**PREFACE**

In today's tech-driven world, computers are the backbone of efficiency. Managing heaps of data manually can be quite a challenge, and that's where the beauty of technology shines. This project delves into the creation of a Property Management System (PMS), making property management a breeze with the prowess of Django and Python.

The goal here is simple – to make property management easy and effective for everyone involved. Whether you're a property owner or a manager, our web-based Property Management System brings everything you need to your fingertips. No more digging through paperwork or chasing down details; our system keeps you in the loop with real-time updates on property status, finances, and tenant information.

This preface is your guide to what's ahead. We'll start by talking about why this system matters and then lay out what we aim to achieve. You'll get a peek into the practical side of things – what the system can do, how it works, and what you need to get started.

So, get ready to explore a smarter way to handle property management. This isn't just about technology; it's about making your life easier. This preface is just the beginning – the journey into a more straightforward and efficient property management experience awaits .

**ACKNOWLEDGEMENT**

Embarking on the journey of developing the Property Management System as part of my fifth-semester project has been both challenging and rewarding. In the realm of B.C.A academics, this project has allowed me to put into practice the theoretical knowledge I have gained.

I extend my sincere appreciation to my teachers, whose guidance and insightful suggestions have been the bedrock of this project. Their willingness to address my countless queries with patience has been pivotal in my learning process. Their generous support and passion for teaching have been a driving force in developing this system.

I am grateful to Sherwood College of Professional Management for providing me with the platform to explore and apply my skills in the field of website development. Special thanks to my project guide Namrita Ma’am for her guidance throughout the project period, and to the entire faculty of department of BCA for their time and assistance.

A heartfelt thank you goes to my parents for their constant support, valuable advice, and encouragement in making informed decisions. Additionally, I want to express my sincere thanks to all the respondents who contributed their time and insights. Without their cooperation, the successful completion of this project would not have been possible.

This project has been a collaborative effort, and I am thankful for the collective support that has made it possible. Each contribution, big or small, has played a significant role in shaping this endeavor into a valuable learning experience.

**TABLE OF CONTENTS**

**1. INTRODUCTION**

* Background
* Objectives
* Purpose and Scope
  + Purpose
  + Scope

**2. TECHNOLOGIES AND ENVIRONMENT**

**3. REQUIREMENTS AND ANALYSIS**

* Problem Definition
* Data Flow Diagram
* Requirements Specification
  + Software
  + Hardware
* Preliminary Product Description
* Conceptual Models
* Model Used
  + Advantages
  + Disadvantages

**4. References**

**5. Code (Based on modules)**

**6. Pictures of working Project.**

**7. Testing**

**8. Future Scope and Conclusion**

PROJECT REPORT

**INTRODUCTION**

**Background:-**

In the dynamic landscape of real estate and property management, the need for efficient and streamlined solutions has become increasingly evident. The traditional methods of managing properties involve a myriad of challenges, from manual record-keeping to communication gaps between stakeholders. Recognizing these challenges, the Property Management System (PMS) project was conceived to revolutionize the way properties are managed.

The project originated from a realization that the integration of technology, specifically using Python with Django, could bring about transformative changes in the property management domain. Python, known for its readability and versatility, coupled with the robust Django web framework, provides an ideal platform to create a comprehensive and user-friendly Property Management System.

**OBJECTIVE**

The primary objective of the Property Management System is to create a centralized platform that simplifies and enhances the management of properties for both property owners and managers. The project aims to address key challenges faced in property management, including but not limited to tenant management, financial tracking, and overall property oversight. Through automation and efficient data handling, the system strives to minimize manual efforts, reduce errors, and improve the overall effectiveness of property management processes.

Furthermore, the project aims to provide a scalable and modular solution, allowing for future enhancements and adaptability to evolving industry needs. The system is designed to be user-friendly, ensuring that even users with limited technical expertise can navigate and utilize its features seamlessly.

**Purpose and Scope :-**

**Purpose:-**

The purpose of the Property Management System is to streamline the complex and multifaceted tasks associated with property management. It seeks to create a unified platform that consolidates information, facilitates communication, and provides real-time insights into property status and financial transactions. By doing so, the system contributes to increased efficiency, transparency, and accuracy in property management operations.

**Scope:-**

The scope of the project encompasses various aspects of property management, including tenant information management, financial tracking, maintenance scheduling, and reporting. The system is designed to cater to the needs of property owners, managers, and tenants, fostering a collaborative and interconnected ecosystem.

**Requirements Specification for Property Management System**

**Functional Requirements**

**Property Management:**

The system should allow the addition, modification, and deletion of property records.It should record property details, including name, location, property type (rent/sale), price, area, category, number of rooms, number of bathrooms, number of parking spaces, and images.

**Category Management:**

The system should allow the creation, modification, and deletion of property categories .It must record information about the category, including the category name and associated image.

**Agent Management:**

The system should enable the addition, modification, and deletion of agent information. It must record agent details, including name, title, and images.

**Reservation Handling:**

The system should record and manage property reservations.

It should allow users to reserve properties by providing their name, email, and additional notes.

**Financial Management:**

The system should record and manage financial transactions related to property transactions.

It should generate reports on financial transactions, sales, and outstanding balances.

**Non-Functional Requirements**

**Security:**

Access to sensitive information (e.g., financial data, agent details) should be restricted and protected by user authentication.Data transmission should be encrypted to ensure the confidentiality of information.

**Availability:**

The system should be available 24/7, with minimal downtime for maintenance and updates scheduled during off-peak hours.

**Reliability:**

The system's performance should consistently meet specified standards. It should be capable of handling concurrent users without significant degradation in performance.

**Speed:**

The system should respond to user requests within an acceptable time frame, aiming for 2-3 seconds for common operations. Retrieval of information, such as property details, should be swift and efficient.

**Usability:**

The system should have an intuitive user interface, providing a positive user experience. User input should guide the design to ensure practical and effective use of the system.

**Portability:**

The system should be accessible across various platforms and devices. It should support major web browsers and operating systems.

**Efficiency:**

The system must efficiently process input data to generate accurate and relevant outputs. It should optimize resource utilization to ensure smooth system operations.

**Scalability:**

The system should be scalable to accommodate an increasing number of properties, users, and transactions.

**ANALYSIS**

**Problem Definition:-**

In the ever-evolving landscape of real estate management, the Property Management System (PMS) emerges as a sophisticated software solution developed to provide property owners and managers with a seamless platform for efficient property administration. This system leverages the power of Django and Python to streamline and automate property-related tasks, offering a centralized hub for property management and optimizing operational workflows. The PMS is designed to enhance the overall property management experience by providing real-time insights, reducing manual errors, and empowering property managers to make informed decisions.

**Challenges in the Existing Property Management Processes:**

1. Manual Administrative Tasks: Traditional property management processes involve manual administrative tasks, leading to inefficiencies and a higher likelihood of errors.
2. Limited Automation: Lack of automation in property-related workflows results in delays, making it challenging to keep up with the dynamic nature of property management.
3. Data Discrepancies: Manual record-keeping may lead to discrepancies in property information, causing challenges in accurate reporting and decision-making.
4. Communication Gaps: Communication gaps between property owners, managers, and tenants can occur due to the absence of a centralized and integrated system.
5. Resource Intensiveness: Traditional property management processes may require significant physical resources, contributing to increased costs.

**The following objective involved in this Property management system project,**

**Objectives of the Property Management System:**

* Efficient Property Administration: Enable property owners and managers to efficiently oversee and manage multiple properties, reducing manual complexities.
* Automation of Processes: Implement automation to streamline various property-related processes, minimizing human errors and enhancing operational efficiency.
* Data Management: Create a robust database capable of handling extensive property information, including details about tenants, leases, maintenance, and financial transactions.
* User-Friendly Interface: Provide a user-friendly interface accessible to both property managers and tenants, promoting ease of use and enhancing the overall user experience.
* Integration with Django and Python: Leverage the Django framework and Python programming language to build a secure, scalable, and feature-rich Property Management System.

**Need for the New Property Management System:**

**Digital Transformation: -**

Introduce a modern Property Management System to digitally transform and streamline property management processes.

**Automation of Workflows:**

Incorporate automation into various workflows, ensuring timely and error-free execution of property-related tasks.

**Comprehensive Data Handling:**

Implement a centralized database to handle comprehensive property information systematically, reducing the risk of data discrepancies.

**Enhanced User Experience:**

Provide a user-friendly interface accessible to property managers and tenants, fostering an enhanced and intuitive user experience.

**Integration with Django and Python:**

Leverage the power of Django and Python for robust system development, ensuring security, scalability, and the ability to meet evolving property management needs.

**Real-Time Insights:**

Offer real-time insights into property performance, financial transactions, and maintenance activities, empowering property managers with actionable data.

**Cost-Efficiency:**

Reduce resource intensiveness associated with traditional property management, contributing to overall cost-efficiency.

**Feasibility Study for Property Management System**

A feasibility study is imperative in the early stages of planning for the Property Management System (PMS) to determine its viability and justifiability. The study encompasses various aspects, ensuring that the proposed PMS can be developed successfully. The primary types of feasibility studies include:

**Technical Feasibility:**

Objective:

Evaluate if the proposed technology infrastructure can support the development and operation of the Property Management System.

Considerations:

Facility to produce property listings, agent performance charts, and overall system statistics.

Response time for property search and data processing.

Communication capabilities for effective data exchange.

Emphasis:

Configuration of the system, ensuring seamless operation and communication among different components.

**Operational Feasibility:**

Objective:

Assess whether the Property Management System will function effectively when developed and installed.

Considerations:

Support from property managers and users.

Acceptance of current property management methods.

Involvement of property managers and users in the planning and development stages.Potential operational impacts and user resistance.

Emphasis:

Understanding user perceptions and addressing potential resistance, ensuring the proposed system aligns with operational needs.

**Economic Feasibility:**

Objective:

Conduct a cost/benefit analysis to determine if the benefits of the Property Management System outweigh the costs.

Considerations:

Identification of benefits such as improved property listings, efficient agent performance tracking, and streamlined operations.

Comparison of expected benefits with development and operational costs. Economic justification for investing in the Property Management System.

Emphasis:

Continuous economic analysis throughout the system life cycle, ensuring the investment is economically justified.

**Legal Feasibility:**

Objective:

Assess whether the proposed Property Management System complies with legal requirements and regulations.

Considerations:

Adherence to real estate laws and regulations.

Compliance with data protection and privacy laws.

Identification of potential legal issues related to property listings and agent interactions.

Emphasis:

Ensuring the Property Management System operates within the legal framework, minimizing the risk of legal challenges.

**TECHNOLOGIES AND ENVIRONMENT**

**Introduction to Python:**

Python, a versatile and dynamically-typed programming language, has gained immense popularity for its readability, simplicity, and versatility.

Created by Guido van Rossum in the late 1980s, Python has evolved into a powerful language with a range of features suitable for diverse applications.

**Features of Python:**

**Readability:**

Python emphasizes code readability and simplicity. Its clear and expressive syntax allows developers to write programs with fewer lines of code compared to other languages. This readability enhances collaboration and maintainability.

**Object-Oriented:**

Python is a fully object-oriented programming language, supporting concepts such as classes, objects, and inheritance. This facilitates modularity, code reuse, and the implementation of complex systems.

**Platform Independent:**

Python code is platform-independent, meaning it can run on various operating systems without modification. This portability is attributed to Python's interpretation and compilation process.

**Multi-Paradigm:**

Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. This flexibility allows developers to choose the most suitable paradigm for their specific needs.

**Dynamic Typing:**

Python is dynamically typed, allowing variables to be assigned without specifying their type. This flexibility simplifies coding but requires careful consideration during development.

**Interpreted Language:**

Python is an interpreted language, which means that code is executed line by line. This results in a quicker development cycle and easier debugging.

**High-Level Language:**

Python is a high-level language that abstracts low-level details, making it more accessible for developers. This abstraction contributes to Python's readability and ease of use.

**Rich Standard Library:**

Python includes an extensive standard library that provides modules and packages for a wide range of tasks. This wealth of resources simplifies development by offering pre-built functionalities.

**Community Support:**

Python boasts a vibrant and active community of developers. This community support is evident through extensive documentation, forums, and a vast ecosystem of third-party libraries and frameworks.

**Web Development**:

Python is widely used in web development, with frameworks like Django and Flask offering robust solutions for building web applications. Its simplicity and readability contribute to rapid development in this domain.

