

PropertyRentals

Online Property Renting Platform

Mini Project Report

Submitted by

Rijul Rojio

Reg. No.:AJC22MCA-2073

In Partial Fulfillment for the Award of the Degree of

**MASTER OF COMPUTER APPLICATIONS
(MCA TWO YEAR)**

[Accredited by NBA]

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE,
Accredited by NAAC . Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, “**PROPERTYRENTALS**” is the bonafide work of **RIJUL ROJIO (Regno: AJC22MCA-2073)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Mr. Jinson Devis

Internal Guide

Ms. Meera Rose Mathew

Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

DECLARATION

I hereby declare that the project report “**PROPERTYRENTALS**” is a bona-fide work of done at Amal Jyothi College of Engineering, towards the partial fulfillment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

Date:

RIJUL ROJIO

KANJIRAPPALLY

Reg: AJC22MCA-2073

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I extend my sincere gratitude to **Rev. Fr. Dr. Rubin Thottupurathu Jose**, our Head of the Department, for the significant support and assistance provided. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Mr. Jinson Devis** for his inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

RIJUL ROJIO

ABSTRACT

The "Property Renting Platform" mini project proposes the development of an intuitive and effective online platform geared towards bridging the gap between property owners and potential tenants in the realm of property rentals. In today's fast-paced urban environment, where property rentals are increasingly in demand, this project endeavors to create a comprehensive solution that addresses the challenges faced by property owners and prospective tenants alike.

Admin Module: Admins have privileged access and control over various aspects of the platform to ensure smooth operations and user satisfaction. Key functionalities include:

User Management: Admins can manage user accounts, including verification, suspension, and deletion of accounts if necessary.

Property Management: Admins can review and approve property listings.

System Maintenance: Admins can monitor and maintain the technical aspects of the platform, ensuring uptime and resolving technical issues.

Transaction Fees: The admin can charge a small transaction fee for processing payments securely through the platform.

Buyers (Tenants) Module: The Buyers module is designed to cater to individuals seeking properties for rent.

User Registration and Profile: Tenants can create accounts, set preferences, and manage their profiles.

Property Search: Tenants can search for properties based on various filters such as location, price range, property type, and amenities.

Property Details: Tenants can view detailed property listings, including descriptions, images, floor plans, and rental terms. The terms according to the property (if with house, for commercial purpose, plot/land (agriculture purpose, construction etc..))

Booking and Payments: Tenants can initiate the rental process, schedule viewings, and make payments securely through the platform. An agreement is placed b/w buyers and property owners to avoid any fraud practises.

Property Owners Module: The Property Owners module empowers individuals who have properties available for rent to list them on the platform and manage their interactions with potential tenants.

Property Listing: Property owners can create detailed listings with descriptions, images, amenities, and rental terms.

Profile Management: Property owners can manage their profiles, including contact information and property portfolio.

Tenant Screening: Property owners can review tenant profiles, before approving rental requests.

Rent Collection: Property owners can manage rental payments and security deposits through integrated payment gateways.

(Notifications)

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	4
2.1	INTRODUCTION	5
2.2	EXISTING SYSTEM	5
2.3	DRAWBACKS OF EXISTING SYSTEM	5
2.4	PROPOSED SYSTEM	6
2.5	ADVANTAGES OF PROPOSED SYSTEM	7
3	REQUIREMENT ANALYSIS	8
3.1	FEASIBILITY STUDY	9
3.1.1	ECONOMICAL FEASIBILITY	9
3.1.2	TECHNICAL FEASIBILITY	9
3.1.3	BEHAVIORAL FEASIBILITY	10
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	10
3.2	SYSTEM SPECIFICATION	14
3.2.1	HARDWARE SPECIFICATION	14
3.2.2	SOFTWARE SPECIFICATION	14
3.3	SOFTWARE DESCRIPTION	14
3.3.1	DJANGO	14
3.3.2	SQLITE	15
4	SYSTEM DESIGN	16
4.1	INTRODUCTION	17
4.2	UML DIAGRAM	17
4.2.1	USE CASE DIAGRAM	18
4.2.2	SEQUENCE DIAGRAM	19
4.2.3	STATE CHART DIAGRAM	20
4.2.4	ACTIVITY DIAGRAM	22
4.2.5	CLASS DIAGRAM	23
4.2.6	OBJECT DIAGRAM	24
4.2.7	COMPONENT DIAGRAM	24

4.2.8	DEPLOYMENT DIAGRAM	25
4.3	USER INTERFACE DESIGN USING FIGMA	27
4.4	DATABASE DESIGN	29
5	SYSTEM TESTING	37
5.1	INTRODUCTION	38
5.2	TEST PLAN	38
5.2.1	UNIT TESTING	39
5.2.2	INTEGRATION TESTING	39
5.2.3	VALIDATION TESTING	40
5.2.4	USER ACCEPTANCE TESTING	40
5.2.5	AUTOMATION TESTING	40
5.2.6	SELENIUM TESTING	40
6	IMPLEMENTATION	54
6.1	INTRODUCTION	55
6.2	IMPLEMENTATION PROCEDURE	55
6.2.1	USER TRAINING	56
6.2.2	TRAINING ON APPLICATION SOFTWARE	56
6.2.3	SYSTEM MAINTENANCE	57
7	CONCLUSION & FUTURE SCOPE	58
7.1	CONCLUSION	59
7.2	FUTURE SCOPE	59
8	BIBLIOGRAPHY	60
9	APPENDIX	62
9.1	SAMPLE CODE	63
9.2	SCREEN SHOTS	80

List of Abbreviation

IDE	Integrated Development Environment
HTML	Hyper Text Markup Language.
CSS	Cascading Style Sheet
SQLite	Relational Database Management System
UML	Unified Modeling Language
JS	Java Script

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The "PropertyRentals" platform is an innovative digital solution designed to streamline and enhance the property rental experience for both landlords and tenants. Our platform is dedicated to providing a seamless and user-friendly interface, allowing users to effortlessly navigate, discover, and engage with available rental properties from the comfort of their homes. Property Rentals offers a diverse range of rental options, including residential properties, commercial spaces, and land for various purposes, ensuring a comprehensive selection to meet the diverse needs of our users. With a commitment to user satisfaction and leveraging advanced technology, Property Rentals aims to establish a digital sanctuary for property seekers, offering a hassle-free and enjoyable journey to find and secure their ideal rental spaces. Whether searching for a home, a business location, or a plot of land for specific purposes, Property Rentals endeavors to simplify the entire rental process, making it a convenient and enriching experience for all users. Through Property Rentals, we aspire to foster a thriving online community of property owners and tenants, contributing to the ease and joy of property rentals on a global scale.

1.2 PROJECT SPECIFICATION

The "PropertyRentals" web application is tailored to facilitate seamless interactions between Admin and users in the dynamic realm of property rentals. It provides a comprehensive set of features to ensure an efficient and enjoyable experience for both landlords and tenants in the online property rental space.

Admin Module:

- **User Management:** Admins wield privileged access to efficiently manage user accounts, overseeing verification, suspension, and deletion when necessary.
- **Property Management:** Admins have the authority to review and approve property listings, ensuring a high standard on the platform.
- **System Maintenance:** Admins monitor and maintain the technical aspects of the platform, guaranteeing uptime and resolving technical issues promptly.
- **Transaction Fees:**
 - The admin can implement a small transaction fee for secure payment processing through the platform.

Buyers (Tenants) Module:

- User Registration and Profile: Tenants can effortlessly create accounts, set preferences, and manage their profiles.
- Property Search: Tenants have the ability to search for properties using various filters such as location, price range, property type, and amenities.
- Property Details: Tenants can access detailed property listings, including descriptions, images, floor plans, and rental terms tailored to specific property types.
- Booking and Payments: Tenants can initiate the rental process, schedule viewings, and securely make payments through the platform. Agreements are established between buyers and property owners to prevent fraudulent practices.

Property Owners Module:

- Property Listing: Property owners can create detailed listings with comprehensive information, including descriptions, images, amenities, and rental terms.
- Profile Management: Property owners have the flexibility to manage their profiles, including contact information and property portfolios.
- Tenant Screening: Property owners can review tenant profiles before approving rental requests, ensuring a secure and trustworthy rental process.
- Rent Collection: Property owners can efficiently manage rental payments and security deposits through integrated payment gateways, with notifications to streamline the process.

Technologies Used:

Frontend: HTML, CSS, JavaScript

Backend: Django.

Database: SQLite.

Payment Gateway Integration: Razor pay.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

In an age marked by digital evolution and shifting paradigms in the real estate landscape, the "PropertyRentals " emerges as a strategic response to the changing dynamics of property rentals. Envisioned as a web application, this project aims to establish seamless interactions between property owners and potential tenants. Carefully designed to transcend traditional rental practices, the Property Rentals offers a comprehensive suite of functionalities, optimizing administrative processes and enhancing the overall experience for both landlords and tenants in the dynamic realm of property rentals.

2.2 EXISTING SYSTEM

The established Property Rentals Management System ("PropertyRentals") encompasses essential features for streamlined operations and an intuitive user experience. Admins efficiently oversee property listings, categorize rentals for easy search, and manage user accounts with personalized benefits. Detailed insights into tenant profiles aid in refining property marketing strategies. Tenants enjoy a seamless registration process, secure payment processing, and the option to book rental spaces. Personalized wishlists, advanced search functionality, and a user-friendly interface enhance the overall rental experience. The system ensures secure access through user authentication, prioritizing data security. With a focus on diverse property options and up-to-date listings, this system fosters tenant loyalty, engagement, and satisfaction in the dynamic landscape of property rentals

2.2.1 NATURAL SYSTEM STUDIED

The Property Rentals Management System functions as a natural system, embodying a dynamic ecosystem where technological components, user behaviors, and market forces seamlessly converge. The technological backbone, comprised of servers and databases, facilitates efficient property management and user interactions, forming the foundation of the system. Users, as integral participants in this ecosystem, exert influence through their behaviors and preferences, shaping the platform's response to evolving market dynamics and trends within the property rental industry. This holistic approach ensures a symbiotic relationship between technology, user engagement, and the ever-changing landscape of property rentals.

2.2.2 DESIGNED SYSTEM STUDIED

The Property Rentals Management System is an intricately designed platform that places a premium on user-centric features for both administrators and users. Admins benefit from efficient property management, intuitive categorization of rental spaces, and dynamic controls over property listings. Comprehensive insights into tenant profiles contribute to strategic decision-making. The system ensures secure access through a robust login mechanism. Tenants experience a seamless journey with effortless registration, secure payment processing, and the flexibility to rent properties. Personalized wishlists and advanced search functionality enhance the overall rental exploration, while a streamlined interface for property bookings facilitates easy transactions. This system is poised to optimize user experiences, drive engagement, and establish a thriving online property rental platform that caters to the diverse preferences of property seekers.

2.3 DRAWBACKS OF EXISTING SYSTEM

- The current system may lack advanced search and filtering features, making it challenging for users to efficiently locate specific rental properties based on detailed criteria such as location, amenities, or rental terms.
- With the expanding portfolio of rental spaces, the existing system may face scalability challenges, potentially leading to performance issues and slower response times. This could impede the seamless exploration and booking experience for users.
- If the existing system is not adequately optimized for mobile devices, it may present a drawback, given the growing preference of users to search and book rental properties using smartphones and tablets.
- Limited Payment Options
- Insufficient Personalization
- Complex Checkout Process

2.4 PROPOSED SYSTEM

The envisioned "PropertyRentals" is a comprehensive and user-friendly online property rental system crafted to cater to a diverse audience of property seekers and landlords. Offering an extensive array of rental spaces across various categories, including residential, commercial, and

land, it provides personalized property recommendations and streamlined booking processes. Robust user account management, secure payment options, and dedicated customer support ensure a seamless experience. Property Rentals aspires to redefine the property rental landscape, simplifying the way individuals discover, secure, and enjoy rental spaces, catering to the needs of tenants and property owners alike.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- The PropertyRentals Platform offers a comprehensive range of rental spaces, including residential, commercial, and land, accommodating diverse preferences and needs of tenants and property owners.
- PropertyRentals prioritizes a streamlined booking process, ensuring users can quickly and easily complete their property rentals. A user-friendly interface contributes to tenant satisfaction and encourages repeat transactions.
- The platform emphasizes the security of financial transactions by offering a variety of secure payment options. This instills confidence in users, addressing concerns related to online payment security and promoting more secure and confident rental bookings.
- PropertyRentals aims to be inclusive, catering to a diverse audience of property seekers, including individuals looking for residential spaces, commercial properties, or land. The user-friendly interface and extensive property listings make the rental process accessible and enjoyable for users with varying preferences and purposes.
- PropertyRentals simplifies the property search process, allowing users to effortlessly discover rental spaces that match their specific needs and preferences. This ensures a quick and efficient experience, saving users time and effort in finding the perfect property.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Feasibility is the degree to which a project can be carried out successfully. A feasibility study is conducted to assess the solution's viability, which establishes whether it is viable and implementable in the program. The feasibility study considers details like the availability of resources, software development costs, the advantages of the software to the business once it is built, and the costs associated with maintaining it. The outcome of the feasibility study should be a report recommending whether the requirements engineering, and system development process should be continued. A system is of no real value to a corporation if it does not serve its goals. Even though this may seem obvious, many organizations create systems that do not support their goals, either because they lack a clear statement of these goals, because they fail to specify the system's business requirements, or because other organizational or political factors have an impact on the procurement of the system.

