Project

Complaint Management System

Project submitted to the

APSSDC

Submitted By

Praveen Yeggada – 23A91A61J4

Likhitha Yeduvaka – N200935

Geenthanjali Jillella – N190995

Parthiv Naga Vardhan Pati – 23A91A61H7

Under the guidance of

Srikanya Tamada

Revati Y

June 2025

TABLE OF CONTENTS

ABSTRACT

This project entails the comprehensive design and implementation of a web-based Complaint Management System (CMS) developed using the powerful Django web framework. The main intent of this system is to provide an efficient, structured, and digital platform for institutions, organizations, and educational establishments to handle and resolve complaints effectively. Traditional methods of managing complaints often rely on paper-based systems or informal verbal communication, which are prone to mismanagement, delay, and data loss.

With this CMS, users can register themselves, securely log in, and submit complaints using a simple and responsive web interface designed to be accessible on both desktop and mobile devices. Each user is provided with a dashboard to track the status of their complaints—whether pending, resolved, or rejected.

Administrators have privileged access to a centralized admin dashboard from where they can view, filter, categorize, and respond to user complaints. The admin interface provides real-time insights into complaint status, helps prevent complaint duplication, and ensures a transparent record of every issue reported and resolved.

Overall, this project significantly reduces manual errors and the effort associated with traditional complaint handling processes. It enhances operational transparency, traceability, user satisfaction, and contributes to an organized and efficient workflow within institutions.

INTRODUCTION

In today's digital world, organizations and institutions are rapidly adopting software systems to improve internal communication and service management. One such critical area that demands efficiency is the management of user complaints, suggestions, and grievances. Typically, complaints are raised through verbal communication, handwritten notes, or informal emails, leading to a lack of tracking, responsibility, and follow-up. As a result, complaints may go unresolved, causing dissatisfaction and miscommunication.

The Complaint Management System (CMS) addresses these challenges by providing a centralized, automated, and accessible platform that facilitates smooth handling of user-submitted complaints. It digitizes the complete lifecycle of a complaint—from submission and tracking to resolution and closure. This approach not only ensures that every complaint is acknowledged but also provides users with transparency and accountability, knowing the progress of their issues at any time.

The system offers various features that support its objectives:

- A role-based access system, ensuring users and administrators have different views and permissions.

- Real-time complaint status updates visible to users on their dashboards.

- Efficient data management using Django’s ORM to store, retrieve, and manipulate data reliably.

- Use of modern UI components through Bootstrap and Material Dashboard for a professional and user-friendly interface.

The core goal of this system is to minimize response time, increase service quality, and build trust between the users and the administration. The Complaint Management System provides a sustainable solution to a real-world administrative problem, paving the way for improved workflow, user experience, and institutional productivity.

TECHNOLOGY STACK USED

This project leverages a powerful combination of backend and frontend technologies to build a responsive and scalable web application. Each technology was selected based on its relevance, ease of integration, and ability to meet the objectives of the system.

- Backend: Python 3 & Django Framework

Python is a high-level, versatile programming language, and Django is a robust, open-source web framework built on Python. Django provides an out-of-the-box admin panel, ORM (Object Relational Mapping), authentication system, and security features like CSRF protection and input sanitization.

- Frontend: HTML5, CSS3, Bootstrap 4, JavaScript

HTML5 is used to structure the web pages, while CSS3 styles them to provide a clean and modern look. Bootstrap 4, a mobile-first UI toolkit, enables rapid front-end development with its predefined components and responsive grid system. JavaScript adds interactivity and enhances the user experience.

- Templates: Django Templating System

Django uses its built-in templating engine to dynamically render HTML pages with data from the server. It allows integration of logic into HTML files using template tags and filters, reducing redundancy and improving maintainability.

- Database: SQLite3

SQLite is a lightweight, file-based relational database that requires no separate server process. It's ideal for development and small-scale deployments, providing sufficient performance and easy integration with Django's ORM.

- Tools & Libraries:

- Django Admin Panel – Offers a ready-made interface for administrators to manage models (complaints, users, etc.).

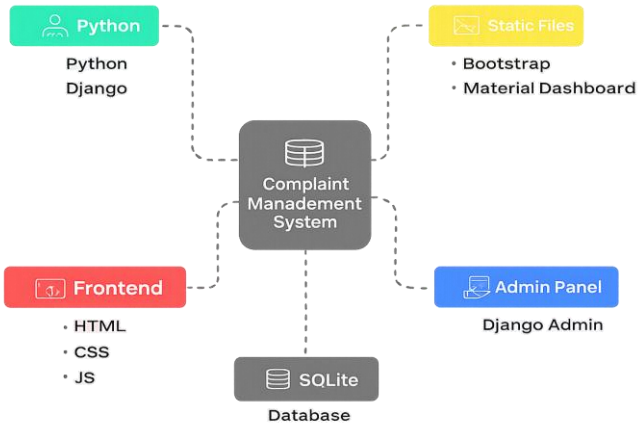
- Material Dashboard – A modern UI theme used to create an aesthetically pleasing admin dashboard.