**Data Science and AI:**

Python has become a leading language for data science and artificial intelligence. Libraries such as NumPy, Pandas, and TensorFlow empower developers to work with large datasets and implement machine learning algorithms.

**Versatility:**

Python's versatility makes it suitable for a broad spectrum of applications, from web development and automation to scientific computing and machine learning.

**Python Software/Environment:**

Python development involves the use of environments and tools that facilitate coding, testing, and execution. The Python Software Foundation provides the Python programming language, and developers commonly use package managers like pip to manage dependencies and virtual environments to isolate project dependencies. Additionally, Integrated Development Environments (IDEs) such as PyCharm and Visual Studio Code offer powerful tools for Python development .

**Python Libraries ( A rich Source )**

**Python has a rich ecosystem of libraries and frameworks that cover a wide range of domains, from web development to machine learning. Here are some popular Python libraries:**

**NumPy:**

Numerical Python, commonly known as NumPy, is a library for numerical operations. It provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays.

**Django:**

Django is a high-level web framework that encourages rapid development and clean, pragmatic design. It follows the Model-View-Controller (MVC) architectural pattern and is widely used for building robust web applications.

**OpenCV:**

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It contains various tools for image processing, computer vision (face recognition) , and machine learning.

**Tkinter :**

The tkinter library is the standard GUI (Graphical User Interface) toolkit for Python. It provides a set of tools for creating desktop applications with graphical user interfaces.

**Python IDE**

There are several popular Integrated Development Environments (IDEs) for Python that provide powerful tools, code assistance, and features to streamline the development process. Here are some of the most widely used Python IDEs:

**Visual Studio Code (VSCode):**

VSCode is a lightweight and extensible code editor developed by Microsoft. It has a rich set of extensions that make it suitable for Python development. It supports debugging, linting, and version control.

**Jupyter Notebooks:**

Jupyter Notebooks provide an interactive and web-based environment for Python development. It's widely used for data science, research, and education. Notebooks combine code, visualizations, and narrative text.

**Types of Python Programs:**

**Script Programs:**

Python is often used for scripting purposes. Script programs are a series of commands or instructions written in a script file that is executed by the Python interpreter.

**# script\_example.py**

**print("Hello, Python!")**

**Console-based Programs:**

Similar to Java application programs, Python can be used to create console-based programs that run in a terminal or command prompt.

**# console\_program.py**

**name = input("Enter your name: ")**

**print("Hello, " + name + "!")**

**import tkinter as tk**

**root = tk.Tk()**

**label = tk.Label(root, text="Hello, Tkinter!")**

**label.pack()**

**root.mainloop()**

**GUI (Graphical User Interface) Applications:**

Python supports GUI programming using libraries such as Tkinter, PyQt, and Kivy. These applications have a graphical interface with buttons, menus, and windows.

Example using Tkinter:

**Jupyter Notebooks**

Jupyter Notebooks are interactive documents that combine code, visualizations, and narrative text. They are widely used for data analysis, research, and education.

**# In a Jupyter Notebook cell**

**print("Hello, Jupyter!")**

**Introduction of Django**

In Django, a database is a structured collection of data that is organized in a way that allows for efficient storage, retrieval, and management. Django provides a high-level Object-Relational Mapping (ORM) framework, which means you can interact with your database using Python code and objects instead of writing raw SQL queries.

Here's a brief overview of how databases work in Django:

**Django Database Configuration**

**Django uses the settings file (settings.py) to configure the database connection. You can specify the database engine, name, user, password, host, and other settings in this file.**

**Taken from database file of setting.py example .**

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

**Models:**

In Django, a model is a Python class that represents a database table. Each attribute of the model class corresponds to a field in the table.

Models define the structure of your database and how data is stored. They are essentially a blueprint for creating database tables.

**From property section : -**

class Category(models.Model):

name = models.CharField(max\_length = 30)

image = models.ImageField(upload\_to='Property/images/', null = True)

def \_\_str\_\_(self):

return self.name

class Meta:

verbose\_name = 'Category'

verbose\_name\_plural = 'Categories'

class Reserve(models.Model):

name = models.CharField(max\_length = 50)

email = models.EmailField()

notes = models.TextField()

def \_\_str\_\_(self):

return self.name

**Migrations:**

Once you define your models, you need to create migrations. Migrations are Django's way of propagating changes you make to your models (such as adding a new field) into your database schema.

Django provides commands to create, apply, and inspect migrations.

**Created by migration**

# Generated by Django database 3.2 on 2021-05-05 15:44

from django.db import migrations

class Migration(migrations.Migration):

dependencies = [

('Property', '0002\_auto\_20210505\_1721'),

]

operations = [

migrations.AlterModelOptions(

name='Property',

options={'verbose\_name': 'Property', 'verbose\_name\_plural': 'Properties'},

),

**Project Scheduling for Property Management System**

Creating an effective project schedule is like constructing a blueprint for success. In the development of our Property Management System using Python and Django, the following steps outline our project scheduling plan:

**Project Initiation (3 days):**

Define the project scope, objectives, and team roles.

Kickstart initial meetings to align everyone on the project goals.

**Draft Project Plan (4 days):**

Develop a comprehensive project plan that outlines tasks, milestones, and timelines.Review and refine the plan based on team feedback.

**Analysis Phase (8 days):**

Dive deep into requirements analysis, understanding the needs of property owners, managers, and tenants.Plan user interviews to gather valuable insights for system design.

**User Interviews (7 days):**

Plan, schedule, and conduct user interviews to ensure the system caters to real-world needs.Document feedback and insights gathered during the interviews.

**System Design (14 days):**

Design the overall architecture, database structure, and user interfaces.

Create workflow diagrams to visualize the interaction between system components.

**Coding Phase (35 days):**

Begin coding based on the design specifications.

Implement functionalities iteratively, allowing for continuous testing and feedback.

**Testing Phase (8 days):**

Conduct thorough testing to ensure the system's functionality, security, and reliability.

Address and fix any identified issues promptly.

**Integration Testing (5 days):**

Test how different modules integrate with each other.

Ensure seamless communication between various components.

**System Level Testing (6 days):**

Test the system as a whole to ensure it meets all requirements.

Simulate real-world scenarios to identify any potential issues.

**Implementation (4 days):**

Prepare for the system's deployment, including setting up servers and configuring databases.Ensure a smooth transition from development to the live environment.

**SYSTEM ARCHITECTURE AND OVERVIEW**

**Software Requirements:**

The Property Management System with Python and Django has specific software requirements to ensure smooth operation and development. The following software components are essential:

**Minimum Software Requirements:**

**Python (version 3.x):**

Python serves as the primary programming language for the development of the system. It provides the foundation for the Django framework and facilitates seamless integration with various libraries and modules.

**Django Framework:**

Django is a high-level Python web framework used for rapid development and clean, pragmatic design. It includes an Object-Relational Mapping (ORM) system for database interaction and follows the Model-View-Controller (MVC) architectural pattern.

**(Model) Database**

SQLite (library for development and testing):

A lightweight, file-based database.

Integrated Development Environment (IDE):

**Visual Studio Code :**

An IDE facilitates efficient coding, debugging, and project management.

Version Control System:

**Github:**

Version control is essential for collaborative development and managing code versions.

Recommended/Maximum Software Requirements:

**Python (latest stable version):**

Ensure the use of the latest stable version for access to the latest features, improvements, and security updates.

Django Framework (latest stable version):

Utilize the latest stable version of Django to benefit from enhancements, bug fixes, and compatibility with the latest Python releases.

Database Management System (DBMS):

**Visual Studio Code**

Utilize the latest versions of IDEs for improved functionality, performance, and compatibility.

Version Control System:

**Git**

Ensure the use of the latest Git version for enhanced security and compatibility.

**Hardware Requirements:**

The hardware specifications for deploying and running the Property Management System are outlined below:

**Minimum Hardware Requirements:**

**Processor:**

Dual-core processor (e.g., Intel Core i3 or equivalent).

RAM (Random Access Memory):

4 GB RAM for basic system functionality.

Storage:

128 GB HDD for storing application files and data.

Display:

Standard monitor resolution for basic visualization.

Recommended/Maximum Hardware Requirements**:**

**Maximum Hardware Requirements:**

**Processor:**

Multi-core processor (e.g., Intel Core i7 or equivalent) for optimal performance.

RAM (Random Access Memory):

16 GB RAM or higher to ensure smooth execution of the application.

Storage:

SSD storage (minimum 256 GB) for fast data access and improved system responsiveness.

Display:

Full HD monitor (1920 x 1080 resolution) for clear visualization of the user interface.

Input Devices:

Standard keyboard and mouse for user input.

**Modules**

**1. Property Module:**

Description:

Manages information related to properties.

Handles property details such as name, location, type, price, and images.

Functionality:

Add, edit, and delete properties.

View a list of available properties.

Filter properties based on location and type.

**2. Category Module:**

Description:

Manages property categories.

Each property belongs to a specific category.

Functionality:

Create, edit, and delete property categories.

Associate properties with relevant categories.

**3. Reserve Module:**

Description:

Handles property reservation requests.

Functionality:

Users can reserve a property by providing their name, email, and additional notes.

View and manage reservation requests.

**4. Agent Module:**

Description:

Manages information about real estate agents.

Functionality:

Add, edit, and delete agent profiles.

Display a list of agents.

**5. About Module:**

Description:

Contains information about the company.

Functionality:

Display the company's vision, mission, and image.

**6. Contact Module:**

Description:

Manages contact details and provides a contact form.

Functionality:

Display contact details.

Allow users to send messages through a contact form.

**7. Home Module:**

Description:

Manages the home page of the application.

Functionality:

Display featured properties and categories.

**8. Search and Filter Module:**

Description:

Provides search and filter functionality.

Functionality:

Enable users to search for properties based on various criteria.

Implement filters for refining property searches.

**9. Admin Module**:

Description:

Administration panel for managing system data.

Functionality:

Admin can add, edit, and delete properties, categories, agents, etc.

Monitor reservation requests.

**10. User Authentication (Auth) Module:**

Description:

Handles user authentication and authorization.

Functionality:

User registration, login, and logout.

Define user roles and permissions.

**Admin Module will cover Five Sub modules.**

a. About section for admin .

b. Agents section for admin.

c. Authentication and Authorization section for admin.

d. Contact detail section for admin.

e. Properties section for admin.

**Admin Interface:**

**Django's admin interface allows you to interact with your database records through models via web-based UI. It is automatically generated based on your models and provides a convenient way to manage your data during development.**

**a. About Section for Admin:**

Enable the admin to manage information about the real estate agency or property management system.

Allow the admin to update details such as the company's history, mission, vision, and any other relevant information.

**b. Agents Section for Admin:**

Provide functionality for the admin to manage real estate agents. Allow addition, modification, and deletion of agent records. Include fields for agent details such as name, title, contact information, and images.

**c. Authentication and Authorization Section for Admin:**

Administer user authentication and authorization processes.

Manage user accounts, roles, and permissions.

Provide the ability to add, modify, or deactivate user accounts.

**d. Contact Detail Section for Admin:**

Enable the admin to manage contact details displayed for customers.

Allow updating contact information.

Optionally, provide a feature to edit the agency's location or integrate a map.

**e. Properties Section for Admin:**

Facilitate property management for the admin.

Allow the addition, modification, and deletion of property listings.

Include fields for property details like name, location, type, price, area, category, images, etc.

Manage property categories (e.g., residential, commercial) and associate images or icons with each category.

Django's ORM abstracts away many complexities of work and work directly with databases, making it easier for developers to focus on building applications rather than dealing with low-level database details. It supports various database backends, including PostgreSQL, MySQL, SQLite, and Oracle, allowing developers to choose the one that best fits their project requirements it only start after making migrations in project .

**Customer Module will cover Five Sub modules.**

a. About section for customer.

b. Agents section for customer.

c. Contact detail section for customer

d. Properties section for admin.

**Customer Interface:**

Django's admin interface allows you to interact with your database records through a web-based UI. It is automatically generated based on your models and provides a convenient way to manage your data during development.

**a. About Section for Customer:**

Display information about the real estate agency or property management system.Include details such as the company's history, mission, vision, and any other relevant information.

**b. Agents Section for Customer:**

Showcase a list of real estate agents associated with the agency. Provide details for each agent, including their name, title, image, and contact information.