3.1.1 ECONOMIC FEASIBILITY

The economic feasibility of your online property rentals project is critical to assess its financial viability and potential profitability. Conducting a thorough evaluation of various financial aspects will help determine if the project is economically sound. Begin with a comprehensive cost-benefit analysis, comparing the total expenses of development, operation, and maintenance against the expected benefits and revenue generation from book sales. Estimate the initial development costs, ongoing operational expenses, and potential revenue from property sales to calculate the projected return on investment (ROI) and the break-even point. Analyze the profit margin per book sale to ensure sustainability and competitiveness. Additionally, conduct market research to understand the demand, competition, and growth potential in the book industry. By conducting a comprehensive economic feasibility analysis, you can make informed decisions and assess the project's financial potential for long-term success.

3.1.2 TECHNICAL FEASIBILITY

The technical feasibility of your online property rentals project is crucial to determine whether it can be practically implemented from a technological standpoint. Evaluating various aspects is essential for its success. Firstly, ensure that your organization has the necessary technology infrastructure or can acquire it to support the platform, including web hosting, databases, and server capacity. Check for skilled software developers who can build and maintain the system effectively. Integration with third-party services like payment gateways and shipping providers must be

seamless. The architecture should be scalable to accommodate increasing traffic and data as the platform grows. Security measures to protect user data and transactions must be robust. Ensure the platform is mobile-friendly for users on various devices, and its performance is fast and responsive. Compliance with data protection and privacy regulations is essential. Lastly, evaluate the financial resources required for development, hosting, maintenance, and support. A comprehensive assessment of technical feasibility will help identify potential challenges and enable you to make informed decisions to create a successful online bookstore platform

3.1.3 BEHAVIORAL FEASIBILITY

The Behavioral feasibility of your online property rentals project is essential to assess whether the proposed system can be effectively integrated into the organization's existing operations. Evaluating various aspects will help determine the project's viability and success. Firstly, ensure that the necessary resources, including human capital, technology, and infrastructure, are available or can be obtained within the project's time frame and budget. Consider the skill set of the organization's staff and identify the need for additional training or hiring. Evaluate how well the new system will integrate with existing processes, such as inventory management and order processing. Plan for change management to address potential resistance to organizational changes resulting from the system implementation. Ensure that third-party services or suppliers can support operational requirements. Verify compliance with legal and regulatory aspects related to online sales and data protection. Gather user feedback to gauge acceptance and identify areas for improvement. Lastly, analyse operational costs to ensure alignment with the organization's budget. A thorough assessment of operational feasibility will help ensure the successful integration of the online bookstore platform into the organization's existing operations.

3.1.4 FEASIBILITY STUDY QUESTIONNAIRE

1. Project Overview?

The "PropertyRentals" platform is an innovative digital solution designed to streamline and enhance the property rental experience for both landlords and tenants. Our platform is dedicated to providing a seamless and user-friendly interface, allowing users to effortlessly navigate, discover, and engage with available rental properties from the comfort of their homes. Property Rentals offers a diverse range of rental options, including residential properties, commercial spaces, and land for various purposes, ensuring a comprehensive selection to meet the diverse needs of our users. With a commitment to user satisfaction and leveraging advanced technology, Property Rentals aims to establish a digital sanctuary for property seekers, offering a hassle-free and enjoyable journey to

find and secure their ideal rental spaces. Whether searching for a home, a business location, or a plot of land for specific purposes, Property Rentals endeavors to simplify the entire rental process.

2. To what extend the system is proposed for?

The proposed PropertyRentals platform is a comprehensive online system designed to serve a broad audience in the property rental domain. Offering diverse rental spaces such as residential, commercial, and land, the platform ensures accessibility for tenants and property owners alike. With a user-friendly interface, transparent property information, and secure transaction processes, it aims to provide a seamless rental experience. Community engagement features like ratings, reviews, wishlists, and periodic discounts enhance user interaction. The platform aspires to revolutionize the property rental landscape, making it accessible and enjoyable for a wide range of users, including tenants seeking their ideal spaces and property owners with varying rental offerings.

3. Specify the Viewers/Public which is to be involved in the System?

The PropertyRentals platform involves individuals seeking rental spaces, property owners looking to list their properties, students searching for accommodation, business owners scouting commercial spaces, and anyone in need of land for various purposes. Tenants explore diverse rental options, while property owners showcase their available spaces. Students find suitable accommodations, and businesses discover commercial properties. The platform serves a broad audience, making it inclusive for individuals with varied rental needs. Whether someone is seeking a home, a storefront, or land for a specific purpose, the Property Rentals system aims to provide a user-friendly and comprehensive solution for all involved in the property rental process.

4. List the Modules included in your System?

Here is a concise list of modules included in The PropertyRentals system:

1. User Registration and Login
2. Property Listing and Management
3. Property Search and Filtering
4. Property Details
5. Sending Rental Requests
6. Payment Gateway Integration
8. User Account Management
9. Admin Panel

Each module plays a crucial role in providing users with a seamless and enjoyable online Property Rentals experience while efficiently managing the platform's operations and enhancing user satisfaction.

5. Identify the users in your project?

The PropertyRentals involves three types of users: Property Owner, Tenant and admin.

6. Who owns the system?

Administrator

7. System is related to which firm/industry/organization?

The system described, a PropertyRentals platform, is related to the real estate and property rental industry. It caters to individuals and businesses seeking rental spaces, as well as property owners looking to list their properties for rent. The platform is designed to streamline the process of discovering, booking, and managing rental properties, covering various categories such as residential, commercial, and land. Targeted towards tenants, property owners, students, businesses, and anyone in need of rental spaces, the system aims to provide a user-friendly and comprehensive solution in the dynamic realm of property rentals.

8. Details of person that you have contacted for data collection.

a. Mostly Online Sources

b. The Process of data collection for PropertyRentals involves collaboration with a diverse range of stakeholders. It encompasses collecting and curating essential data, such as a property information, user profiles, etc... The data collection effort requires engagement with multiple parties, including :

- .Real Estate Experts
- Real Estate owners association

9. Questionnaire to collect details about the project?

a) Who are the primary users or customers you want to cater to through the platform?

Property Owners and Tenants.

b) List the key features and functionalities that users will have access to on the platform

Property Search, Property Management, etc.

c) How will you provide customer support to address user inquiries and concerns?

Email, helpline

d) What technologies and programming languages will you use to develop the platform?

The technologies and languages use in project is python with Django, HTML, CSS, Bootstrap, JavaScript, SQLite

e) Do you have any plans for future expansion or additional features after the initial launch of the online platform?

The PropertyRentals platform envisions a dynamic future with strategic expansions and enhanced features post-launch. The roadmap includes a unique property sales functionality, allowing owners to directly sell their properties within the platform, fostering diverse transaction opportunities. Third-party engagement in property transactions will be facilitated, broadening the platform's versatility. An advanced analytics dashboard will empower administrators with insightful metrics for data-driven decision-making. Machine learning algorithms for automated property approval and subscription plans for property owners will streamline listings and enhance user engagement. The property owners' module will introduce advanced listing management, while the buyers' module will feature expanded search filters for a refined property search experience. To foster seamless communication, an integrated platform will facilitate interactions between property owners and potential buyers or tenants, enhancing the overall user experience. These additions collectively aim to position the Property Rentals platform as a comprehensive and user-centric solution in the dynamic property rental landscape.

f) How do you plan to gather property data and other relevant information for your platform?

To collect comprehensive property data and relevant information for your Property Rentals platform, employ a multi-faceted approach. Establish partnerships with real estate agencies and property owners to access their listings, integrate with property management software systems, leverage APIs and data feeds from real estate databases, employ web scraping tools for data extraction from property listings (while ensuring compliance with terms of service and legal requirements), encourage user contributions and feedback to enhance property details, collaborate with data providers for extensive property databases, access official land registry databases, utilize geographic information systems (GIS) for location-based data, and consider content partnerships and direct submissions from property owners. Ensure strict adherence to data privacy regulations, maintain compliance with legal requirements, and implement regular updates to provide users with accurate and up-to-date property information.

g) Have you conducted a competitor analysis to understand the strengths and weaknesses of existing online property rental platforms? If yes, please provide a summary.

The envisioned Property Rentals platform is a dynamic online solution crafted to provide a seamless and enriching experience for individuals seeking rental spaces. Offering a diverse range of properties, including residential, commercial, and land, the platform ensures users can explore personalized recommendations, transparent property information, and user reviews. Community

engagement is fostered through features such as tenant forums and property showcases, promoting a sense of belonging within the rental community. The primary objectives encompass simplifying the property search process, ensuring secure and transparent transactions, and catering to the varied rental preferences of tenants and property owners. The Property Rentals Hub aspires to redefine the property rental landscape, offering a convenient and inclusive platform for individuals with diverse rental needs.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	Intel core i3
RAM	4GB
Hard disk	256GB SSD

3.2.2 Software Specification

Front End	HTML5, CSS, Bootstrap
Back End	Django, SQLite
Database	SQLite
Client on PC	Windows 7 and above.
Technologies used	Django, HTML5, Bootstrap, JS,

3.3 SOFTWARE DESCRIPTION

3.3.1 DJANGO

Django, a leading open-source web framework, epitomizes the pinnacle of modern web development with its efficiency and versatility. Built on Python, Django follows the Model-View-Controller architecture, prioritizing simplicity and flexibility. Noteworthy features include its Object-Relational Mapping system, streamlined URL routing, and a user-friendly template engine. The built-in admin interface simplifies data management, while robust security measures and middleware support enhance application integrity. Django scales effortlessly, accommodating growing datasets and traffic demands. Its REST Framework extension further extends its utility to API development. Supported by an active community and extensive documentation, Django stands

as a powerful toolkit, seamlessly shaping the development of dynamic and secure web applications for diverse purposes, from content management systems to Restful API. In essence, Django empowers developers with a comprehensive and adaptable framework for crafting sophisticated and salable web solutions.

3.3.2SQLite

SQLite, a lightweight and server less relational database management system, is celebrated for its simplicity, portability, and efficiency. Operating as a self-contained, single-file database, SQLite requires minimal setup and eliminates the need for a separate server process. This design makes it a preferred choice for embedded systems, mobile applications, and small to medium-scale projects. With zero configuration and a server less architecture, SQLite streamlines the database management process. It supports ACID transactions, ensuring data integrity and reliability, even in the face of system interruptions. The dynamic typing feature allows flexible data storage, accommodating various data types within the same column.

Noteworthy is SQLite's cross-platform compatibility, making it versatile across different operating systems and environments. Its wide adoption is evident in applications ranging from mobile apps to web browsers, where it is utilized for local data storage. SQLite stands out as a go-to solution for projects requiring a lightweight, self-contained, and easily deployable database system, embodying simplicity without compromising on functionality and reliability.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Any designed system or product's development process begins with the design phase. An efficient system depends on a well-executed design, which is a creative process. It entails utilizing a variety of techniques and concepts to completely specify a procedure or system for it to be implemented. Regardless of the development paradigm used in software engineering, the design phase is essential. It seeks to produce the architectural detail needed to design a system or product and acts as the process's technical backbone.

To maximize every aspect of efficiency, performance, and accuracy, this program underwent a comprehensive design phase. A user-oriented document is converted into a document for programmers or database workers throughout the design phase.

4.2 UML DIAGRAM

Software engineering uses the Unified Modeling Language (UML), a standardized visual language, to model, develop, and document software systems. UML diagrams provide a concise and organized approach to represent different facets of a software system, serving as a common communication tool for developers, stakeholders, and designers. There are many different varieties of UML diagrams, including class diagrams, sequence diagrams, use case diagrams, and more. Each is designed to communicate a particular piece of knowledge about the design, operation, and interactions of the system. UML diagrams are essential to the software development process because they help with the visualization, analysis, and design of complex systems, which in turn makes the process more effective and efficient.

- Use case diagram
- Sequence diagram
- State diagram
- Activity diagram
- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between system components. An approach for identifying, outlining, and organizing system requirements is called a use case. The word "system" here refers to a thing that is being created or run, like a website for mail-order product sales and services. UML (Unified Modeling Language), a standard language for the modelling of real-world objects and systems, uses use case diagrams. The planning of general requirements, the validation of a hardware design, the testing and debugging of a software product in development, the creation of an online help reference, or the completion of a job focused on customer support are all examples of system objectives. For instance, use cases in a product sales context can involve customer service, item ordering, catalogue updating, and payment processing. There are four elements in a use case diagram.

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases. Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we must use the following guidelines to draw an efficient use case diagram.
- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.

