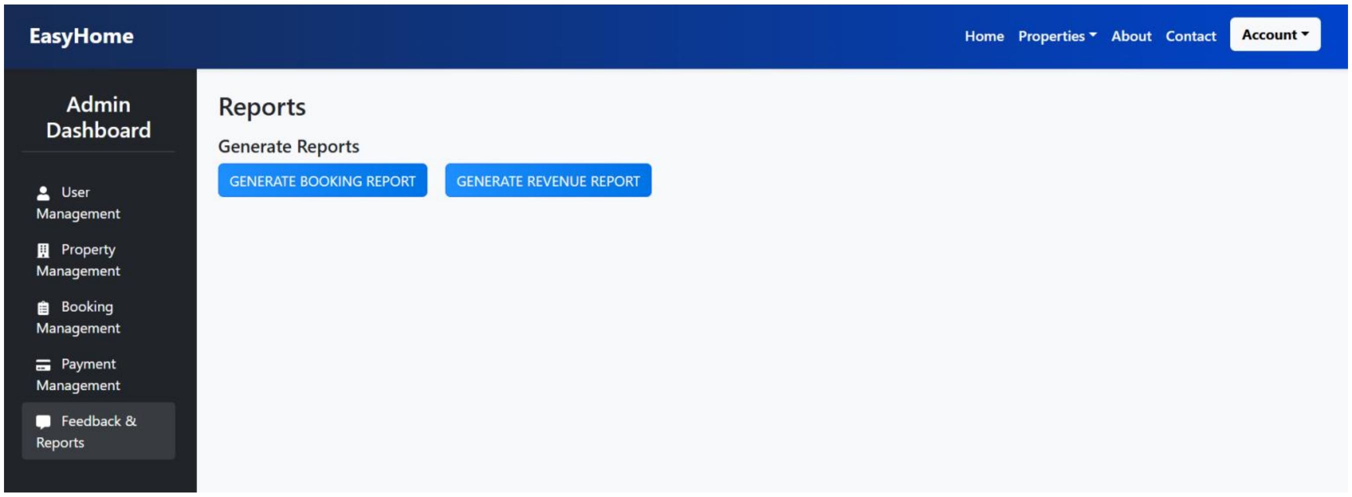
Booking Management

Tenant	Property	Status	Actions
2	1	APPROVED	Approve Reject Cancel
4	2	APPROVED	Approve Reject Cancel
7	4	APPROVED	Approve Reject Cancel