**c. Contact Detail Section for Customer:**

Display contact details for the real estate agency.

Include a form or contact information for customers to reach out for inquiries. Optionally, integrate a map for the agency's location.

**d. Properties Section for Customer:**

Present a list of available properties for customers to browse.

Include filters for property type (rent/sale), location, price range, etc. Each property should have a detailed page with images, description, and contact form for inquiries or reservations.

Django's ORM plays a crucial role in handling database operations seamlessly. The Customer Module involves managing customer-related information, such as details about the real estate agency, agents, contact details, and property listings tailored for customers.

**ENTITY RELATION DIAGRAMS**

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.

At first glance an entity relationship diagram looks very much like a flowchart. It is the specialized symbols, and the meanings of those symbols, that make it unique.

**Common Entity Relationship Diagram Symbols**

An ER diagram is a means of visualizing how the information a system produces is related. There are five main components of an ERD:

* **Entities**, which are represented by rectangles. An entity is an object or concept about which you want to store information.

ENTITY

A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.

ENTITY

* **Actions**, which are represented by diamond shapes, show how two entities share information in the database.

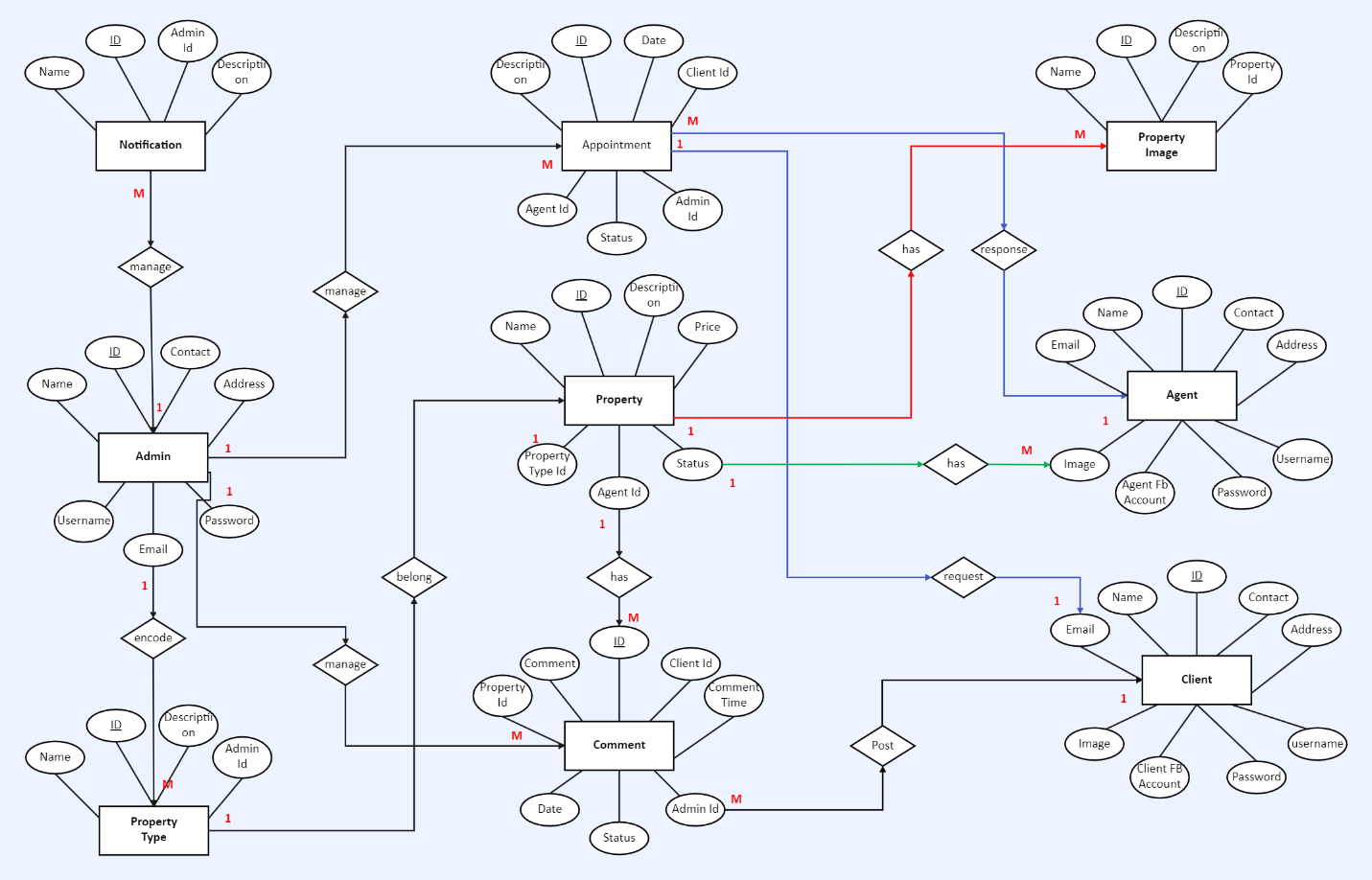
RELATION

* **Attributes**, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.

A multivalued attribute can have more than one value. For Example, an employee entity can have multiple skill values.

* **Connecting lines**, solid lines that connect attributes to show the relationships of entities in the diagram.

**ER DIAGRAM**





**DFD (DATA FLOW DIAGRAM):-**

**Design Principles of the Property Management System:**

A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various sub processes the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships.

The design of the Property Management System (PMS) has been meticulously crafted with a focus on high cohesiveness and loose coupling between modules. This design philosophy ensures minimal interaction between different modules, fostering modular independence and flexibility.

The PMS encompasses various design elements, including data flow diagrams, database tables, process logic, input and output designs, to create a robust and user-friendly property management solution.

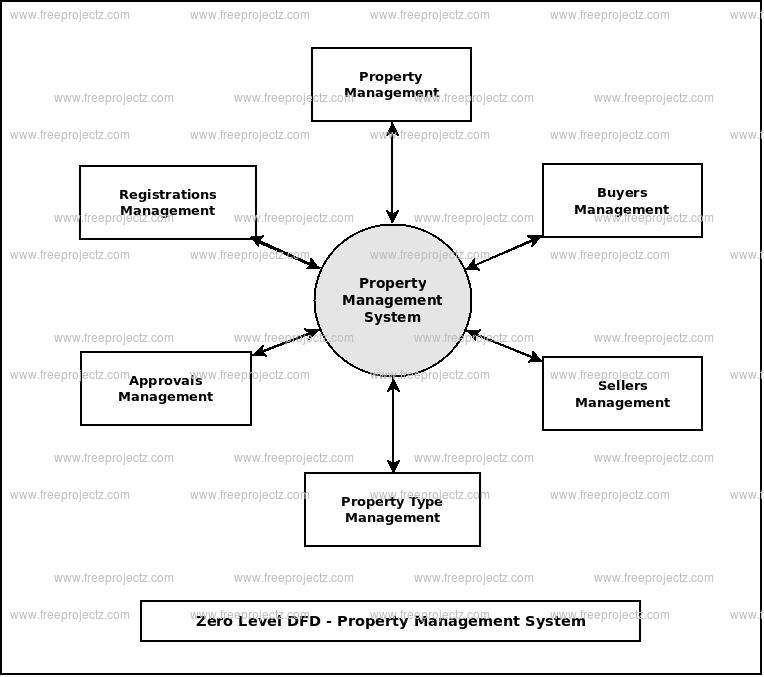
Data flow diagrams visually represent systems and processes that would be hard to describe in a chunk of text. You can use these diagrams to map out an existing system and make it better or to plan out a new system for implementation. Visualizing each element makes it easy to identify inefficiencies and produce the best possible system.

**Data flow diagram levels**

Data flow diagrams are also categorized by level. Starting with the most basic, level 0, DFDs get increasingly complex as the level increases. As you build your own data flow diagram, you will need to decide which level your diagram will be.

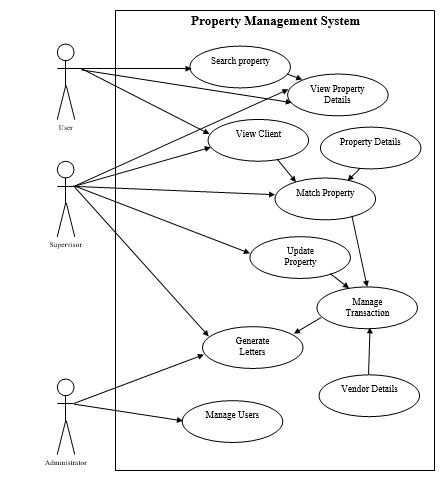
**0 - level dfd**

A 0- level DFD is the most basic form of DFD. It aims to show how the entire system works at a glance. There is only one process in the system and all the data flows either into or out of this process. Context level DFD’s demonstrates the interactions between the process and external entities. They do not contain Data Stores



**1 – level dfd**

Level 1 DFD’s aim to give an overview of the full system. They look at the system in more detail. Major processes are broken down into sub-processes. Level 1 DFD’s also identifies data stores that are used by the major processes. Level 1 is more explanatory and representative than level 0 as here mode is showed along with their sub modules in detail.



CODING

**CODE FOR HOME PAGE :-**

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Property Management System</title>

<meta name="description" content="Free Bootstrap Theme by uicookies.com">

<meta name="keywords" content="free website templates, free bootstrap themes, free template, free bootstrap, free website template">

<link href="https://fonts.googleapis.com/css?family=Open+Sans:300,400" rel="stylesheet">

<link rel="stylesheet" href="{% static 'css/styles-merged.css' %}">

<link rel="stylesheet" href="{% static 'css/style.min.css' %}">

<link rel="stylesheet" href="{% static 'css/custom.css' %}">

<!--[if lt IE 9]>

<script src="js/vendor/html5shiv.min.js"></script>

<script src="js/vendor/respond.min.js"></script>

<endif]-->

</head>

<body>

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Property Management System</title>

<meta name="description" content="Free Bootstrap Theme by uicookies.com">

<meta name="keywords" content="free website templates, free bootstrap themes, free template, free bootstrap, free website template">

<link href="https://fonts.googleapis.com/css?family=Open+Sans:300,400" rel="stylesheet">

<link rel="stylesheet" href="{% static 'css/styles-merged.css' %}">

<link rel="stylesheet" href="{% static 'css/style.min.css' %}">

<link rel="stylesheet" href="{% static 'css/custom.css' %}">

<!--[if lt IE 9]>

<script src="js/vendor/html5shiv.min.js"></script>

<script src="js/vendor/respond.min.js"></script>

<![endif]-->

</head>

<body>

<!-- START: header -->

<div class="probootstrap-loader"></div>

<header role="banner" class="probootstrap-header">

<div class="container">

<a href="/" class="probootstrap-logo">Property Management System <span>.</span></a>

<a href="#" class="probootstrap-burger-menu visible-xs" ><i>Menu</i></a>

<div class="mobile-menu-overlay"></div>

<nav role="navigation" class="probootstrap-nav hidden-xs">

<ul class="probootstrap-main-nav">

<li {% if category\_list %}class="active"{% endif %}><a href="/">Home</a></li>

<li {% if property\_list %}class="active"{% endif %}><a href="{% url 'property:property\_list' %}">Properties</a></li>

<li {% if agents\_list %}class="active"{% endif %}><a href="{% url 'agents:agents\_list' %}">Agents</a></li>

<li {% if about %}class="active"{% endif %}><a href="{% url 'about\_us:about\_us' %}">About</a></li>

<li {% if contact\_details %}class="active"{% endif %}><a href="{% url 'contact:send\_mail' %}">Contact</a></li>

</ul>

<div class="extra-text visible-xs">

<a href="#" class="probootstrap-burger-menu"><i>Menu</i></a>

<h5>Address</h5>

<p>lucknow,India</p>

<h5>Connect</h5>

<ul class="social-buttons">

<li><a href="#"><i class="icon-twitter"></i></a></li>

<li><a href="#"><i class="icon-facebook2"></i></a></li>

<li><a href="#"><i class="icon-instagram2"></i></a></li>

</ul>

</div>

</nav>

</div>

</header>

<!-- END: header -->

<section class="probootstrap-slider flexslider">

<div class="probootstrap-wrap-banner">

<div class="container">

<div class="row">

<div class="col-md-8 col-md-offset-2">

<div class="probootstrap-home-search probootstrap-animate">

<form action="{% url 'property:property\_list' %}" method="get">

<h2 class="heading">Search your dream Property here</h2>

<div class="probootstrap-field-group">

<div class="probootstrap-fields">

<div class="search-field">

<i class="icon-location2"></i>

<input type="text" name="search" class="form-control" placeholder="Enter address">