- jQuery – A JavaScript library used to simplify DOM manipulation and handle asynchronous operations.

- Chart.js – Used to visualize data in graphical format, such as pie charts or bar graphs representing complaint statistics.

This technology stack allows for rapid development, scalability, and a seamless user experience across devices.

ARCHITECTURE



The architecture of the Complaint Management System is designed using a modular and scalable structure based on Django's Model-View-Template (MVT) pattern. The system is logically divided into layers that interact seamlessly to provide a robust, secure, and user-friendly experience.

- Backend Layer (Python & Django)

This layer contains the core logic of the application. It handles routing, session management, authentication, form processing, and database interaction. Django's built-in capabilities make it easy to define models, write views, and manage user authentication.

- Technologies Used: Python 3, Django Framework
- Responsibilities: Handles business logic, request-response cycles, and integrates models with templates.

- 2. Static Files Layer

Static files enhance the visual and interactive components of the system. They include stylesheets, JavaScript files, icons, and dashboard UI libraries.

- Technologies Used: Bootstrap, Material Dashboard
- Responsibilities: Manages CSS for layout, JS for interactivity, and dashboard UI components.

- 3. Frontend Layer

The frontend is responsible for presenting information to users in a structured and intuitive way. It consists of HTML templates dynamically rendered using Django's templating engine.

- Technologies Used: HTML5, CSS3, JavaScript
- Responsibilities: Renders data received from the backend, handles client-side interactions, and provides responsive design.

- 4. Admin Panel Layer

Django's built-in admin interface serves as a powerful tool for administrators to manage users, complaints, and system data without custom development. It uses Django's ORM to interact with the database and provides full CRUD (Create, Read, Update, Delete) capabilities.

- Tool Used: Django Admin
- Responsibilities: Admin dashboard for viewing and managing complaints, users, and status changes.

- 5. Database Layer (SQLite)

This is the data storage layer used to persistently store all system data including user records, complaint details, statuses, and admin actions. SQLite was chosen for development due to its simplicity and zero-configuration.

- Database Used: SQLite
- Responsibilities: Stores structured data and supports query execution via Django ORM.

- Workflow Summary

1. Users interact with the Frontend Layer via web pages.
2. Requests are routed to the Backend Layer, where Django views process the data.
3. Data is stored and retrieved from the Database Layer using Django models.
4. The Static Files Layer enhances the presentation with styles and JavaScript interactions.
5. Administrators manage the system through the Admin Panel Layer.

This architecture ensures separation of concerns, which leads to easier debugging, testing, and future scalability. It's a proven, maintainable approach ideal for institutional web applications like a complaint management portal.

IMPLEMENTATION

The implementation phase focuses on turning the system design into a fully functional web application. The Complaint Management System consists of multiple interrelated components, each handling specific functionality within the application.

- User Registration and Authentication

New users can register using a simple form that captures their details. Django's built-in authentication system handles password hashing, session management, and secure login. Role-based access ensures that only authorized users can access certain pages.

- Complaint Submission

After logging in, users can submit new complaints through a structured form. Fields such as subject, description, and category are validated before being stored in the database. Each complaint is linked to the user's profile, ensuring traceability and security.

- Complaint Management by Admins

Admins have access to a dashboard powered by the Django admin interface and styled with Material Dashboard UI components. From here, they can view all complaints in tabular or graphical form, change the status (e.g., Pending, Solved, Rejected), and delete inappropriate or resolved entries.

- Dashboard and Tracking

Both users and admins have dashboards tailored to their roles. Users see a summary of their submitted complaints along with status updates, while admins have a complete view of the system. Charts, graphs, and tables help in quick visual analysis.

- Responsive Design and UI Enhancements

The frontend uses Bootstrap and Material Dashboard for responsive design and visual appeal. Whether accessed on a mobile phone, tablet, or desktop, the interface adjusts automatically. Alerts, modals, and button interactions are powered by JavaScript and jQuery to enhance usability.

- Error Handling and Validation

Both client-side and server-side validation is implemented to prevent malformed data. Django's form system provides feedback in case of missing or incorrect input. Custom error pages improve the user experience during unexpected failures.

This comprehensive implementation ensures that the system is not only functional but also secure, user-friendly, and scalable for future extensions like notifications, search filters, or API integration.