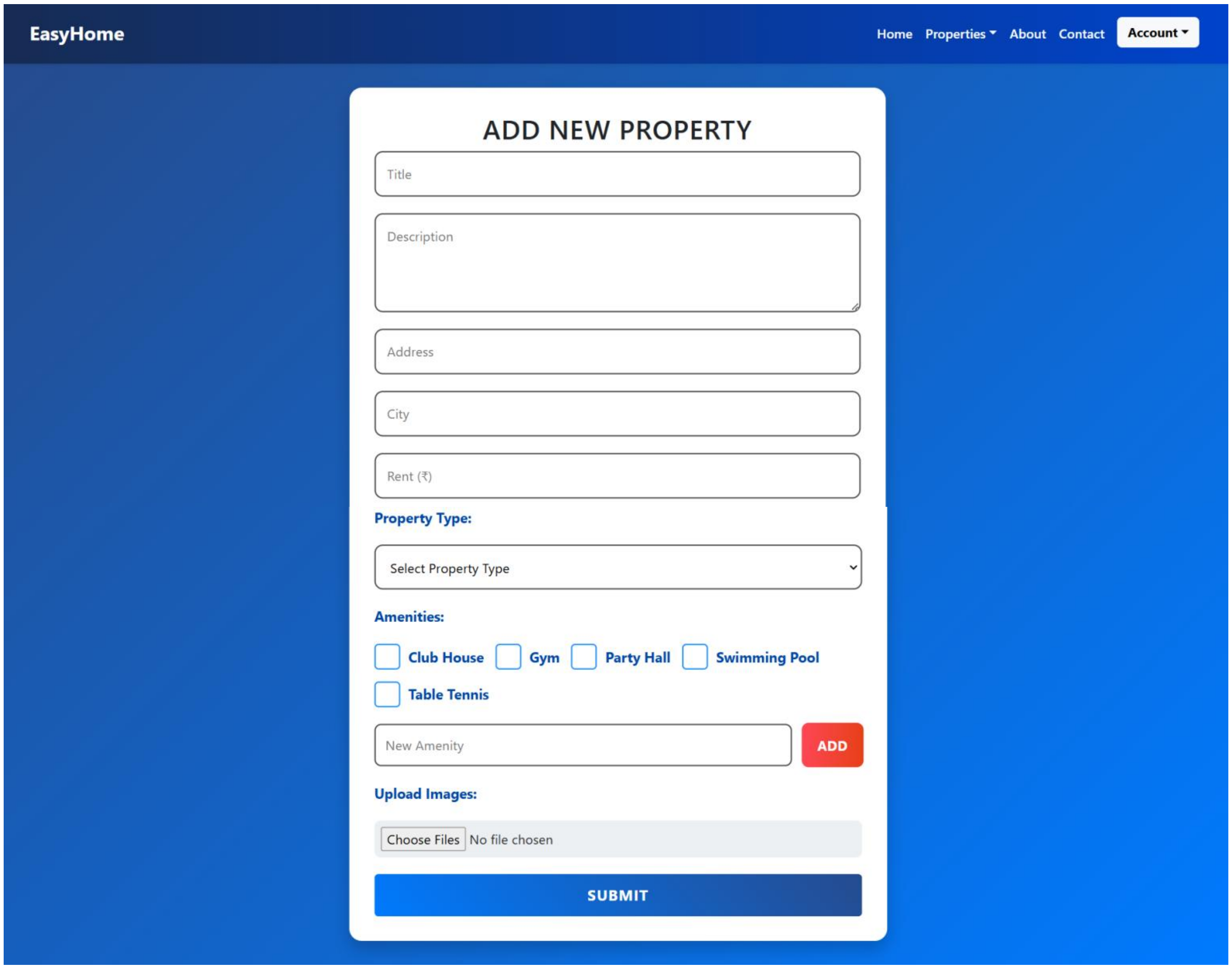


Payment Management

Booking	Amount	Status	Actions
1	25000	COMPLETED	
2	35000	COMPLETED	
3	16000	COMPLETED	



Add New Property:-



6. CONCLUSION

EasyHome - Property Rental System successfully meets the essential needs of property management and rental processes by offering a comprehensive solution that simplifies both tenant and landlord interactions. Its modern design, integration of secure payment systems, and user-friendly interface make it a reliable platform for seamless property searches, bookings, and payment processing.

The system leverages advanced technologies to ensure high performance, security, and scalability, enabling it to adapt to future advancements in the property rental market. By focusing on ease of use and operational efficiency, EasyHome is well-positioned to enhance the rental experience for both tenants and landlords. The platform's flexible architecture and robust features create a strong foundation for future growth and continuous improvement in the evolving real estate sector.

7. REFERENCES

1. https://docs.spring.io/spring-security/reference/
2. https://docs.spring.io/spring-boot/index.html/
3. https://www.amazon.com/Agile-Software-Development-Principles-Patterns/dp/0135974445
4. Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). *Database Systems: The Complete Book*. Prentice Hall.
5. Codd, E. F. (1970). *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM, 13(6), 377-387.
6. Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Addison-Wesley.
7. Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.
8. Soni, D., Nord, R. L., & Hofmeister, C. (1995). *Software Architecture in Industrial Applications*. Proceedings of the 17th International Conference on Software Engineering, 196-207.