</div>

<div class="search-category">

<i class="icon-chevron-down"></i>

<select name="property\_type" id="" class="form-control">

<option value="Rent">For Rent</option>

<option value="Sale">For Sale</option>

</select>

</div>

</div>

<button class="btn btn-success" type="submit"><i class="icon-magnifying-glass t2"></i> Start Search</button>

</div>

</form>

</div>

</div>

</div>

</div>

</div>

<li style="background-image: url(img/slider\_1.jpg);" class="overlay"></li>

</section>

<!-- END: slider  -->

<section class="probootstrap-section probootstrap-section-lighter">

<div class="container">

<div class="row">

<div class="col-md-4">

<div class="probootstrap-card text-center probootstrap-animate">

<div class="probootstrap-card-media svg-sm colored">

<img src="{% static 'img/flaticon/svg/001-prize.svg' %}" class="svg">

</div>

<div class="probootstrap-card-text">

<h2 class="probootstrap-card-heading">Award Winning Brooker</h2>

<p>Far away, behind mountains.</p>

</div>

</div>

</div>

<div class="col-md-4">

<div class="probootstrap-card text-center probootstrap-animate">

<div class="probootstrap-card-media svg-sm colored">

<img src="{% static 'img/flaticon/svg/005-new.svg' %}" class="svg">

</div>

<div class="probootstrap-card-text">

<h2 class="probootstrap-card-heading">New Houses</h2>

<p>Far away, behind the mountains</p>

</div>

</div>

</div>

<div class="col-md-4">

<div class="probootstrap-card text-center  probootstrap-animate">

<div class="probootstrap-card-media svg-sm colored">

<img src="{% static 'img/flaticon/svg/006-coin.svg' %}" class="svg">

</div>

<div class="probootstrap-card-text">

<h2 class="probootstrap-card-heading">Affordable Houses</h2>

<p>Far away, behind the mountains</p>

</div>

</div>

</div>

</div>

</div>

</section>

<!-- END: section -->

<section class="probootstrap-section">

<div class="container">

<div class="row heading">

<h2 class="mt0 mb50 text-center">Explore Our Neighborhoods</h2>

</div>

<div class="row probootstrap-gutter10">

{% for category in category\_list %}

<div class="col-md-6 col-sm-6">

<a href="#" class="probootstrap-hover-overlay">

<img src="{{category.image}}" alt="{{category.name.url}}" class="img-responsive">

<div class="probootstrap-text-overlay">

<h3>{{category.name}}</h3>

<p>{{category.count}} Properties</p>

</div>

</a>

</div>

{% endfor %}

</div>

</div>

</section>

<!-- END: section -->

<section class="probootstrap-section">

<div class="container">

<div class="row heading">

<h2 class="mt0 mb50 text-center">Our Services</h2>

</div>

<div class="row">

<div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 probootstrap-animate">

<div class="service text-center">

<div class="icon"><i class="icon-list2"></i></div>

<h2 class="heading">Property Listing</h2>

<p>Far away, behind the mountains.</p>

</div>

</div>

<div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 probootstrap-animate">

<div class="service text-center">

<div class="icon"><i class="icon-power-cord"></i></div>

<h2 class="heading">Property Management</h2>

<p>Far away, behind the mountains.</p>

</div>

</div>

<div class="clearfix visible-sm-block"></div>

<div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 probootstrap-animate">

<div class="service text-center">

<div class="icon"><i class="icon-price-tag2"></i></div>

<h2 class="heading">Renting Properties</h2>

<p>Far away, behind the  mountains.</p>

</div>

</div>

<div class="clearfix visible-lg-block visible-md-block"></div>

<div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 probootstrap-animate">

<div class="service text-center">

<div class="icon"><i class="icon-direction"></i></div>

<h2 class="heading">Selling Properties</h2>

<p>Far away, behind the  mountains.</p>

</div>

</div>

<div class="clearfix visible-sm-block"></div>

<div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 probootstrap-animate">

<div class="service text-center">

<div class="icon"><i class="icon-home3"></i></div>

<h2 class="heading">Book A Property</h2>

<p>Far away, behind the mountains.</p>

</div>

</div>

<div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 probootstrap-animate">

<div class="service text-center">

<div class="icon"><i class="icon-magnifying-glass"></i></div>

<h2 class="heading">Search Property</h2>

<p>Far away, behind the  mountains.</p>

</div>

</div>

<div class="clearfix visible-lg-block visible-md-block"></div>

</div>

</div>

</section>

<!-- END: section -->

<footer class="probootstrap-footer probootstrap-bg" style="background-image: url(img/slider\_3.jpg)">

<div class="container">

<div class="row mb60">

<div class="col-md-3">

<div class="probootstrap-footer-widget">

<h4 class="heading">About Properties</h4>

<p>Far away, behind the mountains. </p>

</div>

</div>

<div class="col-md-3">

<div class="probootstrap-footer-widget probootstrap-link-wrap">

<h4 class="heading">Quick Links</h4>

<ul class="stack-link">

<li><a href="#">Property Listing</a></li>

<li><a href="#">Rent Properties</a></li>

<li><a href="#">Sell Properties</a></li>

<li><a href="#">Agents</a></li>

<li><a href="#">Testimonial</a></li>

</ul>

</div>

</div>

<div class="col-md-3">

<div class="probootstrap-footer-widget">

<h4 class="heading">Popular Cities</h4>

<ul class="stack-link">

<li><a href="#">lucknow</a></li>

<li><a href="#">punjab</a></li>

<li><a href="#">delhi</a></li>

<li><a href="#">kolkata</a></li>

</ul>

</div>

</div>

<div class="col-md-3">

<div class="probootstrap-footer-widget probootstrap-link-wrap">

<h4 class="heading">Subscribe</h4>

<p>Far away, behind the mountains.</p>

<form action="#">

<div class="form-field">

<input type="text" class="form-control" placeholder="Enter your email">

<button class="btn btn-subscribe">Send</button>

</div>

</form>

</div>

</div>

</div>

<div class="row copyright">

<div class="col-md-6">

<div class="probootstrap-footer-widget right">

<ul class="probootstrap-footer-social">

<li><a href="#"><i class="icon-twitter"></i></a></li>

<li><a href="#"><i class="icon-facebook"></i></a></li>

<li><a href="#"><i class="icon-instagram2"></i></a></li>

</ul>

</div>

</div>

</div>

</div>

</footer>

<div class="gototop js-top">

<a href="#" class="js-gotop"><i class="icon-chevron-thin-up"></i></a>

</div>

<script src="{% static 'js/scripts.min.js' %}"></script>

<script src="{% static 'js/main.min.js' %}"></script>

<script src="{% static 'js/custom.js' %}"></script>

</body>

</html>

<!-- START: header -->

<div class="probootstrap-loader"></div>

<header role="banner" class="probootstrap-header">

<div class="container">

<a href="/" class="probootstrap-logo">Property Management System <span>.</span></a>

<a href="#" class="probootstrap-burger-menu visible-xs" ><i>Menu</i></a>

<div class="mobile-menu-overlay"></div>

<nav role="navigation" class="probootstrap-nav hidden-xs">

<ul class="probootstrap-main-nav">

<li {% if category\_list %}class="active"{% endif %}><a href="/">Home</a></li>

<li {% if property\_list %}class="active"{% endif %}><a href="{% url 'property:property\_list' %}">Properties</a></li>

<li {% if agents\_list %}class="active"{% endif %}><a href="{% url 'agents:agents\_list' %}">Agents</a></li>

<li {% if about %}class="active"{% endif %}><a href="{% url 'about\_us:about\_us' %}">About</a></li>

<li {% if contact\_details %}class="active"{% endif %}><a href="{% url 'contact:send\_mail' %}">Contact</a></li>

</ul>

<div class="extra-text visible-xs">

<a href="#" class="probootstrap-burger-menu"><i>Menu</i></a>

<h5>Address</h5>

<p>lucknow,India</p>

<h5>Connect</h5>

<ul class="social-buttons">

<li><a href="#"><i class="icon-twitter"></i></a></li>

<li><a href="#"><i class="icon-facebook2"></i></a></li>

<li><a href="#"><i class="icon-instagram2"></i></a></li>

</ul>

</div>

</nav>

</div>

</header>

<!-- END: header -->

<section class="probootstrap-slider flexslider">

<div class="probootstrap-wrap-banner">

<div class="container">

<div class="row">

<div class="col-md-8 col-md-offset-2">

<div class="probootstrap-home-search probootstrap-animate">

<form action="{% url 'property:property\_list' %}" method="get">

<h2 class="heading">Search your dream Property here</h2>

<div class="probootstrap-field-group">

<div class="probootstrap-fields">

<div class="search-field">

<i class="icon-location2"></i>

<input type="text" name="search" class="form-control" placeholder="Enter address">

</div>

<div class="search-category">

<i class="icon-chevron-down"></i>

<select name="property\_type" id="" class="form-control">

<option value="Rent">For Rent</option>

<option value="Sale">For Sale</option>

</select>

</div>

</div>

<button class="btn btn-success" type="submit"><i class="icon-magnifying-glass t2"></i> Start Search</button>

</div>

</form>

</div>

</div>

</div>

</div>

</div>

<li style="background-image: url(img/slider\_1.jpg);" class="overlay"></li>

</section>

<!-- END: slider  -->

<section class="probootstrap-section probootstrap-section-lighter">

<div class="container">

<div class="row">

<div class="col-md-4">

<div class="probootstrap-card text-center probootstrap-animate">

<div class="probootstrap-card-media svg-sm colored">

<img src="{% static 'img/flaticon/svg/001-prize.svg' %}" class="svg">

</div>

<div class="probootstrap-card-text">

<h2 class="probootstrap-card-heading">Award Winning Brooker</h2>

<p>Far away, behind mountains.</p>

</div>

</div>

</div>

<div class="col-md-4">

<div class="probootstrap-card text-center probootstrap-animate">

<div class="probootstrap-card-media svg-sm colored">

<img src="{% static 'img/flaticon/svg/005-new.svg' %}" class="svg">

</div>

<div class="probootstrap-card-text">

<h2 class="probootstrap-card-heading">New Houses</h2>

<p>Far away, behind the mountains</p>

</div>

</div>

</div>

<div class="col-md-4">

<div class="probootstrap-card text-center  probootstrap-animate">

<div class="probootstrap-card-media svg-sm colored">

<img src="{% static 'img/flaticon/svg/006-coin.svg' %}" class="svg">

</div>

<div class="probootstrap-card-text">

<h2 class="probootstrap-card-heading">Affordable Houses</h2>

<p>Far away, behind the mountains</p>

</div>

</div>

</div>

</div>

</div>

</section>

<!-- END: section -->

<section class="probootstrap-section">

<div class="container">

<div class="row heading">

<h2 class="mt0 mb50 text-center">Explore Our Neighborhoods</h2>

</div>

<div class="row probootstrap-gutter10">

{% for category in category\_list %}

<div class="col-md-6 col-sm-6">

<a href="#" class="probootstrap-hover-overlay">

<img src="{{category.image}}" alt="{{category.name.url}}" class="img-responsive">