ADVANTAGES

- Centralized Complaint Tracking

All user complaints are stored and managed in a centralized database. Easy to retrieve, update, and monitor complaint history.

- Faster Response Time

Real-time status tracking enables quicker resolution. Administrators can act on complaints immediately through the dashboard.

- Secure User Authentication

Role-based access using Django's built-in authentication system ensures that users and admins see only what they're allowed to.

- Responsive User Interface

The UI is built with Bootstrap and Material Dashboard, offering a clean, modern, and mobile-friendly design.

- Easy to Maintain and Scale

Developed using Django's MVT (Model-View-Template) architecture, making the code modular and easy to extend. Easily scalable for more users, departments, or features in the future.

- Visual Reporting and Analytics

Admins can view trends using visual tools like charts (using Chart.js), aiding decision-making and performance review.

- Efficient Admin Panel

Django's admin interface allows for efficient backend management without needing to build a custom interface from scratch.

- User Empowerment

Users can track the progress of their complaints, which builds trust and ensures transparency.

- Automation of Workflow

Reduces dependency on manual complaint logs (emails, paper forms, etc). Streamlined workflow ensures no complaint is missed or delayed.

RESULT AND OUTPUT

Upon successful development and deployment of the Complaint Management System, the platform demonstrated a robust and user-friendly interface that significantly enhanced the overall complaint management process. Users were able to register effortlessly, log in securely, and submit their complaints through a well-structured online form. The complaint submission process was streamlined, with validation mechanisms ensuring that all required data was accurately captured and stored.

Users benefited from the real-time status tracking of their complaints. Each complaint could be viewed in a personalized dashboard, where users could see whether it was pending, resolved, or rejected. This transparency led to increased user satisfaction and trust in the system.

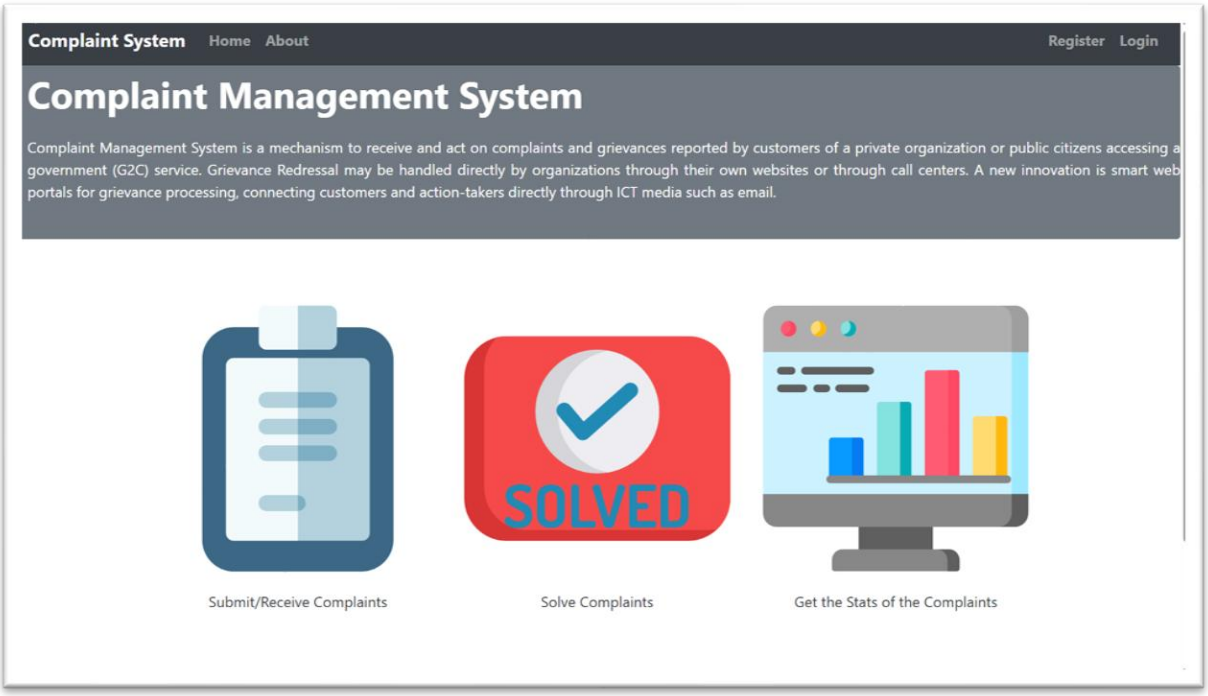
For administrators, the backend interface provided through the Django Admin Panel was both powerful and intuitive. Admins were able to view all submitted complaints, apply filters, sort based on status or submission date, and take immediate action by changing the status or removing irrelevant entries. The admin dashboard utilized Material Dashboard components, giving it a professional, clean layout with widgets and data summaries.