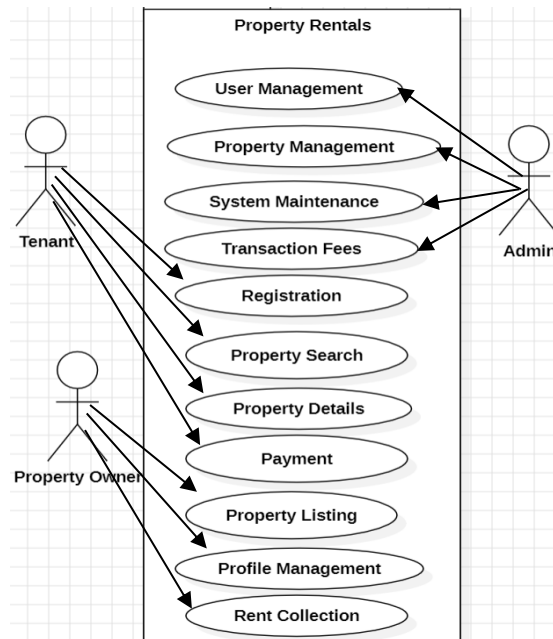


Figure 1: Use Case diagram

4.2.2 SEQUENCE DIAGRAM

A sort of Unified Modeling Language (UML) diagram called a sequence diagram is used in software engineering to depict the communications and interactions that take place across time between various system components or objects. It gives a live view of the interactions between objects as they work together to complete a certain job or situation.

Objects are shown as lifelines in a sequence diagram, and vertical lines indicate their existence over time. The flow of calls or messages between objects is shown by arrows and lines between lifelines. These diagrams are very helpful for demonstrating the interactions within a particular use case, assisting in determining the sequence of operations and the functions that each object performs.

Sequence diagrams are essential for understanding the behavior of a system and for ensuring that the software functions as intended, as they offer a clear and detailed depiction of how different components or objects collaborate to achieve a particular functionality.

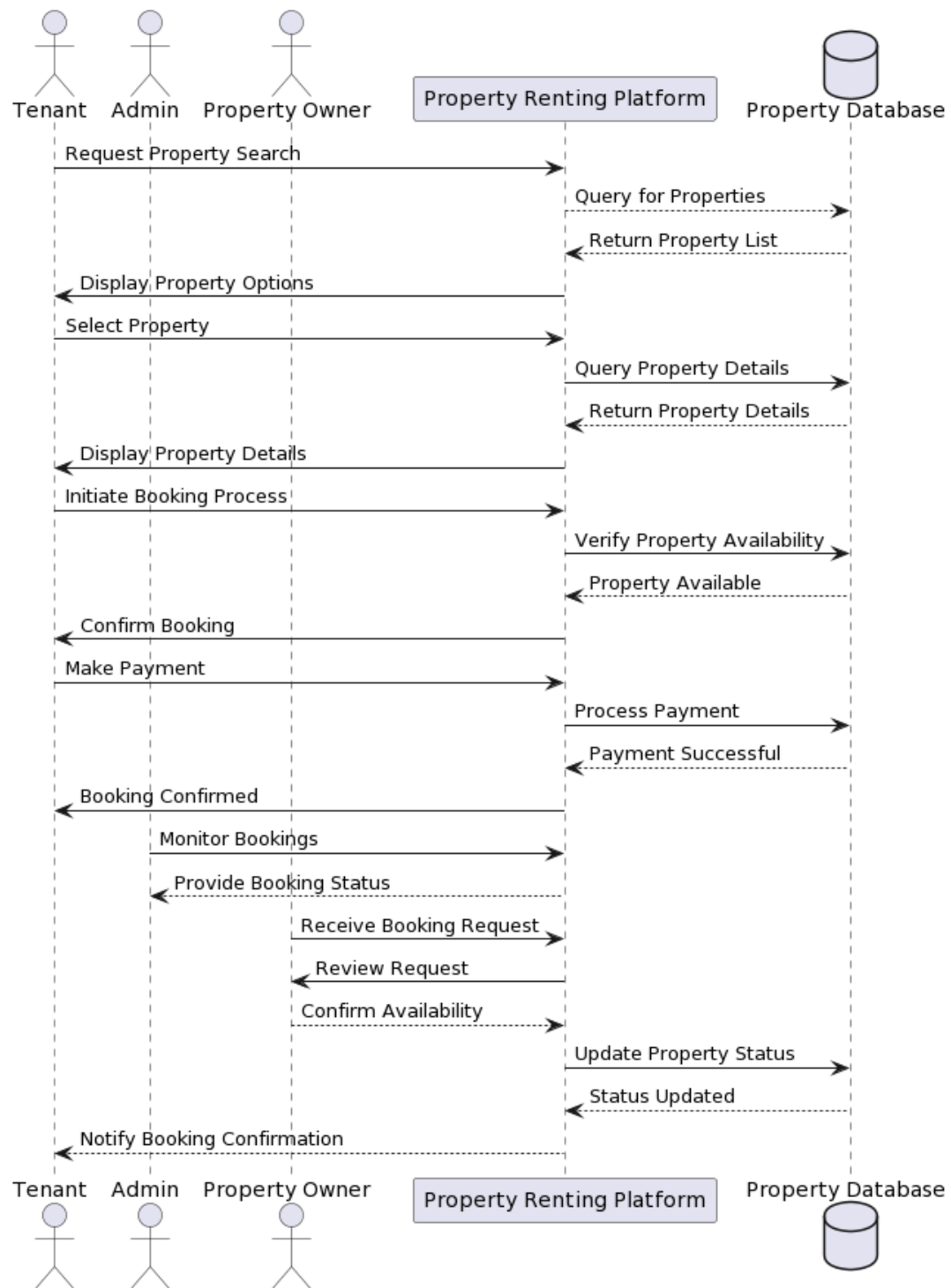


Figure 2: Sequence diagram

4.2.3 STATE CHART DIAGRAM

A state diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML). In this context, a state defines a stage in the evolution or

behavior of an object, which is a specific entity in a program or the unit of code representing that entity. A state diagram resembles a flowchart in nature; however, a flowchart shows the processes within a system that alters the state of an object rather than the actual state changes themselves. The first step to creating a state chart diagram is identifying the initial and final states of a system. Then, all the possible existing states are placed in relation to the beginning and the end. Lastly, all of the events that trigger state changes are labeled as transition elements.

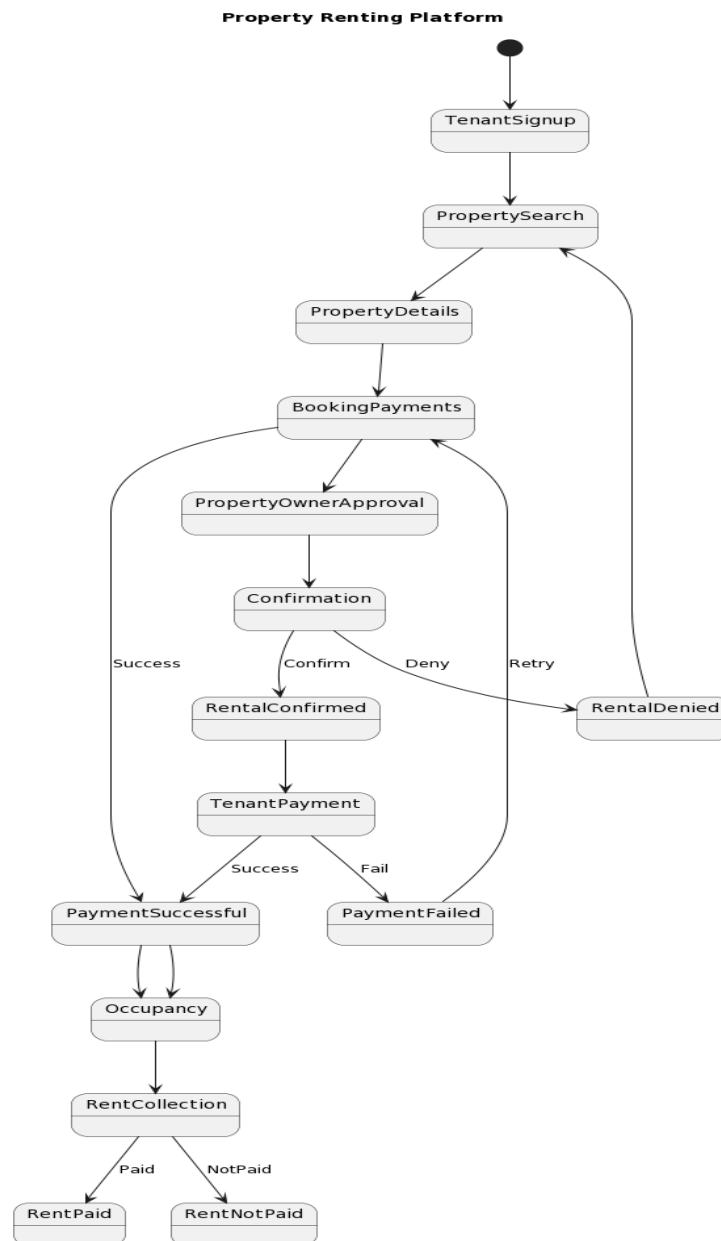


Figure 3: State Chart diagram

4.2.4 ACTIVITY DIAGRAM

An activity diagram is essentially a flowchart that shows how one activity leads to another. The action might be referred to as a system operation. One operation leads to the next in the control flow. This flow may be parallel, contemporaneous, or branched. Activity diagrams use many features, such as fork, join, etc., to cope with all types of flow control. An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

Activity diagram is another important behavioral diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

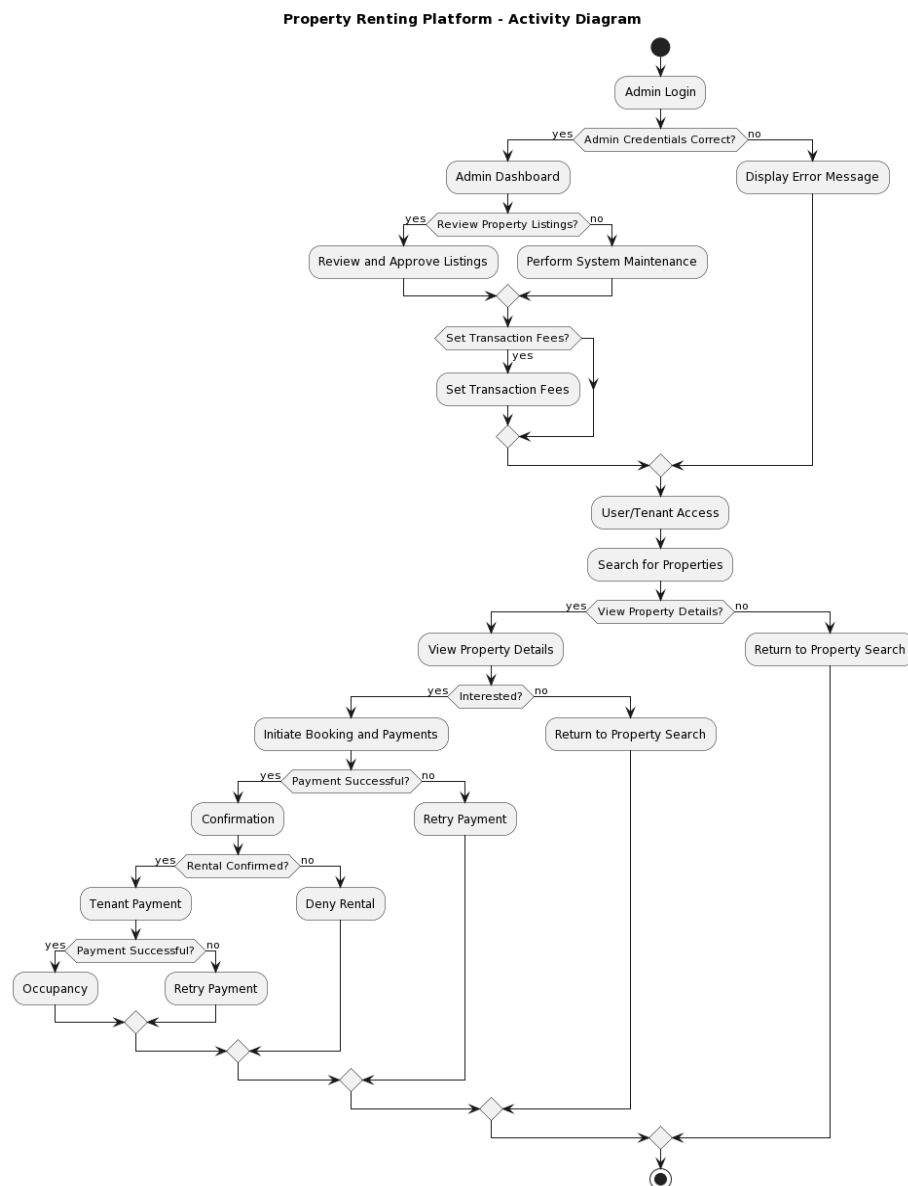


Figure 4: Activity diagram

4.2.5 CLASS DIAGRAM

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling. Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. Class diagrams are useful in many stages of system design. In the analysis stage, a class diagram can help you to understand the requirements of your problem domain and to identify its components. In an object-oriented software project, the class diagrams that you create during the early stages of the project contain classes that often translate into actual software classes and objects when you write code. Later, you can refine your earlier analysis and conceptual models into class diagrams that show the specific parts of your system, user interfaces, logical implementations, and so on. Your class diagrams then become a snapshot that describes exactly how your system works, the relationships between system components at many levels, and how you plan to implement those components.