<div class="probootstrap-text-overlay">

<h3>{{category.name}}</h3>

<p>{{category.count}} Properties</p>

</div>

</a>

</div>

{% endfor %}

</div>

</div>

</section>

<!-- END: section -->

<section class="probootstrap-section">

<div class="container">

<div class="row heading">

<h2 class="mt0 mb50 text-center">Our Services</h2>

</div>

<div class="row">

<div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 probootstrap-animate">

<div class="service text-center">

<div class="icon"><i class="icon-list2"></i></div>

<h2 class="heading">Property Listing</h2>

<p>Far away, behind the mountains.</p>

</div>

</div>

<div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 probootstrap-animate">

<div class="service text-center">

<div class="icon"><i class="icon-power-cord"></i></div>

<h2 class="heading">Property Management</h2>

<p>Far away, behind the mountains.</p>

</div>

</div>

<div class="clearfix visible-sm-block"></div>

<div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 probootstrap-animate">

<div class="service text-center">

<div class="icon"><i class="icon-price-tag2"></i></div>

<h2 class="heading">Renting Properties</h2>

<p>Far away, behind the  mountains.</p>

</div>

</div>

<div class="clearfix visible-lg-block visible-md-block"></div>

<div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 probootstrap-animate">

<div class="service text-center">

<div class="icon"><i class="icon-direction"></i></div>

<h2 class="heading">Selling Properties</h2>

<p>Far away, behind the  mountains.</p>

</div>

</div>

<div class="clearfix visible-sm-block"></div>

<div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 probootstrap-animate">

<div class="service text-center">

<div class="icon"><i class="icon-home3"></i></div>

<h2 class="heading">Book A Property</h2>

<p>Far away, behind the mountains.</p>

</div>

</div>

<div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 probootstrap-animate">

<div class="service text-center">

<div class="icon"><i class="icon-magnifying-glass"></i></div>

<h2 class="heading">Search Property</h2>

<p>Far away, behind the  mountains.</p>

</div>

</div>

<div class="clearfix visible-lg-block visible-md-block"></div>

</div>

</div>

</section>

<!-- END: section -->

<footer class="probootstrap-footer probootstrap-bg" style="background-image: url(img/slider\_3.jpg)">

<div class="container">

<div class="row mb60">

<div class="col-md-3">

<div class="probootstrap-footer-widget">

<h4 class="heading">About Properties</h4>

<p>Far away, behind the mountains. </p>

</div>

</div>

<div class="col-md-3">

<div class="probootstrap-footer-widget probootstrap-link-wrap">

<h4 class="heading">Quick Links</h4>

<ul class="stack-link">

<li><a href="#">Property Listing</a></li>

<li><a href="#">Rent Properties</a></li>

<li><a href="#">Sell Properties</a></li>

<li><a href="#">Agents</a></li>

<li><a href="#">Testimonial</a></li>

</ul>

</div>

</div>

<div class="col-md-3">

<div class="probootstrap-footer-widget">

<h4 class="heading">Popular Cities</h4>

<ul class="stack-link">

<li><a href="#">lucknow</a></li>

<li><a href="#">punjab</a></li>

<li><a href="#">delhi</a></li>

<li><a href="#">kolkata</a></li>

</ul>

</div>

</div>

<div class="col-md-3">

<div class="probootstrap-footer-widget probootstrap-link-wrap">

<h4 class="heading">Subscribe</h4>

<p>Far away, behind the mountains.</p>

<form action="#">

<div class="form-field">

<input type="text" class="form-control" placeholder="Enter your email">

<button class="btn btn-subscribe">Send</button>

</div>

</form>

</div>

</div>

</div>

<div class="row copyright">

<div class="col-md-6">

<div class="probootstrap-footer-widget right">

<ul class="probootstrap-footer-social">

<li><a href="#"><i class="icon-twitter"></i></a></li>

<li><a href="#"><i class="icon-facebook"></i></a></li>

<li><a href="#"><i class="icon-instagram2"></i></a></li>

</ul>

</div>

</div>

</div>

</div>

</footer>

<div class="gototop js-top">

<a href="#" class="js-gotop"><i class="icon-chevron-thin-up"></i></a>

</div>

<script src="{% static 'js/scripts.min.js' %}"></script>

<script src="{% static 'js/main.min.js' %}"></script>

<script src="{% static 'js/custom.js' %}"></script>

</body>

</html>

**CODE FOR PROPERTY DETAILS PAGE :-**

{% extends 'base.html' %}

{% load bootstrap4 %}

{% block body %}

<section class="probootstrap-slider flexslider2 page-inner">

<div class="overlay"></div>

<div class="probootstrap-wrap-banner">

<div class="container">

<div class="row">

<div class="col-md-8">

<div class="page-title probootstrap-animate">

<div class="probootstrap-breadcrumbs">

<a href="#">Home</a><span><a href="{% url 'property:property\_list' %}">Properties</a>

</div>

<h1>{{property\_detail}}</h1>

</div>

</div>

</div>

</div>

</div>

<ul class="slides">

<li style="background-image: url(img/slider\_1.jpg);"></li>

<li style="background-image: url(img/slider\_4.jpg);"></li>

<li style="background-image: url(img/slider\_2.jpg);"></li>

</ul>

</section>

<!-- END: slider  -->

<section class="probootstrap-section">

<div class="container">

<div class="row">

<div class="col-md-12">

<p><img src="{{property\_detail.image.url}}" class="img-responsive" alt="{{property\_detail.name.url}}"></p>

</div>

</div>

<div class="row">

<div class="col-md-12">

<h2>Details</h2><br>

<p>Type : {{property\_detail.property\_type}}</p>

<p>Area : {{property\_detail.area}} sqft</p>

<p>Rooms: {{property\_detail.number\_of\_rooms}}</p>

<p>Bathrooms : {{property\_detail.number\_of\_bathrooms}}</p>

<p>Garages : {{property\_detail.number\_of\_parking}}</p>

</div>

</div>

</div>

</section>

<section class="probootstrap-section">

<div class="container">

<div class="row">

<div class="col-md-12">

<h2>Reserve This Property</h2>

<form action='' method="POST">

{% csrf\_token %}

{% bootstrap\_form reserve\_form %}

{% buttons %}

<button type="submit" class="btn btn-primary">Reserve</button>

{% endbuttons %}

</form>

</div>

</div>

</div>

</section>

{% endblock body %}

**CODE FOR PROPERTY LISTING PAGE :-**

{% extends 'base.html' %}

{% load bootstrap4 %}

{% block body %}

<section class="probootstrap-slider flexslider2 page-inner">

<div class="overlay"></div>

<div class="probootstrap-wrap-banner">

<div class="container">

<div class="row">

<div class="col-md-8">

<div class="page-title probootstrap-animate">

<div class="probootstrap-breadcrumbs">

<a href="#">Home</a><span>Properties</span>

</div>

<h1>Properties</h1>

</div>

</div>

</div>

</div>

</div>

<ul class="slides">

<li style="background-image: url(img/slider\_1.jpg);"></li>

<li style="background-image: url(img/slider\_4.jpg);"></li>

<li style="background-image: url(img/slider\_2.jpg);"></li>

</ul>

</section>

<!-- END: slider  -->

<section class="probootstrap-section probootstrap-section-lighter">

<div class="container">

<div class="row">

{% if property\_list %}

{% for property in property\_list %}

<div class="col-md-4 col-sm-6">

<div class="probootstrap-card probootstrap-listing">

<div class="probootstrap-card-media">

<img src="{{property.image.url}}" class="img-responsive" alt="{{property.name.url}}">

</div>

<div class="probootstrap-card-text">

<h2 class="probootstrap-card-heading"><a href="{% url 'property:property\_detail' property.id %}">{{property}}</a></h2>

<div class="probootstrap-listing-location">

<i class="icon-location2"></i> <span>{{property.location}}</span>

</div>

<div class="probootstrap-listing-category for-sale"><span>FOR {{property.property\_type}}</span></div>

<div class="probootstrap-listing-price"><strong>Rs. {{property.price}}K</strong></div>

</div>

<div class="probootstrap-card-extra">

<ul>

<li>

Area

<span>{{property.area}}</span>

</li>

<li>

Beds

<span>{{property.number\_of\_rooms}}</span>

</li>

<li>

Baths

<span>{{property.number\_of\_bathrooms}}</span>

</li>

<li>

Garages

<span>{{property.number\_of\_parking}}</span>

</li>

</ul>

</div>

</div>

<!-- END listing -->

</div>

{% endfor %}

{% else %}

<div class="col-md-4 col-sm-6">

<h2>No Such Properties Found</h2>

</div>

{% endif %}

</div>

</div>

</section>

{% endblock body %}

**PRELIMINRY PRODUCT DESCRPTION**

The Preliminary Product Description outlines the key features and functionalities of the Property Management System (PMS) developed using Django and Python. This comprehensive description provides a clear understanding of the system's capabilities and how it addresses the needs of property management.

**System Overview:**

The Property Management System with Django is a robust and scalable web-based application designed to streamline property management operations. Leveraging the power of Django's framework and Python programming language, this system offers a comprehensive solution for managing various aspects of property portfolios, tenants, and associated transactions.

**Key Features:**

**User-Friendly Dashboard:**

The system boasts an intuitive and user-friendly dashboard that provides a quick overview of essential property management metrics. Users can access important information at a glance, facilitating efficient decision-making.

Property Listing and Details:

Property managers can effortlessly add, edit, and remove property listings. Each property's details, including location, specifications, and amenities, are meticulously organized for easy reference.

**Tenant Management:**

The PMS enables efficient tenant management, allowing property managers to maintain a detailed database of tenants. This includes tenant contact information, lease details, and communication history.

Lease and Agreement Tracking:

Tracking lease agreements and related documentation is simplified through the system. Property managers can set reminders for lease renewals, monitor rental payments, and manage legal documentation seamlessly.

**Financial Management:**

The system includes robust financial management features, allowing property managers to track income and expenses associated with each property. Automated financial reports provide insights into the financial health of the property portfolio.

**Maintenance Requests and Tracking:**

Tenants can submit maintenance requests through the system, streamlining the communication between tenants and property managers. Maintenance tasks are tracked, and updates are communicated in real-time.

**Communication and Notifications:**

Integrated communication features enable property managers to communicate with tenants, property owners, and other stakeholders. Automated notifications for important events, such as lease expirations or maintenance schedules, enhance communication efficiency.

**Security and Access Control:**

The PMS prioritizes security with role-based access control. Different user roles, such as property manager, administrator, and tenant, have specific permissions, ensuring data privacy and security.

**Scalability and Customization:**

Designed with scalability in mind, the system can adapt to the growing needs of property portfolios. Additionally, customization options allow users to tailor the system to their specific property management workflows.

TESTING

**Testing Overview for Property Management System with Python and Django:**

**Why Testing is Essential:**

**Error Detection:**

Testing is essential for identifying and rectifying errors within the system before deployment, contributing to a robust and error-free application.

**Integrity Enhancement:**

By ensuring that the system aligns with user requirements and functions correctly, testing enhances the overall integrity of the Property Management System.

**Prevention of Errors:**

Testing aids in the identification of error-prone areas, contributing to the prevention of errors and improving the overall reliability of the system.

**Value Addition:**

Testing adds significant value to the product by confirming its adherence to user requirements and ensuring a seamless user experience.

**Causes of Errors:**

**Communication Gap:**

Differences in thought processes, backgrounds, and experiences can lead to a communication gap. In the context of the PMS, clear communication between developers and stakeholders is essential.

**Time Constraints:**

Time constraints may lead to rushed development, compromising the quality of the system. Balancing features and schedules requires careful consideration.

**Over Commitment:**

Developer overcommitment due to enthusiasm can result in missed deadlines and compromised quality, emphasizing the need for realistic project planning.

**Insufficient Testing:**

Inadequate testing is a significant source of errors. Comprehensive testing throughout all development phases is crucial to the success of the PMS.

**Inadequate Requirements Gathering:**

Rushed development without a thorough understanding of business and technical requirements can lead to errors in the system.

**Fast-Changing Technology:**

Adapting to rapidly evolving technologies may result in inadequate expertise and improper implementation, emphasizing the importance of staying informed.

**Testing Principles:**

**Error Discovery:**

Testing is focused on discovering yet-to-be-identified errors within the system, promoting continuous improvement.

**Traceability to Requirements:**

Tests are designed to be traceable to customer requirements, ensuring alignment with specified criteria.

**Early Planning:**

Comprehensive test planning is conducted well in advance of actual testing, contributing to systematic and efficient testing processes.

**Incremental Testing:**

Testing progresses incrementally, moving from "in the small" to "testing in the large," ensuring a thorough evaluation at various levels.

**Exhaustive Testing Limitation:**

Acknowledging the limitation of exhaustive testing, focus is placed on high-priority areas and critical functionalities.

**Independent Third-Party Training:**

Training is conducted by an independent third party to ensure unbiased and comprehensive evaluation.

**Testing Objectives:**

**Error Detection:**

The primary objective of testing is to detect and rectify errors, ensuring a high-quality and error-free Property Management System.

**Test Case Quality:**

Well-designed test cases contribute to the overall quality of testing by addressing a range of scenarios and potential errors.

**Success Criteria:**

A successful test is one that uncovers yet-to-be-discovered errors, allowing for corrective actions to be taken.

**Testing Technique Used:**

The Testing technique employed for the Property Management System involves continuous and systematic testing at various levels. Black box testing is the primary approach, focusing on functional requirements and ensuring that test cases cover a variety of scenarios. The testing process begins at the module level and extends to a comprehensive evaluation of the entire system.

Continuous testing ensures the full functionality and reliability of the Property Management System. Meticulously designed test cases identify potential errors and contribute to the removal of any discrepancies within the project. The testing approach is systematic .

**Kinds of Testing:**

**System Testing:**

The Property Management System (PMS) has undergone rigorous system testing using the black box testing method, ensuring that the software functions seamlessly with Python and Django. This testing approach is instrumental in validating the operational functionality, input acceptance, output accuracy, and maintaining the integrity of the database.

**Black Box Testing:**

Emphasis is placed on black box testing, focusing on functionality and requirements without internal code knowledge. This aligns with the specifications of the PMS**.**

**White Box Testing:**

While black box testing is the primary focus, white box testing principles may also be applied selectively to ensure comprehensive coverage.

**Unit Testing:**

Unit testing is conducted at the micro scale, verifying individual functions and code modules. This ensures the integrity of specific components.

**Test Cases Designed**

**TEST CASES FOR CLIENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **TEST CASE ID** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** |
| 1 | Password and key combination | Start | Operation started |
| 2 | No Input | Alert Message | Select a valid configuration from setting for password and admin |

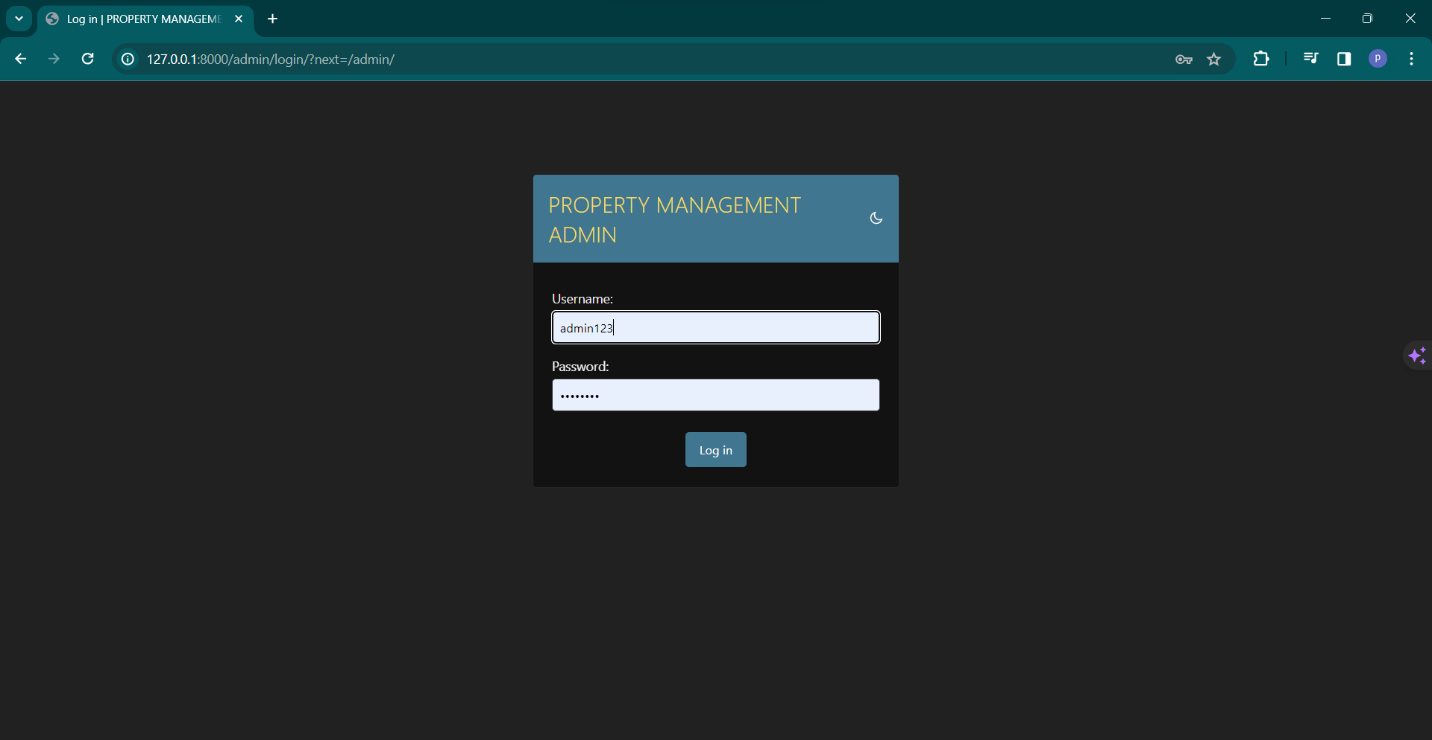
**TEST CASES FOR SERVER**

|  |  |  |  |
| --- | --- | --- | --- |
| **TEST CASE ID** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** |
| 1 | http://127.0.0.1:8000/ | Connection establish | Client Connected |
| 2 | Get Status | Alert Box [module not found ] | If no client selected “alert message” is displayed |
| 3 | Choose Path | Files [ module ] installing by pip | Library installed |