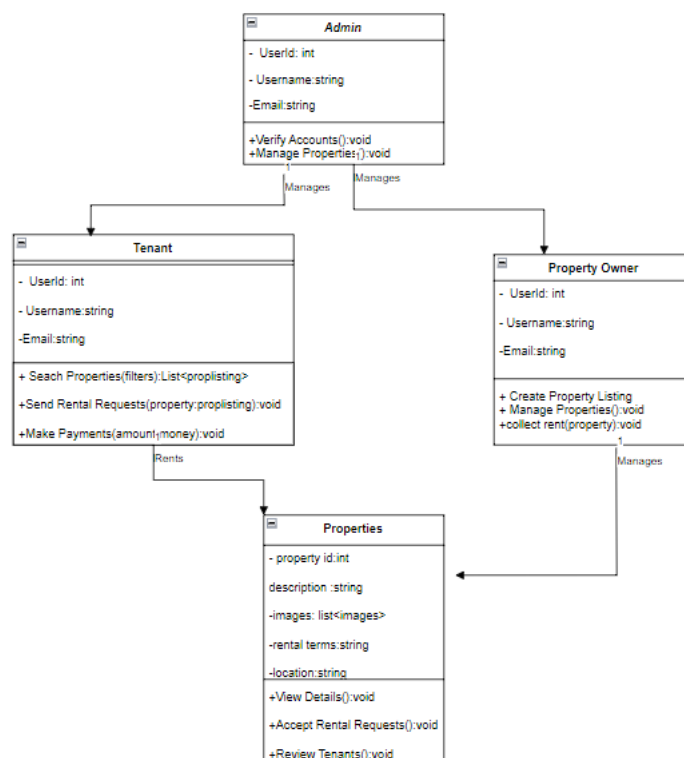


Figure 5: Class diagram

4.2.6 OBJECT DIAGRAM

Since class diagrams are the source of object diagrams, class diagrams are a prerequisite for object diagrams. An instance of a class diagram is represented by an object diagram. Class and object diagrams both use the same fundamental ideas. The static view of a system is also represented by object diagrams, but this static view represents a momentary snapshot of the system. To represent a group of items and their connections as an instance, object diagrams are employed

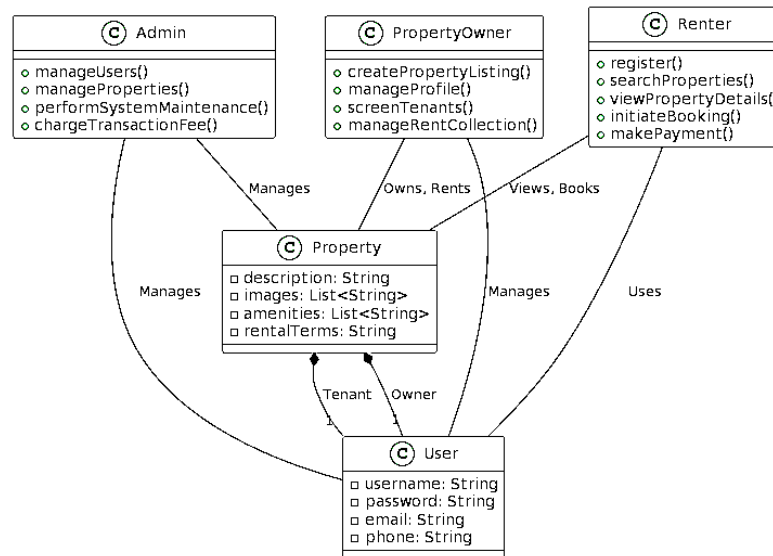


Figure 6: Object diagram

4.2.7 COMPONENT DIAGRAM

UML diagrams are used to represent and describe the behavior of object-oriented systems, assisting in visualization, explanation, and thorough documentation. Particularly in class diagrams, where they record the static behavior of a system, these diagrams play a vital role. On the other hand, graphics can impair the efficiency of real multitasking systems. To maintain coordination, each part within the broader process has a distinct duty and only communicates with other essential components.

A component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.

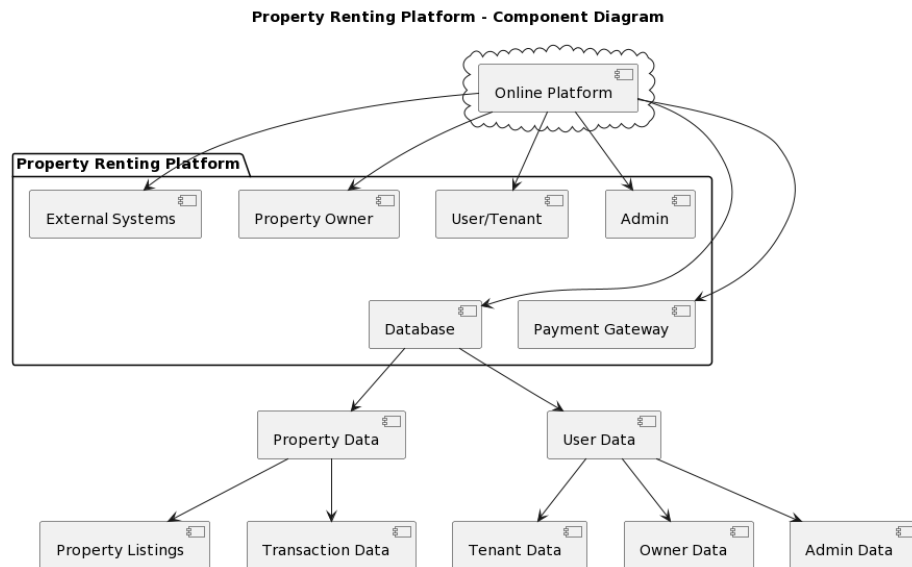


Figure 7: Component diagram

4.2.8 DEPLOYMENT DIAGRAM

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths. In contrast to other UML diagram types, which primarily depict the logical components of a system. The deployment diagram does not focus on the logical components of the system, but it put its attention on the hardware topology.

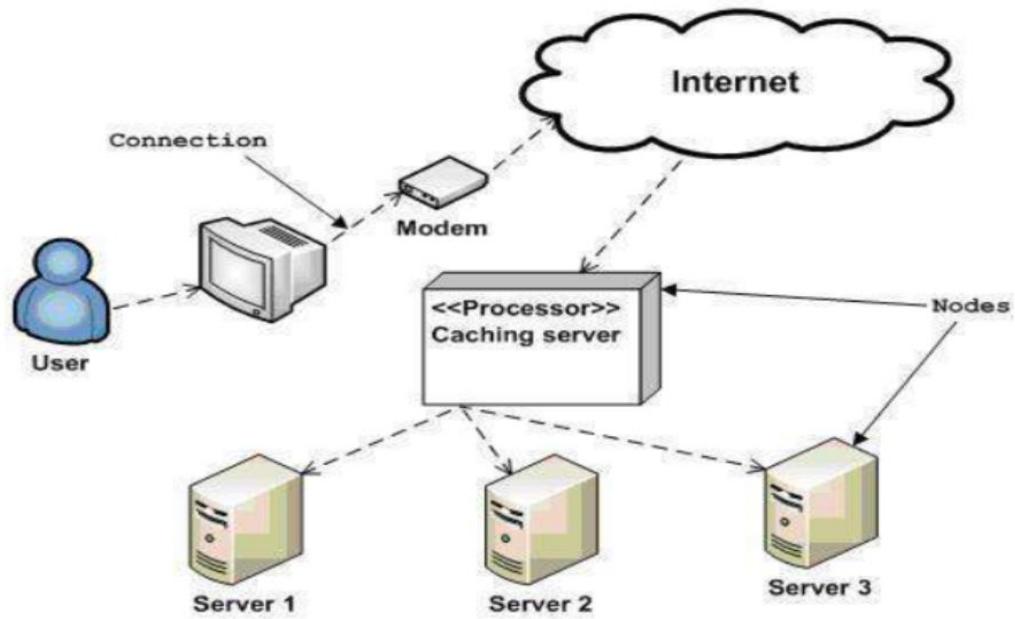


Figure 8: Deployment diagram

4.3 USER INTERFACE DESIGN USING FIGMA

Home Page



Registration Form

REGISTER NOW

UPLOAD YOUR PHOTO

FIRST NAME:

LAST NAME:

ADDRESS:

EMAIL ID:

PASSWORD:

USER ROLE: TENANT/OWNER:

REGISTER

Tenant Page

Welcome to Property Rental

Rent or buy properties hassle-free

Enter location

Filter Search

Rent Buy

4.4 DATABASE DESIGN

A database is a collection of data that has been organized to make it simple to manage and update. Information security could be one of the main aims of any database. The database design process is divided into two steps. The goal of the first step is to gather user requirements to create a database that as clearly as possible satisfies user needs. It is known as information-level design and is carried out without the aid of any DBMS. An information-level design is changed to a specific DBMS design that will be used to develop the system in the following stage. The physical-level design phase is where the characteristics of the specific DBMS are considered.

4.4.1 RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

A Relational Database Management System (RDBMS) is a critical software application for organizing and managing data in a structured manner. It stores data in tables with rows and columns, where each row represents a record, and each column is a specific attribute or field. RDBMS systems like MySQL, Oracle, or Microsoft SQL Server ensure data integrity, enforce relationships between tables, and allow for efficient data retrieval and manipulation through Structured Query Language (SQL). These systems are widely used in various applications, such as e-commerce websites, financial systems, and inventory management, due to their robustness, scalability, and ability to handle complex data structures, making them essential for modern data-driven environments.

A Relational Database Management System (RDBMS) is a cornerstone of modern data management. It structures data into tables, where each row represents a unique entry, and each column corresponds to a specific attribute, ensuring a logical and organized storage system. RDBMS systems employ complex algorithms for data retrieval and manipulation, guaranteeing data consistency and adherence to predefined relationships between tables. Popular RDBMS software, including PostgreSQL, MySQL, and Microsoft SQL Server, provides robust features for transactions, data security, and scalability. These systems are integral to a myriad of applications, from healthcare records and customer databases to inventory control and financial platforms, supporting the efficient storage, retrieval, and analysis of vast datasets, making them essential tools in today's data-driven world.

4.4.2 NORMALIZATION

Normalization is a database design technique used in relational database management systems (RDBMS) to eliminate data redundancy and improve data integrity. It involves organizing data in a way that reduces data duplication and ensures that relationships between tables are defined and maintained.

The primary goals of normalization are:

Minimizing Data Redundancy: By breaking down data into separate tables and ensuring that each piece

of data is stored only once, normalization helps reduce the chances of data inconsistencies or errors.

Ensuring Data Integrity: Normalization enforces the rules of referential integrity, which means that relationships between tables are well-defined and maintained. This ensures that data remains accurate and consistent.

Normalization typically involves dividing a database into multiple related tables and using primary keys and foreign keys to establish relationships between these tables. There are several normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF), each with specific rules and requirements. The level of normalization achieved depends on the specific needs of the database and the trade-off between data redundancy and query performance.

By applying normalization principles, designers can create efficient and reliable database structures that support data integrity and ease data maintenance and manipulation.

There are several normal forms (NF) that define specific rules and requirements for achieving progressively higher levels of normalization. Here are the most common normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF):

First Normal Form (1NF):

- Each table has a primary key.
- All columns contain atomic (indivisible) values.
- There are no repeating groups or arrays in columns.

Second Normal Form (2NF):

- The table is in 1NF.
- All non-key attributes are fully functionally dependent on the entire primary key. In other words, all non-key attributes must be dependent on the entire primary key, not just part of it.

Third Normal Form (3NF):

- The table is in 2NF.
- There is no transitive dependency, meaning that non-key attributes are not dependent on other non-key attributes.

Boyce-Codd Normal Form (BCNF):

- A stricter version of 3NF.
- It enforces that every non-trivial functional dependency involves a super key.

Fourth Normal Form (4NF):

- Addresses multi-valued dependencies.
- Eliminates any multi-valued dependencies within the data.

Fifth Normal Form (5NF):

- Also known as Project-Join Normal Form (PJ/NF).
- Handles cases where data can be derived by joining multiple tables.

4.4.3 SANITIZATION

In Python Django, sanitization refers to the process of cleaning and validating data to ensure that it is safe and free from malicious content before it is used or stored in a database. Sanitization is a crucial security practice to prevent various forms of attacks, such as cross-site scripting (XSS) and SQL injection, which can compromise the security and integrity of a web application.