PROJECT --- SCREENSHOT

**ADMIN PANEL**

**Admin login page :-**

****

**Admin Fields :-**

![A screenshot of a computer

Description automatically generated

**About Section:-**

**A screenshot of a computer

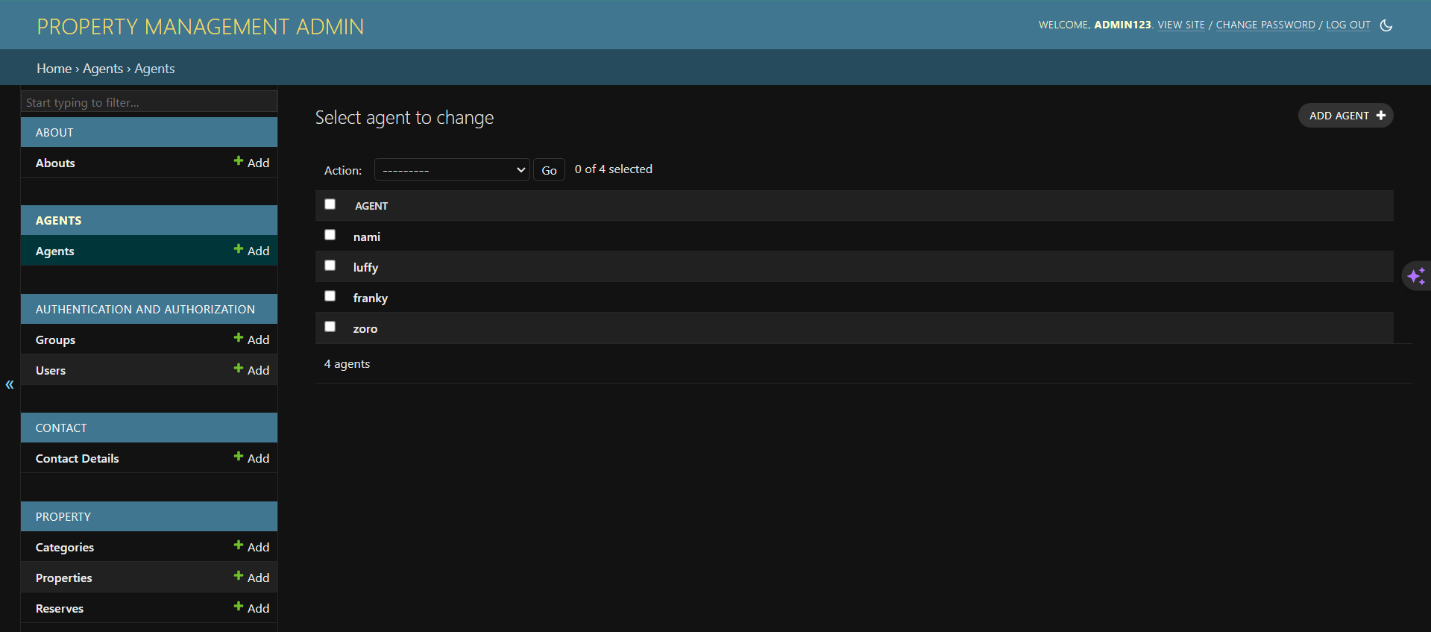
Description automatically generated**

**Inside about section table :-**

**A screenshot of a computer

Description automatically generated**

**Agent selection:-**

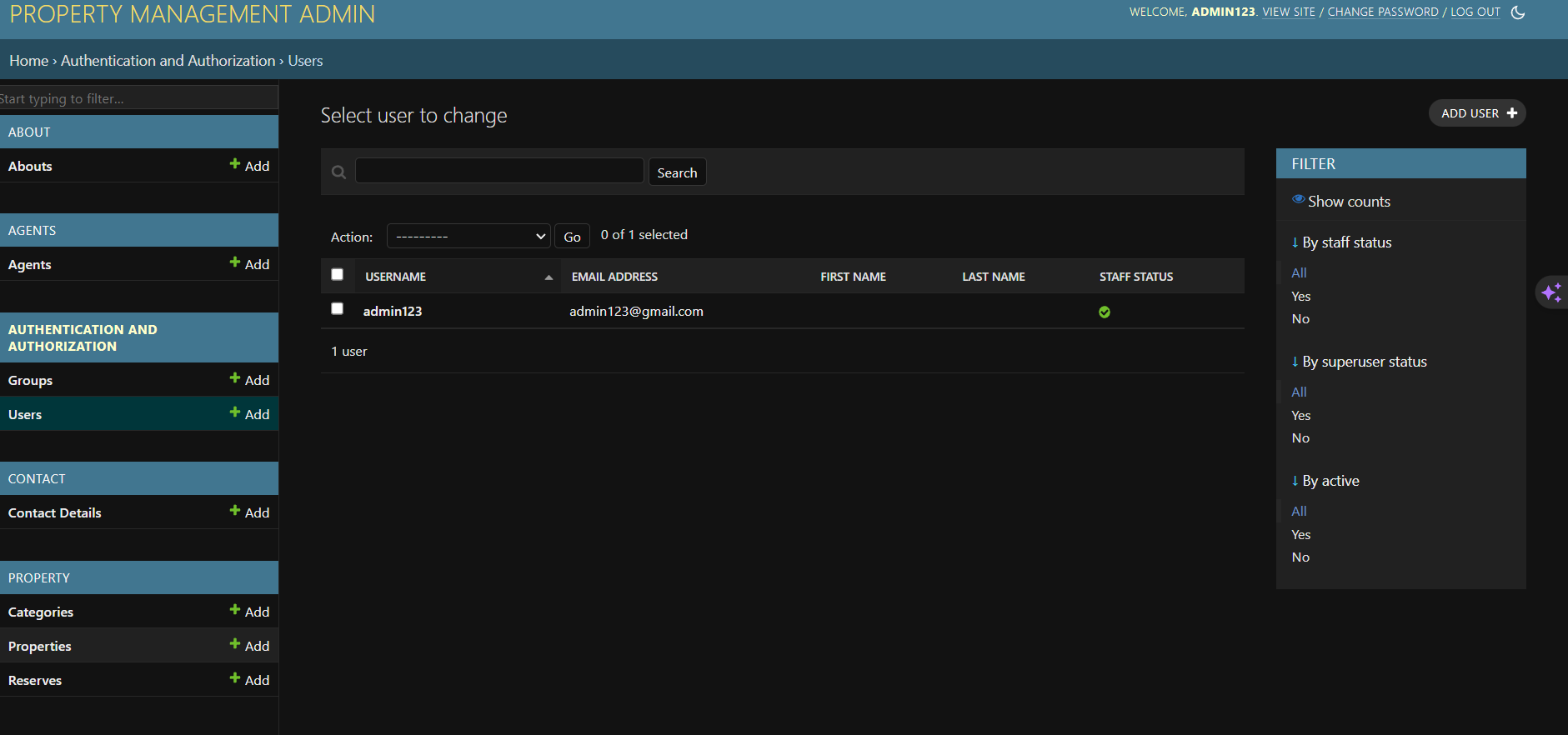


**Admin authorization and authentication :-**

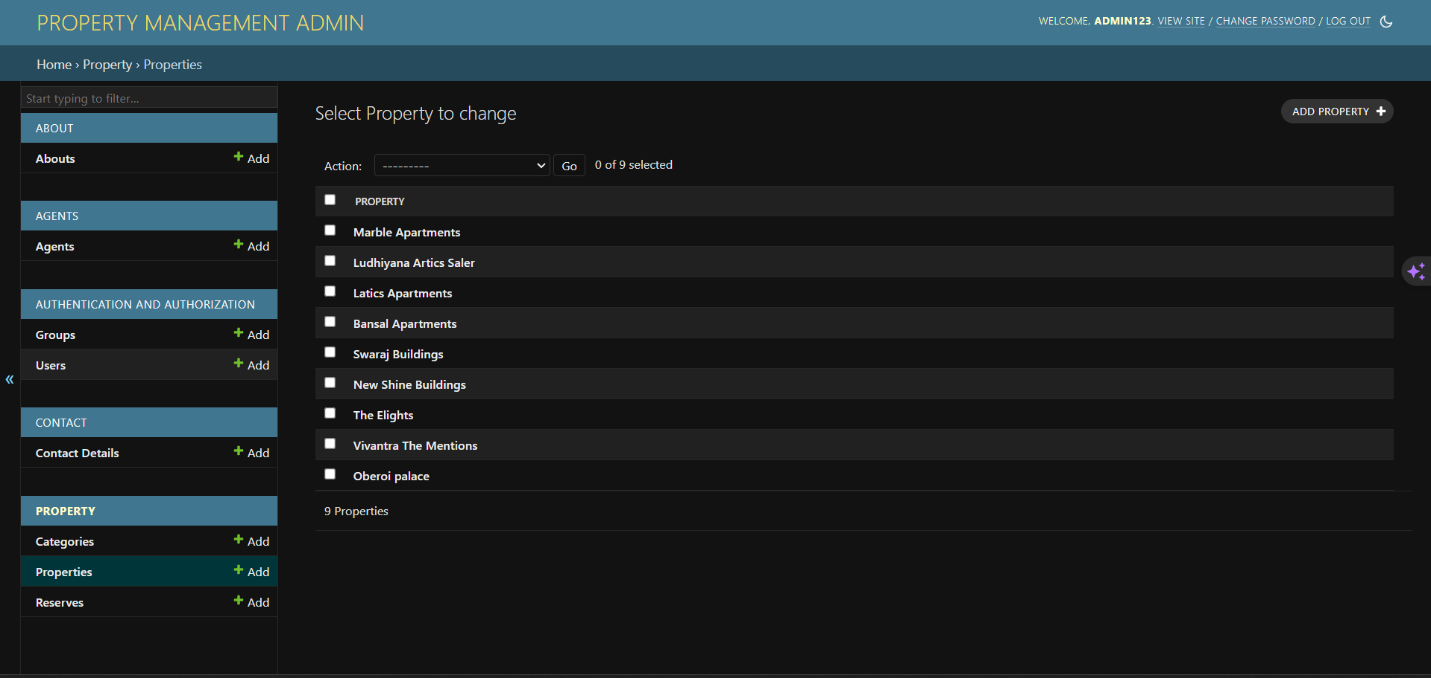
A screenshot of a computer

Description automatically generated

**Picture for user account :**

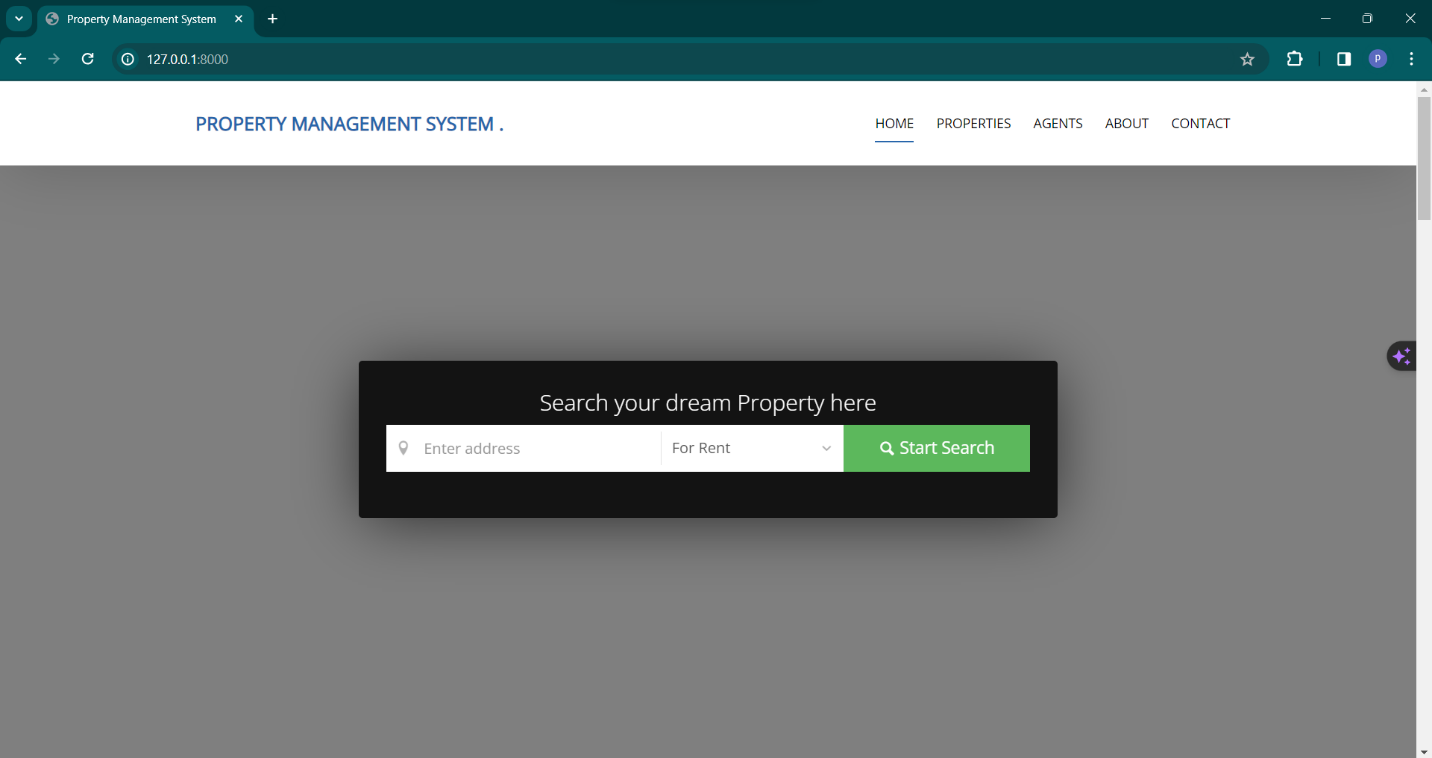


**Admin section for controlling properties**

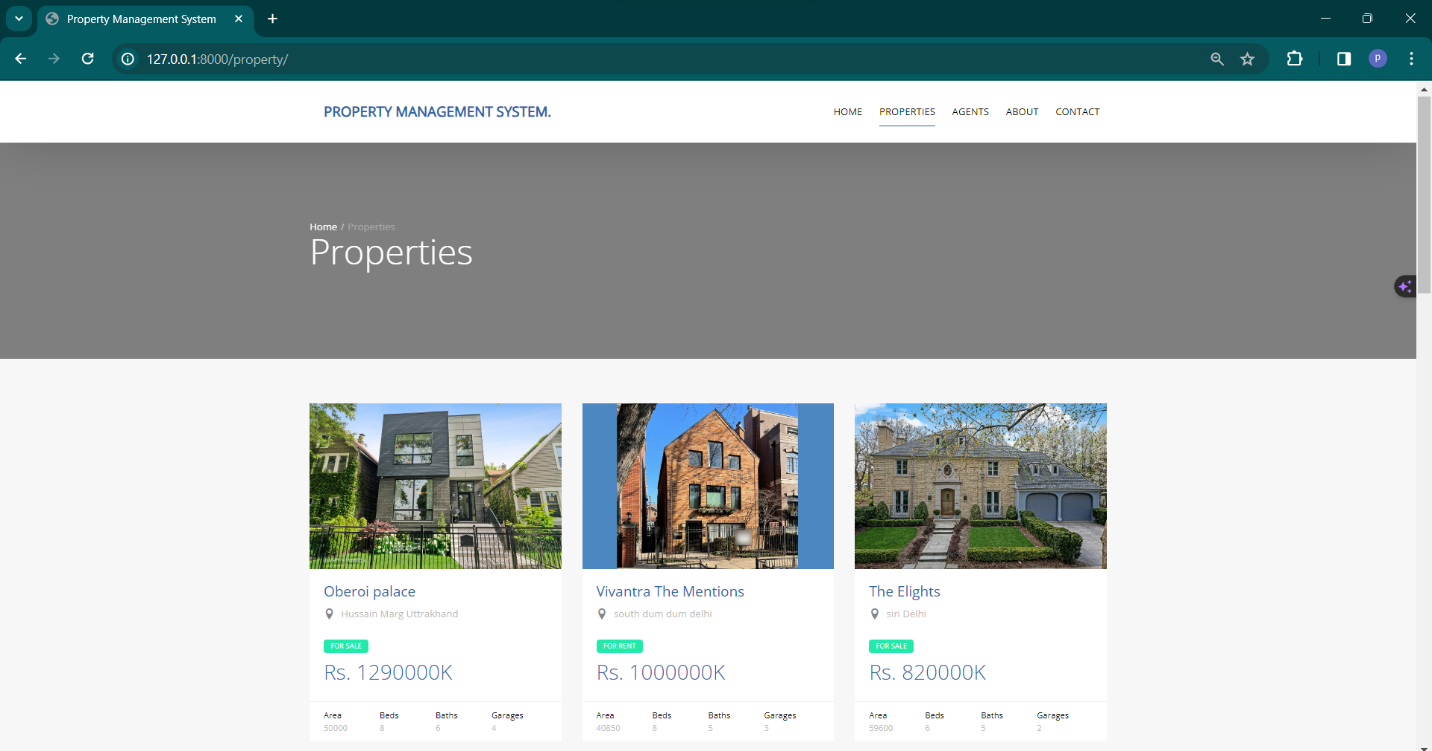


**USER PANEL**

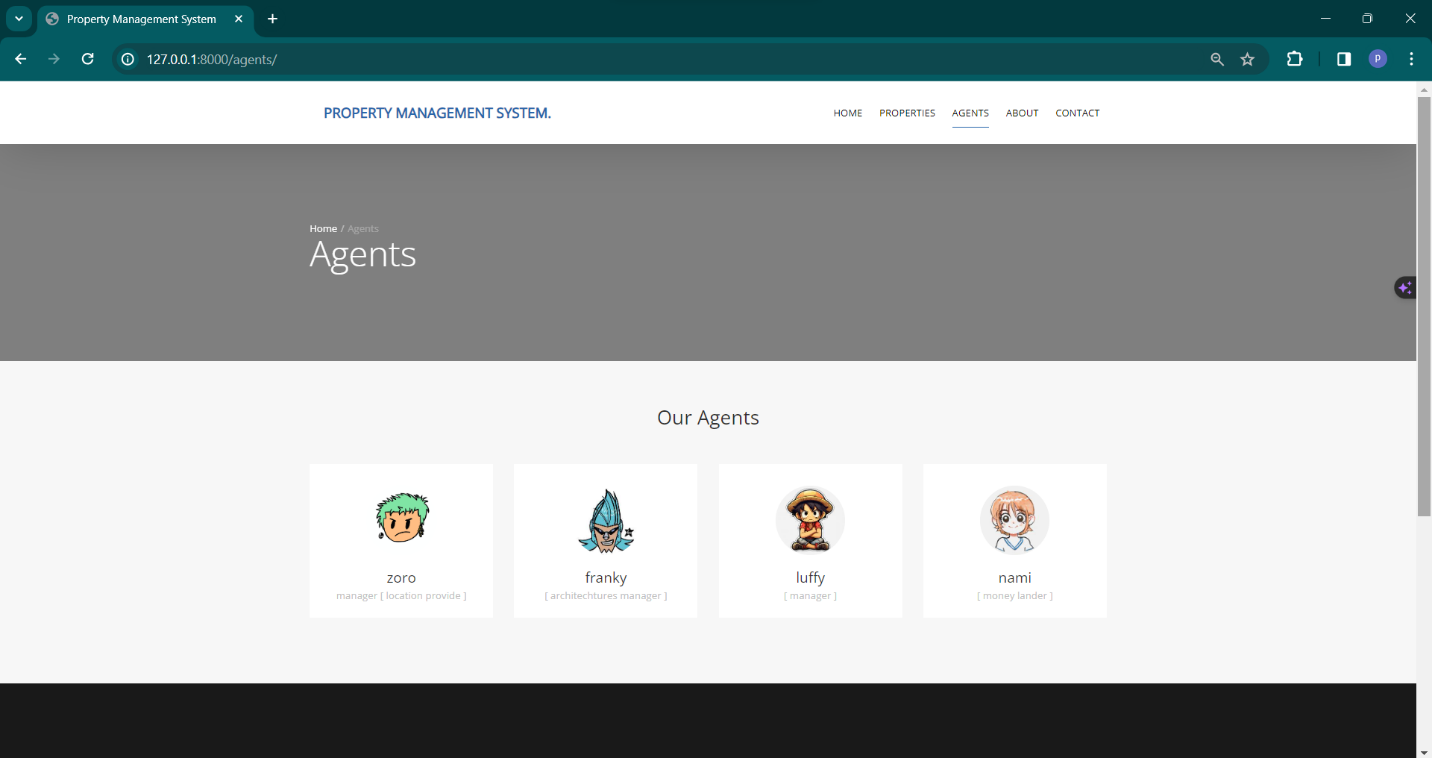
**Picture for Home screen :**



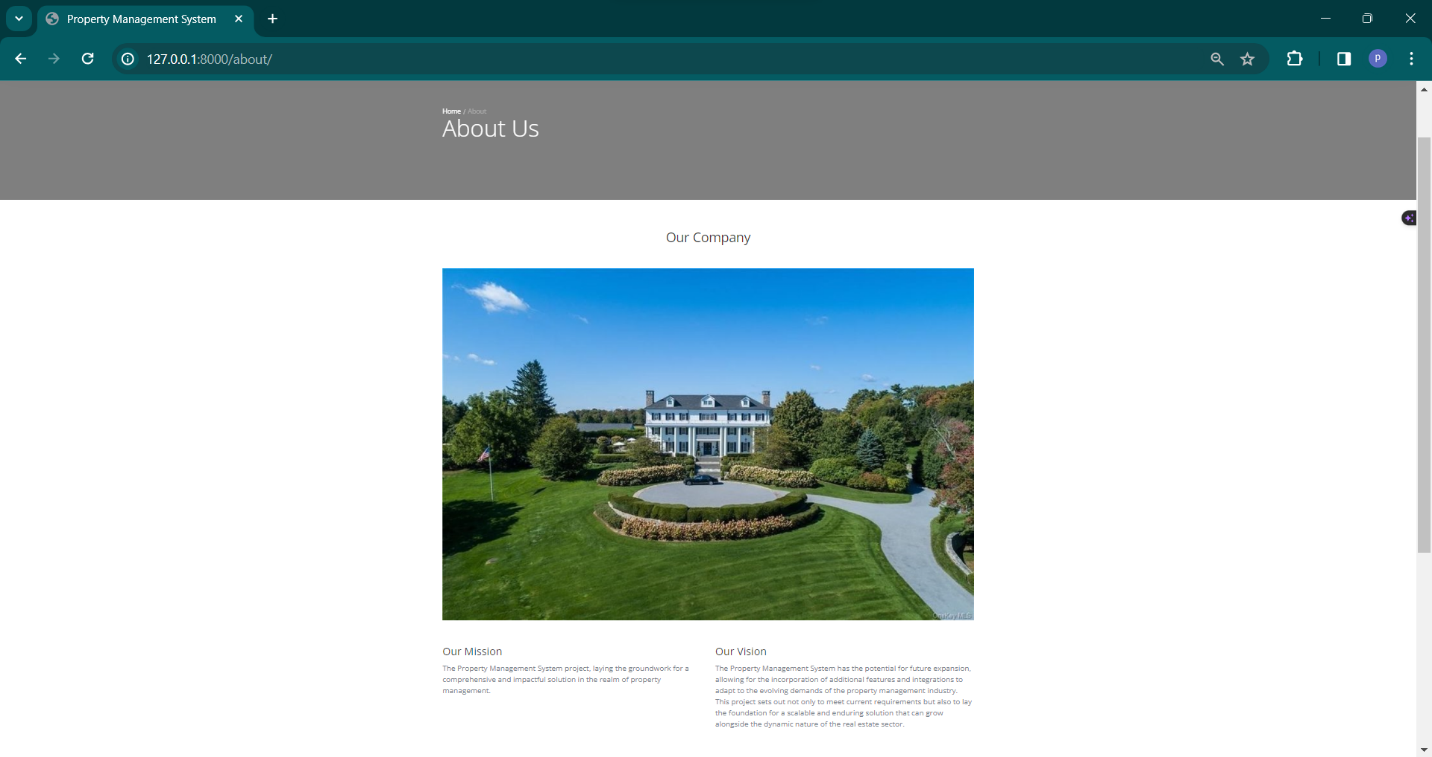
**Picture for Properties :**



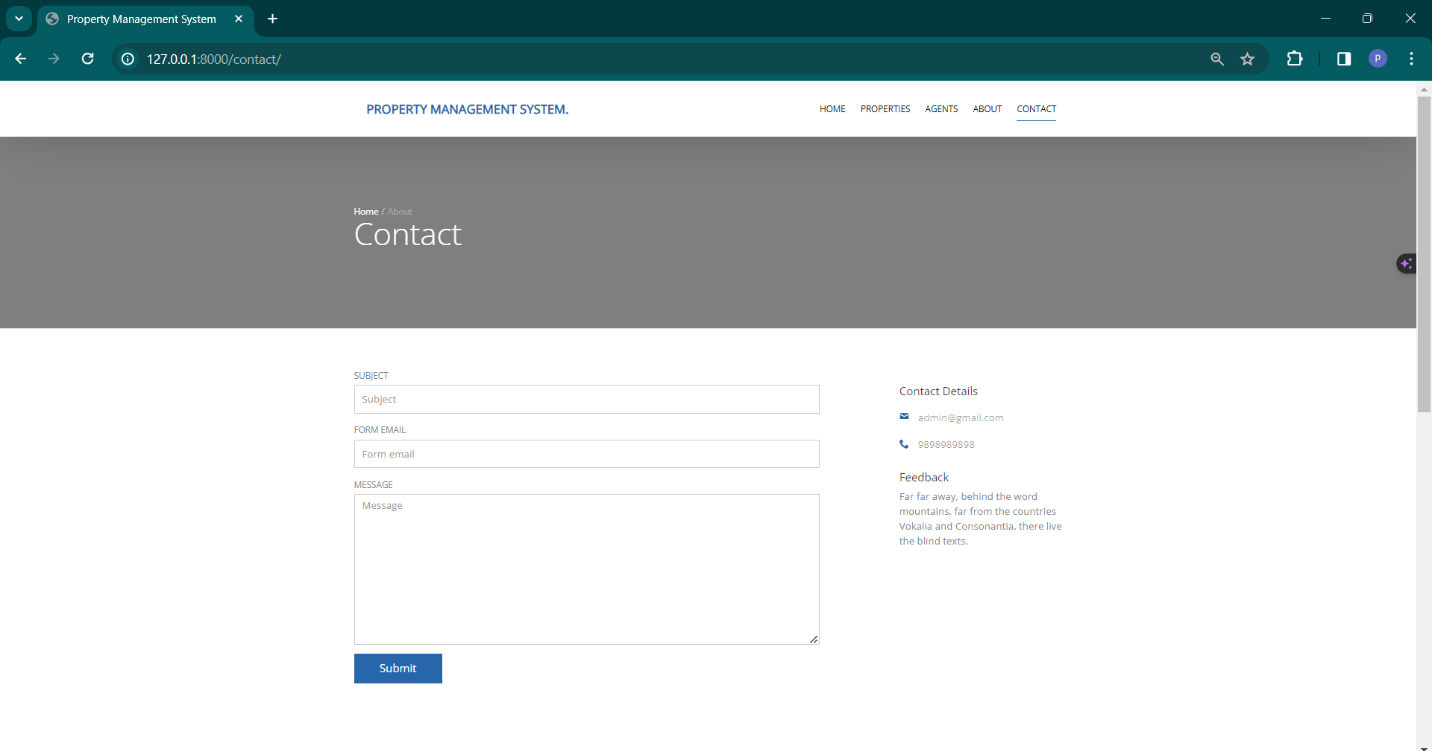
**Picture for View agents :**



**Picture for about us :**



**Picture for Contact : -**



**Future Scope and Enhancement of the Property Management System:**

The Property Management System (PMS) with Python and Django has been designed with flexibility and adaptability in mind, allowing for future enhancements and the incorporation of new features to meet evolving user needs. The software's future scope includes:

**Online Payment Gateways:**

Introducing a new module for online payment gateways can enhance the PMS by allowing tenants to make rent payments electronically. Integration with popular payment platforms can streamline financial transactions and improve user convenience.

**Authorization Facility:**

Implementing an authorization facility enhances security by determining user permissions. This feature can restrict access to confidential information, ensuring that only authorized personnel can view sensitive data.

**Authentication Improvements:**

Strengthening authentication mechanisms within the existing system can further enhance security. Advanced authentication methods, such as multi-factor authentication, can be integrated to provide an additional layer of protection.

**Improved Security Levels:**

Continuous improvement of security measures is essential. Regular updates to address emerging security threats, encryption enhancements, and adherence to industry best practices contribute to a more secure property management environment.

**Additional Utilities:**

Introducing extra utilities like a notepad and an online help system can enhance the overall user experience. A notepad feature can enable users to jot down quick notes, while an online help system provides comprehensive guidance and support.

**Mobile Responsiveness:**

Ensuring the system's responsiveness across different devices would contribute to a more versatile and accessible platform for users.

**Portability:**

The application's portability ensures its adaptability for use on different computer terminals with various operating systems and standards.

**Resource Quality:**

The resources used for project development adhere to high-quality standards, demonstrated by their Microsoft certification. This commitment to quality contributes to the robustness and reliability of the software.

As the PMS evolves, future enhancements will be undertaken by developers to keep pace with technological advancements, industry requirements, and user expectations. The software's modularity and adaptability make it well-positioned to meet the changing needs of property management, providing a reliable and efficient solution for users.

**Conclusion:**

In conclusion, the development and implementation of the Property Management System (PMS) have been a significant undertaking, resulting in a comprehensive solution tailored to meet the needs of our real estate management processes. This project aimed to streamline property listings, facilitate efficient communication with clients, and enhance the overall management of real estate assets. As we reflect on the key aspects of the project, several notable points emerge:

**Functional Property Listings:**

The system successfully delivers a user-friendly platform for managing property listings, allowing for easy navigation, property categorization, and detailed information retrieval.

**Effective Communication with Clients:**

Implementation of the Contact module enables seamless communication with clients. The integration of a contact form ensures that queries and requests are efficiently processed.

**Agent and About Sections:**

The inclusion of dedicated sections for Agents and About Us provides a comprehensive view of our team and organizational values, contributing to a more transparent and trustworthy brand image.

**Dynamic Home Page:**

The Home module dynamically presents property categories, fostering a visually appealing and informative landing page for users.

**Integration of Django Framework:**

Utilizing the Django web framework has proven to be a judicious choice, providing a robust and scalable foundation for the development of the entire system.

**Challenges Overcome**:

**Search and Filtering Functionality:**

Overcoming challenges in implementing search and filtering functionality was crucial. The final system now allows users to search for properties based on various criteria, enhancing the user experience.

**Reservation System:**

The integration of a reservation system required careful consideration of data models and user interactions. The Reserve module now facilitates efficient property reservation management.

**Website Reference:-**

1. [www.github.com](http://www.github.com)
2. [www.javaworld.com](http://www.javaworld.com)
3. [www.djangoproject.com](http://www.djangoproject.com)
4. [www.w3schools.com](http://www.w3schools.com)
5. [www.m.youtube.com](http://www.m.youtube.com)