4.4.4 INDEXING

Indexing in Django is a fundamental database optimization technique. It involves creating data structures, or indexes, on specific fields to expedite data retrieval. These indexes act as signposts that enable the database to swiftly locate and fetch relevant data, especially in tables with substantial amounts of information. Django offers automatic index creation for primary keys, unique fields, and foreign keys. Additionally, developers can define custom indexes for fields frequently used in filtering, sorting, or searching. The choice of proper indexing plays a pivotal role in improving query performance, resulting in more responsive and scalable web applications. It's essential to consider application-specific query patterns and create indexes accordingly, as well as to understand the capabilities and limitations of the chosen database backend. Effective indexing is a cornerstone of efficient database operations in Django, contributing to enhanced application performance.

4.5 TABLE DESIGN

1.tbl_customuser

Primary key: **user_id**

No	Field name	Datatype	Key Constraints	Description of the field
1	user_id	IntegerField(20)	PK	Users_id
2	username	Charfield(20)	Not Null	Users Username
3	Password	Charfield(20)	Not Null	Password of user
4	Email	EmailField(20)	Unique	Email_id of user
5	role	CharField(20)	Not Null	Role of the registered user

2.tbl _userprofile(propertyowner)Primary key: **UserProfile_id**Foreign key: **user_id** references table **tbl_customuser**

No	Field name	Datatype	Key Constraints	Description of the field
1	UserProfile_id	IntegerField(20)	PK	Unique id for Userprofile
2	user_id	IntegerField(20)	FK	Unique id for User
3	fullname	Charfield(20)	Null	fullname of user
4	date_of_birth	date	Null	d.o.b of user
5	gender	Charfield (20)	Null	gender of user
6	phone	IntegerField(15)	Null	Phone number of user
7	Photo_id	file	Null	Photo of the user
8	address	CharField(20)	Null	Address of the user

3.tbl _profile(tenant)Primary key: **profile_id**Foreign key: **user_id** references table **tbl_customuser**

No	Field name	Datatype	Key Constraints	Description of the field
1	Profile_id	IntegerField(20)	PK	Unique id for Userprofile
2	user_id	IntegerField(20)	FK	Unique id for User
3	fullname	Charfield(20)	Null	fullname of user
4	date_of_birth	date	Null	d.o.b of user
5	gender	Charfield (20)	Null	gender of user
6	phone	IntegerField(15)	Null	Phone number of user
7	Photo_id	file	Null	Photo of the user
8	address	CharField(20)	Null	Address of the user

4.tbl _property

Primary key: **property_id**

Foreign key: **userprofile_id** references table **tbl _userprofile**

No	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Property_id	IntegerField(10)	PK	Unique id for Property
2	Property_type	CharField	Null	Type of Property
3	address	Charfield(20)	Not Null	Property address
4	Monthly_rent	Decimalfield(20)	Not Null	Monthly rent of property
5	Security_deposit	Decimalfield (20)	Not Null	Security deposit of property
6	Lease_duration	Charfield (2000)	Not Null	Property lease duration
7	Availability_date	DateField	Not Null	Date the property is available
8	Approval_status	CharField(20)	Not Null	Admin Property Approval
9	Userprofile_id	IntegerField(10)	FK	Property owner id

5. tbl _Propertyimage

Primary key: **pimage_id**

Foreign key: **property_id** references table **tbl _property**

No	Field name	Datatype (Size)	Key Constraints	Description of the field
1	pimage_id	IntegerField(10)	PK	Unique id for images
2	Category	CharField(200)	Null	Property image category
3	image	ImageField()	Not Null	Category wise image
4	Property_id	IntegerField(10)	FK	Id of the property

6. tbl_AmenityPrimary key: **amenity_id**Foreign key: **property_id** references table **tbl_property**

No	Field name	Datatype(Size)	Key Constraints	Description of the field
1	amenity_id	IntegerField(10)	PK	Property amenityid
2	Sqfootage	Positiveintegerfield(10)	Not null	Property quarefoot
3	Bedrooms	Positiveintegerfield(10)	Not null	Noofbedrooms
4	Bathrooms	Positiveintegerfield(10)	Not null	Noofbathrooms
5	Stories	Positiveintegerfield(10)	Not null	Property storiesno
6	Acresofland	Decimalfield(10)	Not null	Acresof property type land
7	Rentspace	CharField(200)	Not null	Full or partial
8	noofrooms	Positiveintegerfield(10)	Not null	Number of rooms
9	parkingspace	Textfield(10)	Not null	Yes or no for commercial property
10	purpose	Textfield(10)	Not null	land property type purpose
11	acresorcent	Decimalfield(10)	Not null	Acres/cent the property in
12	Property_id	IntegerField(10)	FK	Property id

7. tbl _propertydocument

Primary key: **doc_id**

Foreign key: **property_id** references table **tbl _property**

No	Field name	Datatype (Size)	Key Constraints	Description of the field
1	doc_id	IntegerField(10)	PK	Unique id for property document
2	Document_type	CharField(10)	NotNull	Property document type
3	document	FileField(10)	NotNull	Property document
4	Property_id	IntegerField(10)	NotNull	Property id

8.tbl _rentalrequest

Primary key: **request_id**

Foreign key: **property_id** references table **tbl _property**

Foreign key: **profile_id** references table **tbl _profile**

No	Field name	Datatype	Key Constraints	Description of the field
1	request_id	IntegerField(20)	PK	Unique id for each rental request
2	Property_id	IntegerField(20)	FK	Property id
3	Profile_id	IntegerField(20)	FK	Tenant who had made rental request
4	Status	CharField(20)	NotNull	Pending/accepted/rejected
5	Created_at	DateTimeField	NotNull	Time rental request has created

9. tbl_leaseagreement

Primary key: **agreement_id**

Foreign key: **property_id** references table **tbl_property**

No:	Field name	Datatype	Key Constraints	Description of the field
1	agreement_id	IntegerField(20)	PK	Unique id for each agreement
2	Property_id	IntegerField(20)	FK	Property id
3	Start_date	DateField	NotNull	rentalstartdate
4	End_date	DateField	NotNull	Rentalenddate
5	Rent_amount	DecimalField(20)	NotNull	Total rentamount each month
6	Security_amount	DecimalField	Not Null	Initial security amount

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing is a way to check if the computer program works like it's supposed to. We use testing to make sure the software does what it is supposed to do. Validation means checking or testing things like software to make sure they meet the requirements and standards they are supposed to follow. Software testing is a way to check if a program works well. It goes along with other methods like checking and walking through the program. Validation means making sure that what the user wanted is what they got. There are several rules that can serve as testing objectives.

They are:

Testing is a process of executing a program with the intent of finding an error.

- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If a test works well and follows its goals, it can find mistakes in the software. The test showed that the computer program is working like it's supposed to and is doing well.

There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

5.2 TEST PLAN

A test plan suggests several required steps that need be taken to complete various testing methodologies. The activity that is to be taken is outlined in the test plan. A computer program, its documentation, and associated data structures are all created by software developers. It is always the responsibility of the software developers to test each of the program's separate components to make sure it fulfills the purpose for which it was intended. To solve the inherent issues with allowing the builder evaluate what they have developed, there is an independent test group (ITG). Testing's precise goals should be laid forth in quantifiable language. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test.

The levels of testing include:

- Unit testing
- Integration Testing
- Data validation Testing & Output Testing

5.2.1 UNIT TESTING

Unit testing concentrates verification efforts on the software component or module, which is the smallest unit of software design. The component level design description is used as a guide when testing crucial control paths to find faults inside the module's perimeter. The level of test complexity and the untested area determined for unit testing. Unit testing is white-box focused, and numerous components may be tested simultaneously. To guarantee that data enters and exits the software unit under test properly, the modular interface is tested. To make sure that data temporarily stored retains its integrity during each step of an algorithm's execution, the local data structure is inspected. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Before starting any other test, tests of data flow over a module interface are necessary. All other tests are irrelevant if data cannot enter and depart the system properly. An important duty during the unit test is the selective examination of execution pathways. Error circumstances must be foreseen in good design, and error handling paths must be put up to cleanly reroute or halt work when an error does arise. The final step of unit testing is boundary testing. Software frequently fails at its limits.

In the Sell-Soft System, unit testing was carried out by treating each module as a distinct entity and subjecting them to a variety of test inputs. The internal logic of the modules had some issues, which were fixed. Each module is tested and run separately after coding. All unused code was eliminated, and it was confirmed that every module was functional and produced the desired outcome

5.2.2 INTEGRATION TESTING

Integration testing is a methodical approach for creating the program's structure while also carrying out tests to find interface issues. The goal is to construct a program structure that has been determined by design using unit tested components. The program is tested. Correction is challenging since the size of the overall program makes it challenging to isolate the causes. As soon as these mistakes are fixed, new ones arise, and the process repeats itself in an apparently unending cycle. All the modules were integrated after unit testing was completed in the system to check for an interface inconsistency. A distinctive program structure also developed when discrepancies in program structures.

5.2.3 VALIDATION TESTING OR SYSTEM TESTING

The testing process comes to an end here. This involved testing the entire system in its entirety, including all forms, code, modules, and class modules. Popular names for this type of testing include

system tests and black box testing. The functional requirements of the software are the main emphasis of the black box testing approach. That example, using Black Box testing, a software engineer can create sets of input conditions that will fully test every program requirement. The following sorts of problems are targeted by black box testing: erroneous or missing functions, interface errors, data structure or external data access errors, performance errors, initialization errors, and termination errors.

5.2.3 OUTPUT TESTING OR USER ACCEPTANCE TESTING

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required.

This done with respect to the following points:

- Input Screen Designs,
- Output Screen Designs,

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future.

5.2.4 AUTOMATION TESTING

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. Essentially, it's a test to double-check that the equipment or software does exactly what it was designed to do. It tests for bugs, defects, and any other issues that can arise with product development. Although some types of testing, such as regression or functional testing can be done manually, there are greater benefits of doing it automatically. Automation testing can be run at any time of the day. It uses scripted sequences to examine the software. It then reports on what has been found, and this information can be compared with earlier test runs. Automation developers generally write in the following programming languages: C#, JavaScript, and Ruby.

5.2.5 SELENIUM TESTING

Selenium is an open-source automated testing framework used to verify web applications across different browsers and platforms. Selenium allows for the creation of test scripts in various

programming languages such as Java, C#, and Python. Jason Huggins, an engineer at Thought Works, developed Selenium in 2004 while working on a web application that required frequent testing. He created a JavaScript program called "JavaScriptTestRunner" to automate browser actions and improve testing efficiency. Selenium has since evolved and continues to be developed by a team of contributors. In addition to Selenium, another popular tool used for automated testing is Cucumber. Cucumber is an open-source software testing framework that supports behavior-driven development (BDD). It allows for the creation of executable specifications in a human-readable format called Gherkin. One of the advantages of using Cucumber is its ability to bridge the gap between business stakeholders and technical teams. By using a common language, Cucumber facilitates effective communication and collaboration during the testing process. It promotes a shared understanding of the requirements and helps ensure that the developed software meets the intended business goals. Cucumber can be integrated with Selenium to combine the benefits of both tools. Selenium is used for interacting with web browsers and automating browser actions, while Cucumber provides a structured framework for organizing and executing tests. This combination allows for the creation of end-to-end tests that verify the behavior of web applications across different browsers and platforms, using a business-readable and maintainable format.

Test Case 1

Code

Feature: Test login Functionality

Scenario: Check login is successful with valid credentials

- Given browser is open
- And user is on login page
- When user enters username and password
- And user clicks on login
- Then user is navigated to the owner page

```
package stepdefinition;  
  
import io.cucumber.java.en.Given;  
import io.cucumber.java.en.And;  
import io.cucumber.java.en.When;  
import io.cucumber.java.en.Then;  
  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;
```

```
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import static org.junit.Assert.assertEquals;

public class loginsteps {
    private WebDriver driver;

    boolean testPassed = true; // Initialize a flag to check test status

    @Given("browser is open")
    public void browserIsOpen() {
        // Set up WebDriver
        System.setProperty("webdriver.chrome.driver", "C:\\\\chromedriver-
win64\\\\chromedriver.exe");
        driver = new ChromeDriver();
    }

    @And("user is on login page")
    public void userIsOnLoginPage() {
        driver.get("http://127.0.0.1:8000/login/");
    }

    @When("user enters username and password")
    public void userEntersUsernameAndPassword() {
        WebElement usernameInput = driver.findElement(By.id("username"));
        WebElement passwordInput = driver.findElement(By.id("password"));

        usernameInput.sendKeys("rijulrojo");
        passwordInput.sendKeys("rR@12345");
    }

    @And("user clicks on login")
    public void userClicksOnLogin() {
        WebElement loginButton = driver.findElement(By.id("login")); // Replace "login-button"
with the actual ID of the login button
        loginButton.click();
    }

    @Then("user is navigated to the owner page")
    public void navigateToHomePage() {
```

```
try {
    // Add a delay to ensure the page loads completely
    // Thread.sleep(2000);
    // Check if the test is passed
    if (driver.getCurrentUrl().contains("http://127.0.0.1:8000/ownerpage")) {
        System.out.println("Test Passed: User is on the home page");
    } else {
        System.out.println("Test Failed: User is not on the home page");
        testPassed = false;
    }

    // Assert the test status using JUnit's Assert class
    assertEquals(true, testPassed);
} finally {
    // Close the driver after all assertions are made
    if (driver != null) {
        driver.quit();
    }
}
}
```

Screenshot

```
Scenario: Check login is successful with valid credentials # src/test/resources/Features/login.feature:3
  Given browser is open # stepdefinition.loginsteps.browserIsOpen()
  And user is on login page # stepdefinition.loginsteps.userIsOnLoginPage()
  When user enters username and password # stepdefinition.loginsteps.userEntersUsernameAndPassword()
  And user clicks on login # stepdefinition.loginsteps.userClicksOnLogin()
Test Passed: User is on the home page
  Then user is navigated to the owner page # stepdefinition.loginsteps.navigateToHomePage()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m3.693s
```

Test Report

Test Case 1					
Project Name: PropertyRentals					
Login Test Case					
Test Case ID: Test_1			Test Designed By:Rijul Rojio		
Test Priority (Low/Medium/High): High			Test Designed Date: 04-12-2023		
Module Name: Login Screen			Test Executed By : Mr. Jinson Devis		
Test Title : User Login			Test Execution Date: 04-12-2023		
Description: Verify login with valid username and password					
Pre-Condition :					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Username	Username: rijulrojio	User should be able to login	User is logged in and navigated to corresponding Owner Page	Pass
3	Provide Valid Password	Password: rR@12345			
4	Click on Login button				
Post-Condition: User is validated with database and successfully login into account. The Account session details are logged in database					

Test Case 2:**Code**

```
import time
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By

class LoginTest(unittest.TestCase):
```

```
def setUp(self):
    self.driver = webdriver.Chrome()

def test_successful_login(self):
    self.driver.get('http://127.0.0.1:8000/login')
    username_input = self.driver.find_element(By.ID, 'username')
    password_input = self.driver.find_element(By.ID, 'password')

    # Replace 'valid_username' and 'valid_password' with actual credentials
    username_input.send_keys('rijulrojo')
    password_input.send_keys('rR@12345')

    login_button = self.driver.find_element(By.ID, 'login')
    login_button.click()

    time.sleep(2) # Adjust the time as needed
    self.assertIn('http://127.0.0.1:8000/ownerpage', self.driver.current_url.lower())

    # Click on the "Dashboard" link
    dashboard_link = self.driver.find_element(By.LINK_TEXT, 'Dashboard')
    dashboard_link.click()

    # Check if redirected to the owner's dashboard
    self.assertIn('http://127.0.0.1:8000/ownerpg', self.driver.current_url.lower())

    # Click on the "Add Properties" link
    add_properties_link = self.driver.find_element(By.LINK_TEXT, 'Add
Properties')
    add_properties_link.click()
    property_type_dropdown = self.driver.find_element(By.ID, 'property_type')
    address_input = self.driver.find_element(By.ID, 'address')
    monthly_rent_input = self.driver.find_element(By.ID, 'monthly_rent')
    security_deposit_input = self.driver.find_element(By.ID, 'security_deposit')
    lease_duration_dropdown = self.driver.find_element(By.ID, 'lease_duration')
    availability_date_input = self.driver.find_element(By.ID, 'availability_date')
```

```
property_type_dropdown.send_keys('Apartment')
address_input.send_keys('123 Main St')
monthly_rent_input.send_keys('1500')
security_deposit_input.send_keys('2000')
lease_duration_dropdown.send_keys('1 year')
availability_date_input.send_keys('01-02-2022')

# Submit the form
submit_button = self.driver.find_element(By.CSS_SELECTOR, '#add-
property-form button[type="submit"]')
submit_button.click()

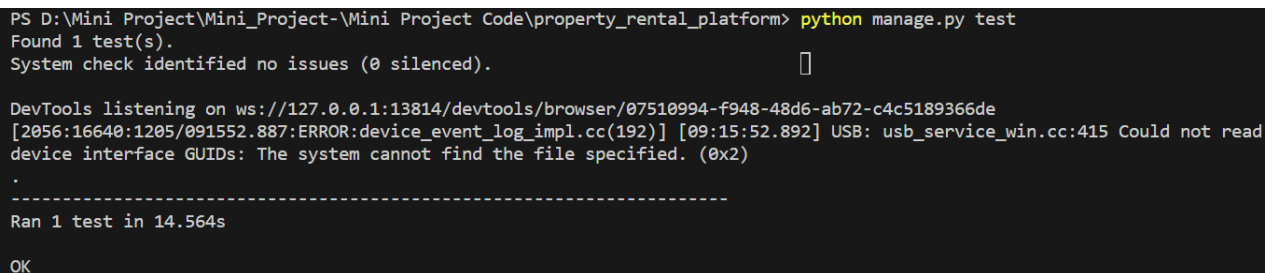
time.sleep(2) # Adjust the time as needed

# Now you can add assertions to check if the property was added successfully
# For example, check for success messages or check if the property appears in
the list

def tearDown(self):
    self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

Screenshot



```
PS D:\Mini Project\Mini Project\Mini Project Code\property_rental_platform> python manage.py test
Found 1 test(s).
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:13814/devtools/browser/07510994-f948-48d6-ab72-c4c5189366de
[2056:16640:1205/091552.887:ERROR:device_event_log_impl.cc(192)] [09:15:52.892] USB: usb_service_win.cc:415 Could not read
device interface GUIDs: The system cannot find the file specified. (0x2)
.
-----
Ran 1 test in 14.564s
OK
```

Test report

Test Case 2					
Project Name: PropertyRentals					
Test Case ID: Test_2			Test Designed By: Rijul Rojio		
Test Priority (Low/Medium/High): High			Test Designed Date: 04-12-2023		
Module Name: AddProperty			Test Executed By : Mr.Jinson Devis		
Test Title:Adding Properties			Test Execution Date: 04-12-2023		
Description: Property Owner adding their properties					
Pre-Condition : User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Username	Username: rijulrojio	User should be able to login	User is logged in and navigated to corresponding Owner Page	Pass
3	Provide Valid Password	Password: rR@12345			
4	Click on Login button				
5	Click on Dashboard button			Owner dashboard is displayed	Pass
6	Click on Add Property	By.ID,'property_type','address','monthly_rent','security_deposit','lease_duration','availability_date		Property is added succesfully	Pass
Post-Condition: Property is Added Sucessfully					

Test Case 3:**Code**

```
import time
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By

class LoginTest(unittest.TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()

    def test_successful_login(self):
        self.driver.get('http://127.0.0.1:8000/login')
        username_input = self.driver.find_element(By.ID, 'username')
        password_input = self.driver.find_element(By.ID, 'password')

        # Replace 'valid_username' and 'valid_password' with actual credentials
        username_input.send_keys('jubil')
        password_input.send_keys('jJ@12345')

        login_button = self.driver.find_element(By.ID, 'login')
        login_button.click()
        time.sleep(2) # Adjust the time as needed
        # Check if redirected to the tenantpage
        self.assertIn('http://127.0.0.1:8000/tenantpage', self.driver.current_url.lower())

        # Click on the "Dashboard" link
        dashboard_link = self.driver.find_element(By.LINK_TEXT, 'Dashboard')
        dashboard_link.click()

        time.sleep(2) # Adjust the time as needed
        self.assertIn('http://127.0.0.1:8000/tenantpg', self.driver.current_url.lower())
        property_type_dropdown = self.driver.find_element(By.ID, 'property_type')
```



```
min_rent_input = self.driver.find_element(By.NAME, 'min_rent')
max_rent_input = self.driver.find_element(By.NAME, 'max_rent')
search_button = self.driver.find_element(By.CSS_SELECTOR, '#property-
search-form button[type="submit"]')
```

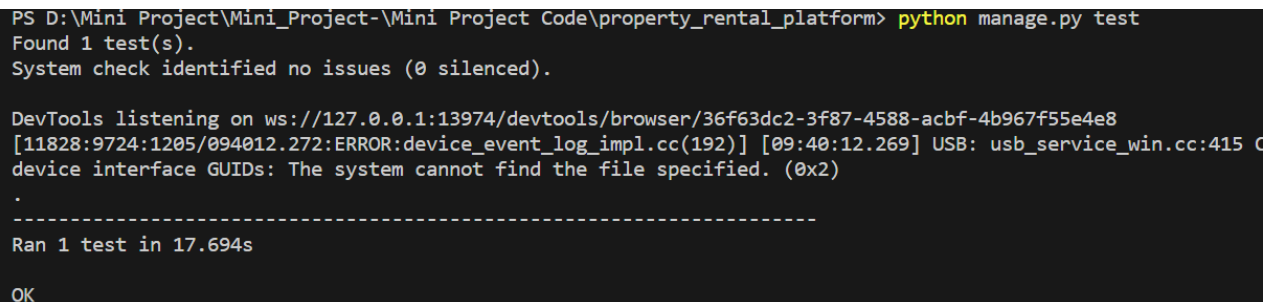
```
# Replace these values with the desired search criteria
property_type_dropdown.send_keys('Apartment')
min_rent_input.send_keys('1000')
max_rent_input.send_keys('2000')
search_button.click()

time.sleep(2) # Adjust the time as needed

def tearDown(self):
    self.driver.quit()
```

```
if __name__ == "__main__":
    unittest.main()
```

Screenshot



```
PS D:\Mini Project\Mini_Project-\Mini Project Code\property_rental_platform> python manage.py test
Found 1 test(s).
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:13974/devtools/browser/36f63dc2-3f87-4588-acbf-4b967f55e4e8
[11828:9724:1205/094012.272:ERROR:device_event_log_impl.cc(192)] [09:40:12.269] USB: usb_service_win.cc:415 C
device interface GUIDs: The system cannot find the file specified. (0x2)
.
-----
Ran 1 test in 17.694s

OK
```

Test report

Test Case 3					
Project Name: PropertyRentals					
Test Case ID: Test_3			Test Designed By: Rijul Rojio		
Test Priority (Low/Medium/High): High			Test Designed Date: 04-12-2023		
Module Name: Searching			Test Executed By : Mr.Jinson Devis		
Test Title : Property Searching			Test Execution Date: 04-12-2023		
Description:The tenant can search for properties filtered.					
Pre-Condition : User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Username	Username: jubil	User should be able to login	User is logged in and navigated to corresponding Tenant Page	Pass
3	Provide Valid Password	Password: rR@12345			
4	Click on Login button				
5	Click on Dashboard button			Tenant dashboard is displayed	Pass
6	Search Properties	BY.ID,'property_type'		Selected property type is displayed if available	Pass
7	Enter the rent amount range	BY.NAME,'min_rent','max_rent		The properties under this particular price range displayed	Pass
Post-Condition: Properties according to the filters is displayed if available.					

Test Case 4:**Code**

```
import time
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
class LoginTest(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
    def test_successful_login(self):
        self.driver.get('http://127.0.0.1:8000/login')
        username_input = self.driver.find_element(By.ID, 'username')
        password_input = self.driver.find_element(By.ID, 'password')
        # Replace 'valid_username' and 'valid_password' with actual credentials
        username_input.send_keys('rijulrojio')
        password_input.send_keys('rR@12345')

        login_button = self.driver.find_element(By.ID, 'login')
        login_button.click()
        time.sleep(2) # Adjust the time as needed
        # Check if redirected to the ownerpage
        self.assertIn('http://127.0.0.1:8000/ownerpage', self.driver.current_url.lower())

        # Click on the "Dashboard" link
        dashboard_link = self.driver.find_element(By.LINK_TEXT, 'Dashboard')
        dashboard_link.click()

        time.sleep(2) # Adjust the time as needed

        # Check if redirected to the ownerpg
        self.assertIn('http://127.0.0.1:8000/ownerpg', self.driver.current_url.lower())

        # Click on the "Manage Properties" link
        manage_properties_link = self.driver.find_element(By.LINK_TEXT, 'Manage
```

Properties')

```
        manage_properties_link.click()

        time.sleep(2) # Adjust the time as needed

        # Check if redirected to the manageprop
        self.assertIn('http://127.0.0.1:8000/manageprop',
self.driver.current_url.lower())

        # Click on the "Accept Request" button
        accept_request_button = self.driver.find_element(By.CSS_SELECTOR, 'form
button.btn-primary')
        accept_request_button.click()

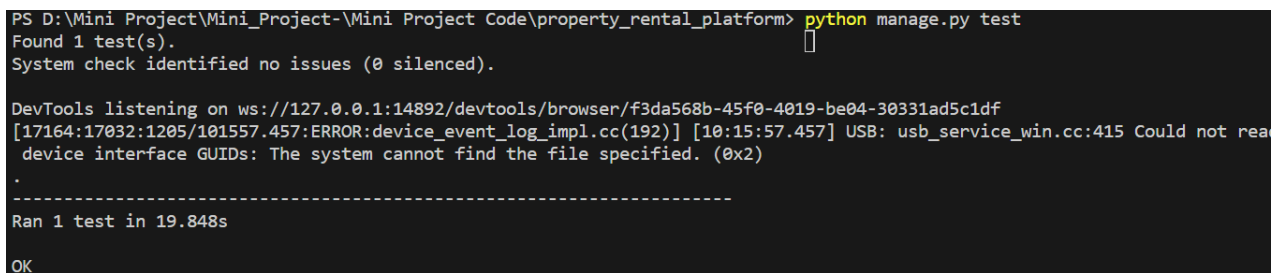
        time.sleep(2) # Adjust the time as needed

        # Now you can add assertions to check if the acceptance was successful
        # For example, check for success messages or changes in the UI

    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

Screenshot



```
PS D:\Mini Project\Mini Project Code\property_rental_platform> python manage.py test
Found 1 test(s).
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:14892/devtools/browser/f3da568b-45f0-4019-be04-30331ad5c1df
[17164:17032:1205/101557.457:ERROR:device_event_log_impl.cc(192)] [10:15:57.457] USB: usb_service_win.cc:415 Could not read
device interface GUIDs: The system cannot find the file specified. (0x2)
.
-----
Ran 1 test in 19.848s
OK
```

Test report

Test Case 4					
Project Name: PropertyRentals					
Test Case ID: Test_4			Test Designed By: Rijul Rojio		
Test Priority (Low/Medium/High): High			Test Designed Date: 04-12-2023		
Module Name:Rental Requests			Test Executed By : Mr. Jinson Devis		
Test Title : Accept or reject the rental requests			Test Execution Date: 04-12-2023		
Description: Property Owner can accept the rental request made by the tenant					
Pre-Condition :					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Username	Username: rijulrojio	User should be able to login	User is logged in and navigated to corresponding Owner Page	Pass
3	Provide Valid Password	Password: rR@12345			
4	Click on Login button				
5	Click on Dashboard button			Owner dashboard is displayed	Pass
6	Click on manage properties			The Owner can see his properties in the system with options to manage it	Pass
7	Click on accept request			The property owner can see rental request that came for their properties,can accept request or don't.	Pass
Post-Condition: Property owner can accept rental requests.					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion. Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover. The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.

6.2 IMPLEMENTATION PROCEDURES

The implementation procedures for this project involve a systematic and phased approach to bring the online crime reporting portal to fruition. This complex system, designed to cater to the needs of various stakeholders, requires careful planning and execution.

To commence the implementation, the project team will initiate a detailed requirements analysis phase. This stage involves comprehensive discussions with law enforcement officials, control room staff, prison wardens, and potential end-users to determine their specific needs and expectations.

The results of this analysis will inform the development of a detailed system specification and design.

The development phase will follow, involving the creation of the portal's architecture, databases, and user interfaces. It's during this stage that the functionalities outlined in the project's scope, including user management, crime reporting, and communication features, will be integrated into the system. The portal will be designed with a user-friendly interface to ensure ease of use for all categories of users.

Simultaneously, a dedicated team will work on the incorporation of advanced technologies, such as machine learning capabilities for crime trend analysis and predictive features. These technologies will be implemented carefully to ensure the portal's robustness and security.

Once the development phase is complete, rigorous testing will be conducted to identify and rectify any bugs or issues. The portal will undergo extensive quality assurance and user acceptance testing to ensure that it meets all the necessary requirements and is free of vulnerabilities.

Deployment of the system will be carried out in a controlled manner, ensuring minimal disruption to the existing processes, and allowing for gradual adaptation by the involved stakeholders. Comprehensive training sessions will be conducted to familiarize law enforcement personnel, control room staff, prison wardens, and registered users with the portal's functionality and features. Post-deployment, the project team will continue to provide support and maintenance, addressing any issues that may arise and making necessary improvements as the system matures. Regular updates and security measures will be implemented to safeguard user data and maintain the portal's efficiency.

6.2.1 USER TRAINING

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer-based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database, and call up routine that will produce reports and perform other necessary functions

6.2.2 TRAINING ON THE APPLICATION SOFTWARE

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date

entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy.

6.2.3 SYSTEM MAINTENANCE

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than “Finding Mistakes”.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The Property Renting Platform endeavors to redefine how property owners and tenants interact, creating a digital haven for property rentals. Through features such as advanced analytics for Admins, efficient property listing management for Owners, and detailed property search for Tenants, the platform seeks to address the evolving challenges in the fast-paced urban environment. By prioritizing user satisfaction, inclusivity, and the utilization of advanced technologies, the Property Renting Platform aims to establish itself as a cornerstone in the landscape of online property rental platforms. The proposed system's strengths lie in its user-friendly interfaces, advanced functionalities, and a commitment to providing a comprehensive solution for the diverse needs of property owners and tenants alike.

7.2 FUTURE SCOPE

For the PropertyRentals platform, the future scope is equally promising, marked by avenues for innovation and growth. By integrating advanced technologies such as machine learning and data analytics, the platform can refine its property recommendation system, offering personalized suggestions aligned with user preferences and behaviors. Expanding community engagement features to include forums, virtual property tours, and user-generated content can foster a sense of collaboration among property owners and tenants. Global expansion is a viable prospect, with the platform accommodating multilingual support to cater to diverse markets and communities. Strategic partnerships with real estate agencies, property developers, and local authorities can enhance the platform's reach and credibility. Additionally, exploring the potential for integrating augmented reality (AR) or virtual reality (VR) for immersive property viewing experiences can further set the platform apart in the competitive landscape of property rentals. Overall, the future of the Property Rentals platform holds exciting possibilities for technological advancements, community building, and extended market reach.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Harvard Business Review - Unlocking Value through Questions:
- AlsoAsked - People Also Ask keyword research tool: A tool to find questions people also ask related to a specific topic.
- Google Support - How to search on Google: Tips on effective Google searches, including finding quick answers.
- Qualtrics - Multiple Choice Question: Information about the multiple-choice question type for surveys.
- MyGreatLearning - 180+ SQL Interview Questions and Answers in 2023: A compilation of SQL interview questions for job seekers

WEBSITES:

- <https://www.grafiati.com/>
- <https://annas-archive.org/>
- <https://chat.openai.com/>

CHAPTER 9

APPENDIX

9.1 Sample Code

Login.html

```
<!-- login.html -->
{ % load static % }
<!DOCTYPE html>
<html>
<head>
    <title>Login Page</title>
    <style>
        /* Reset some default styles */
        body, h1, h2, p, ul, li {
            margin: 0;
            padding: 0;
        }

        /* CSS for Header */
        body {
            font-family: Arial, sans-serif;
            background-image: url('{ % static "images/background-image.jpg" % }');
            background-size: cover;
            background-repeat: no-repeat;
            background-attachment: fixed;
            margin: 0; /* Remove default margin for mobile */
            background-color: antiquewhite;
        }

        header {
            background-color: rgba(64, 140, 227, 0.7);
            color: #fff;
            text-align: left;
            padding: 20px;
        }

        a {
```

```
text-decoration: none; /* Remove underline */
color: #f2f5f3; /* Change link color to orange; you can use any color you like */
}

header h1 {
    font-size: 36px;
    margin-bottom: 10px;
}

/* CSS for Main Content */
main {
    background-image: url(""); /* Add your background image URL here */
    padding: 20px;
    background-color: rgba(136, 184, 138, 0.9);
    border-radius: 10px;
    margin: 20px;
    min-height: 80vh; /* Ensure main content takes up at least 80% of the viewport
height */
    display: flex;
    flex-direction: column;
    justify-content: space-between;
}

/* CSS for Card Container */
.card-container {
    background-color: rgba(255, 255, 255, 0.9);
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
}

/* CSS for Login Form */
.login-form {
    padding: 20px;
```



```
}

.login-form h2 {
    font-size: 24px;
    margin-bottom: 20px;
}

/* Form fields and buttons styling */
.form-group {
    margin-bottom: 15px;
}

.form-group label {
    font-weight: bold;
    display: block;
    margin-bottom: 5px;
}

.form-group input[type="text"],
.form-group input[type="password"] {
    width: 100%;
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

/* CSS for Buttons */
.btn {
    background-color: #258dfc;
    color: #fff;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
}
```

```
        transition: background-color 0.3s;
    }

    .btn:hover {
        background-color: #89b8a3;
    }

    /* CSS for Forgot Password Button */
    .forgot-password {
        text-align: left;
        margin-top: 10px;
    }

    .forgot-password a {
        text-decoration: none;
        color: #007BFF;
    }

    /* CSS for Footer */
    footer {
        background-color: rgba(0, 0, 0, 0.7);
        color: #fff;
        text-align: center;
        padding: 10px;
        width: 100%;
    }

    /* CSS for Large Message Box */
    .large-message-box {
        display: none;
        position: fixed;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        background-color: #fff;
```

```
        border: 1px solid #ccc;
        border-radius: 5px;
        box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
        padding: 20px;
        z-index: 9999;
    }

    /* CSS for Message Box Content */
    .message-content {
        font-size: 18px;
        text-align: center;
    }

    /* CSS for Close Button */
    .close-button {
        position: absolute;
        top: 10px;
        right: 10px;
        cursor: pointer;
        color: #333;
    }

    /* Media Queries for Responsive Design */
    @media screen and (max-width: 768px) {
        .card-container {
            padding: 10px;
        }

        .login-form h2 {
            font-size: 20px;
        }

        .form-group input[type="text"],
        .form-group input[type="password"] {
            width: 100%;
        }
    }
```

```
        padding: 8px;
    }

    .btn {
        padding: 8px 16px;
    }
}
</style>

<link rel="shortcut icon" href="{% static 'images/Key.png' %}" type="">

</head>
<body>
    <header>

        <h1><a href="{% url 'index' %}">PROPERTY RENTALS</a></h1>
    </header>

    <main>
        {% if messages %}
        <div class="large-message-box">
            <span class="close-button" onclick="closeMessageBox()">&times;</span>
            <div class="message-content">
                {% for message in messages %}
                    {{ message|safe }}<br>
                {% endfor %}
            </div>
        </div>
        </div>
        {% endif %}

        <div class="card-container">
            <div class="login-form">
                <h2>Login</h2>
                <form action="{% url 'login' %}" method="post">
                    {% csrf_token %}
```

```
<div class="form-group">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required
placeholder="username">
</div>
<div class="form-group">
  <label for="password">Password:</label>
  <input type="password" id="password" name="password" required
placeholder="password">
</div>
<div>
  <button type="submit" class="btn" id="login">Login</button>
</div>
</form>
<div class="forgot-password">
  <h4>Don't have an account? <a href="{ % url 'signup' % }" style="text-decoration: none;">Signup</a></h4>
</div>
<div class="forgot-password">
  <a href="{ % url 'password_reset' % }">Forgot Password?</a>
</div>
<div class="forgot-password">
  <a href="{ % url 'social:begin' 'google-oauth2' % }" class="google-signin-button">
    SignIn with Google
  </div>
</div>
</div>
</main>

<footer>
  <p>&copy; 2023 Property Rentals</p>
```

```
</footer>

<script>
    // Function to show the large message box
    function showMessageBox() {
        const messageBox = document.querySelector(".large-message-box");
        messageBox.style.display = "block";

        // Automatically hide the message box after 5 seconds (5000 milliseconds)
        setTimeout(() => {
            messageBox.style.display = "none";
        }, 2000); // Adjust the time (in milliseconds) as needed
    }

    // Call the showMessageBox function when the page loads
    window.addEventListener("load", showMessageBox);
</script>
```

```
</body>
```

```
</html>
```

Login Views.py

```
def login(request):
    if request.method == "POST":
        username = request.POST.get('username')
        password = request.POST.get('password')
        if username == "admin" and password == "admin":
            # Admin login credentials are fixed
            request.session['username'] = username
            return redirect("adminhome") # Redirect to the admin dashboard

        user = authenticate(request, username=username, password=password)

        if user is not None:
            auth_login(request, user)
            request.session['username'] = username
```

```
        if user.role == 'tenant':
            return redirect("tenantpage")
        elif user.role == 'owner':
            return redirect("ownerpage") # Create an 'ownerpage' URL
        elif user.role == 'admin':
            return redirect("adminhome")
    else:
        messages.error(request, "Invalid login credentials")

response = render(request, 'login.html')
response['Cache-Control'] = 'no-store, must-revalidate'
return response
```

Other View Functions

```
class EmailThread(threading.Thread):
    def __init__(self, email_message):
        self.email_message = email_message
        super().__init__() #Call the parent class's __init__ method

    def run(self):
        self.email_message.send()

def index(request):
    return render(request, 'index.html')

def signup(request):
    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
        confirmPassword = request.POST.get('confirm_password')
        role = request.POST.get('role') # Add role selection in your signup form
```

```
if CustomUser.objects.filter(email=email).exists():
    messages.error(request, "Already Registered Email!!!")
elif CustomUser.objects.filter(username=username).exists():
    messages.error(request, "Username already exists!!!")
elif password != confirmPassword:
    messages.error(request, "Passwords do not match!!!")
else:
    user = CustomUser(username=username, email=email, role=role)
    user.set_password(password)
    user.is_active=False #make the user inactive
    user.save()
    current_site=get_current_site(request)
    email_subject="Activate your account"
    message=render_to_string('activate.html',{
        'user':user,
        'domain':current_site.domain,
        'uid':urlsafe_base64_encode(force_bytes(user.pk)),
        'token':generate_token.make_token(user)

    })

    email_message=EmailMessage(email_subject,message,settings.EMAIL_HOST_USER,[email
    ],)

    EmailThread(email_message).start()
    messages.info(request,"Active your account by clicking the link send to your
    email")

    messages.success(request, "Waiting for Account Activation")
    return redirect("login")

    return render(request, 'signup.html')
def login(request):
    if request.method == "POST":
```



```
username = request.POST.get('username')
password = request.POST.get('password')
if username == "admin" and password == "admin":
    # Admin login credentials are fixed
    request.session['username'] = username
    return redirect("adminhome") # Redirect to the admin dashboard

user = authenticate(request, username=username, password=password)

if user is not None:
    auth_login(request, user)
    request.session['username'] = username
    if user.role == 'tenant':
        return redirect("tenantpage")
    elif user.role == 'owner':
        return redirect("ownerpage") # Create an 'ownerpage' URL
    elif user.role == 'admin':
        return redirect("adminhome")
else:
    messages.error(request, "Invalid login credentials")

response = render(request, 'login.html')
response['Cache-Control'] = 'no-store, must-revalidate'
return response

@login_required(login_url='login')
def logout(request):
    auth_logout(request) # Use the logout function to log the user out
    return redirect('index')

def tenant(request):
    if request.method == "POST":
        full_name = request.POST.get('fullname')
        date_of_birth = request.POST.get('dateofbirth')
        gender = request.POST.get('gender')
```

```
phone_number = request.POST.get('phoneno')
current_address = request.POST.get('currentaddress')
photoid_file = request.FILES.get('photoid')

# Get the user's profile based on the currently logged-in user
user_profile, created = Profile.objects.get_or_create(user=request.user)

# Update user_profile fields with the data from the POST request
user_profile.full_name = full_name
user_profile.date_of_birth = date_of_birth
user_profile.gender = gender
user_profile.phone_number = phone_number
user_profile.current_address = current_address

if photoid_file:
    user_profile.photo_id = photoid_file

# Save the updated user_profile
user_profile.save()

# Redirect or show a success message as needed
messages.success(request, "Profile updated successfully.")
return redirect("tenantpg")

user_profile = Profile.objects.get(user=request.user)
context = {"user_profile": user_profile}

if 'username' in request.session:
    response = render(request, 'tenantpage.html', context)
    response['Cache-Control'] = 'no-store, must-revalidate'
    return response
else:
    return redirect('index')

def owner(request):
```

```
if request.method == "POST":
    full_name = request.POST.get('fullname')
    date_of_birth = request.POST.get('dateofbirth')
    gender = request.POST.get('gender')
    phone_number = request.POST.get('phoneno')
    current_address = request.POST.get('currentaddress')
    photoid_file = request.FILES.get('photoid')

    # Get the user's profile based on the currently logged-in user
    user_profile, created = UserProfile.objects.get_or_create(user=request.user)

    # Update user_profile fields with the data from the POST request
    user_profile.full_name = full_name
    user_profile.date_of_birth = date_of_birth
    user_profile.gender = gender
    user_profile.phone_number = phone_number
    user_profile.current_address = current_address
    user_profile.photoid_file=photoid_file

    if photoid_file:
        user_profile.photo_id = photoid_file

    # Save the updated user_profile
    user_profile.save()

    # Redirect or show a success message as needed
    messages.success(request, "Profile updated successfully.")
    return redirect("ownerpg")

user_profile = UserProfile.objects.get(user=request.user)
context = {"user_profile": user_profile}
return render(request, "ownerpage.html", context)

if 'username' in request.session:
```

```
        response = render(request, 'ownerpage.html')
        response['Cache-Control'] = 'no-store, must-revalidate'
        return response
    else:
        return redirect('index')
def adminh(request):

    if 'username' in request.session:
        response = render(request, 'adminhome.html')
        response['Cache-Control'] = 'no-store, must-revalidate'
        return response
    else:
        return redirect('index')

def adminreg(request):
    role_filter = request.GET.get('role')

    # Filter users based on role and exclude superusers
    if role_filter:
        if role_filter == "tenant":
            profiles = CustomUser.objects.filter(Q(role=role_filter) & ~Q(is_superuser=True))
        else:
            profiles = CustomUser.objects.filter(Q(role=role_filter) & ~Q(is_superuser=True))
    else:
        profiles = CustomUser.objects.filter(~Q(is_superuser=True)) # Exclude superusers

    return render(request, 'adminregusers.html', {'profiles': profiles})
def tenantpg(request):
    filtered_properties = Property.objects.filter(approval_status='Approved')
    property_type = request.GET.get('property_type')
    min_rent = request.GET.get('min_rent')
    max_rent = request.GET.get('max_rent')

    # Create Q objects to build complex queries
```

```
filter_params = Q()

if property_type:
    filter_params &= Q(property_type=property_type)

if min_rent:
    filter_params &= Q(monthly_rent__gte=min_rent)

if max_rent:
    filter_params &= Q(monthly_rent__lte=max_rent)

# Apply the filters
filtered_properties = filtered_properties.filter(filter_params)

if 'username' in request.session:
    username = request.session['username']

    # Get the user's profile
    try:
        user_profile = Profile.objects.get(user__username=username)
    except Profile.DoesNotExist:
        user_profile = None

    # Check if a rental request has been accepted for each property
    for property in filtered_properties:
        property.is_rental_request_accepted =
property.rentalrequest_set.filter(tenant=user_profile, status='Accepted').exists()

    context = {
        'user_profile': user_profile,
        'properties': filtered_properties,
    }

    response = render(request, 'tenantpg.html', context)
```

```
        response['Cache-Control'] = 'no-store, must-revalidate'
        return response
    else:
        return redirect('index')
def manageprop(request):
    if 'username' in request.session:
        username = request.session['username']

    if request.method == 'POST':
        # Check if a property deletion request was submitted
        property_id_to_delete = request.POST.get('property_id_to_delete')
        if property_id_to_delete:
            # Soft delete the property by updating the 'deleted' field
            property_to_delete = get_object_or_404(Property, id=property_id_to_delete)
            property_to_delete.deleted = True
            property_to_delete.save()
            # You can add a success message here if needed

        # Get the user's profile
        try:
            user_profile = UserProfile.objects.get(user__username=username)
        except UserProfile.DoesNotExist:
            user_profile = None

        # Check if the user profile is found
        if user_profile:
            # Get non-deleted properties associated with the property owner
            properties = Property.objects.filter(property_owner=user_profile.user,
            deleted=False)

        context = {
            'user_profile': user_profile,
            'properties': properties, # Add properties to the context
        }
```

```
        return render(request, 'manageprop.html', context)
    else:
        # Handle the case where the user profile is not found
        return redirect('index')
    else:
        return redirect('index')
def submit_rental_request(request, property_id):
    if request.method == 'POST':
        # Assuming you have a form with the tenant's information
        # In a real scenario, you might want to authenticate the user or get the tenant
        information in some way

        # Get the property and tenant based on the IDs
        property = Property.objects.get(pk=property_id)
        tenant = request.user.profile # Assuming the tenant information is associated with the
        user's profile

        # Check if a rental request already exists for this property and tenant
        existing_request = RentalRequest.objects.filter(property=property,
        tenant=tenant).exists()

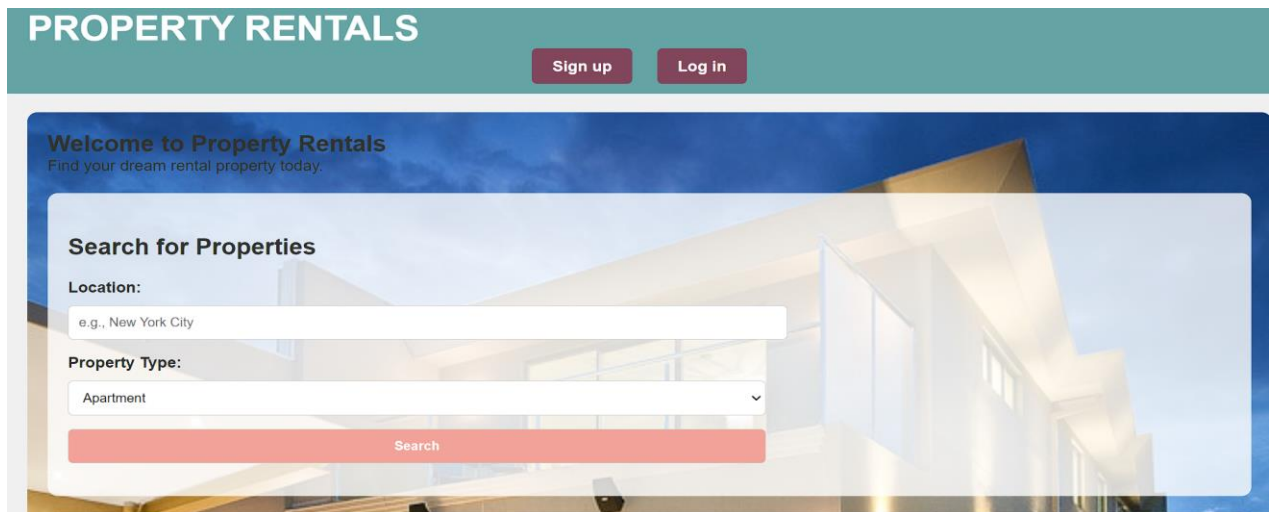
        if not existing_request:
            # Create a new rental request
            rental_request = RentalRequest(property=property, tenant=tenant)
            rental_request.save()

            messages.success(request, 'Rental request submitted successfully!')
        else:
            messages.warning(request, 'You have already submitted a rental request for this
            property.')

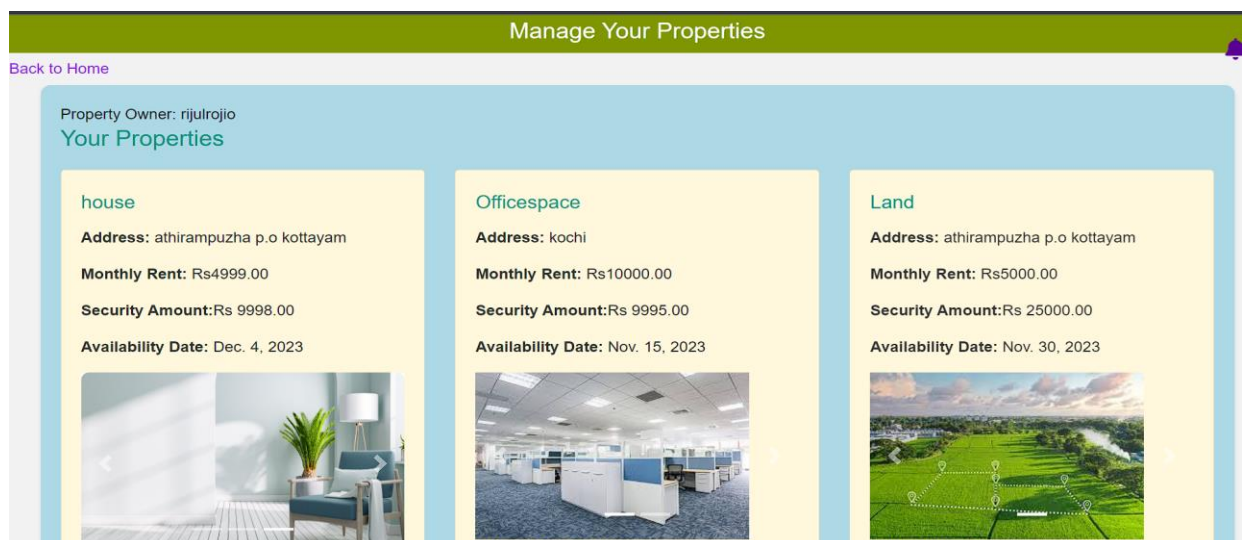
        # Redirect back to the property details page or wherever you want
        # Handle the case where the request method is not POST (optional)
        return redirect('tenantpage')
```

9.1 Screen Shots

Homepage




Manage Properties



Tenant Dashboard

[Log out](#) [Back](#)

Tenant Dashboard



Welcome, jubil


Search for Properties

Property Type:


Any

Enter the range for rent: [Search](#)


Available Properties



Property type: house
Address: athirampuzha p.o kottayam



Property type: ware
Address: pala p.o kottayam



Property type: Land
Address: athirampuzha p.o kottayam