The front-end UI used Bootstrap 4 and responsive design principles, ensuring compatibility with a wide range of devices including smartphones, tablets, and desktops. Features such as modern card-based layouts, collapsible menus, and graphical data visualizations using Chart.js contributed to a rich and interactive user experience.

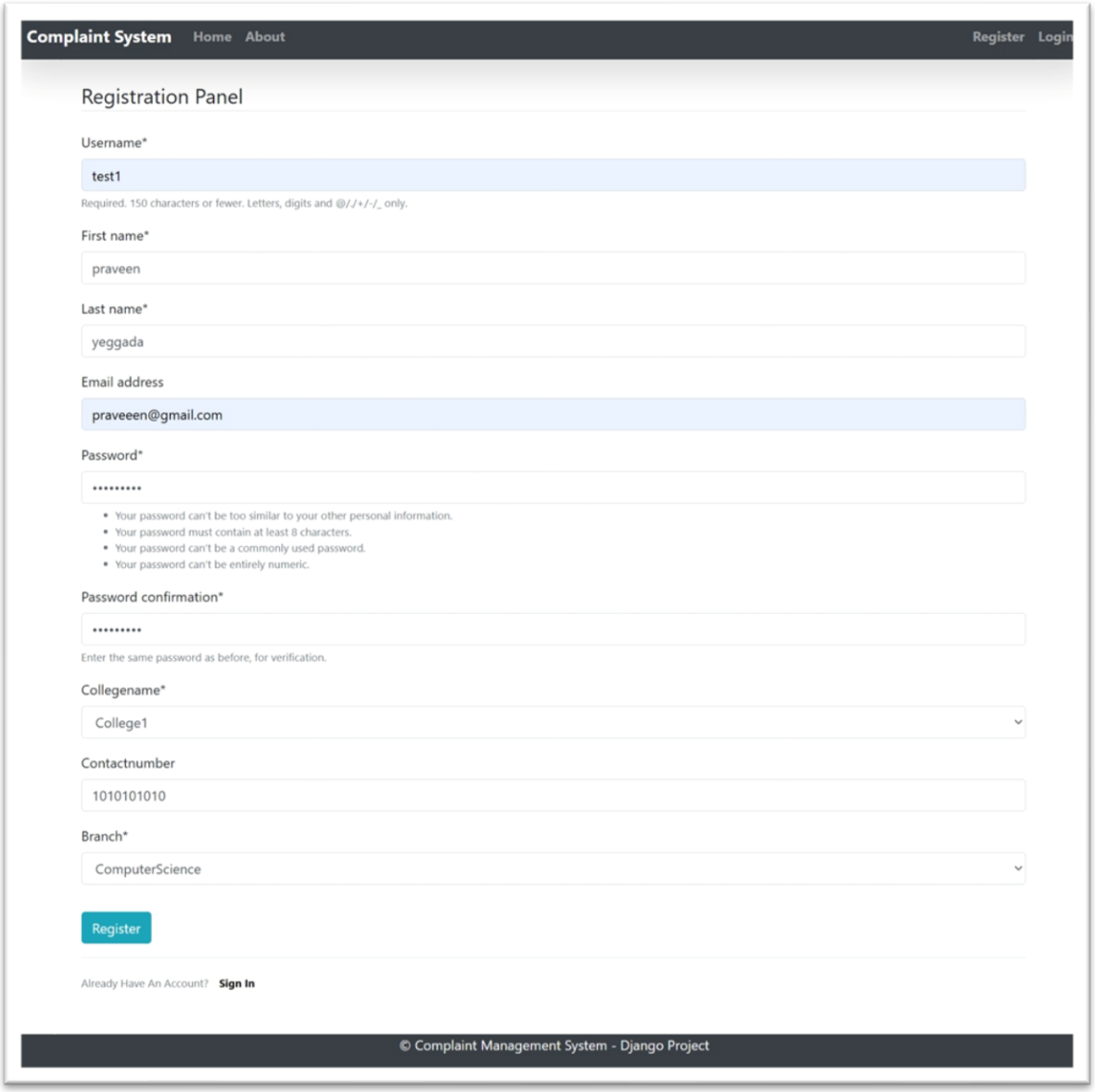
Furthermore, user interface screenshots displayed visually appealing components like input forms, notification alerts, charts, and navigation panels—showcasing the platform's capability as a complete end-to-end solution for digital complaint tracking within any organization or academic institution.

Home_Page:

About_Page:

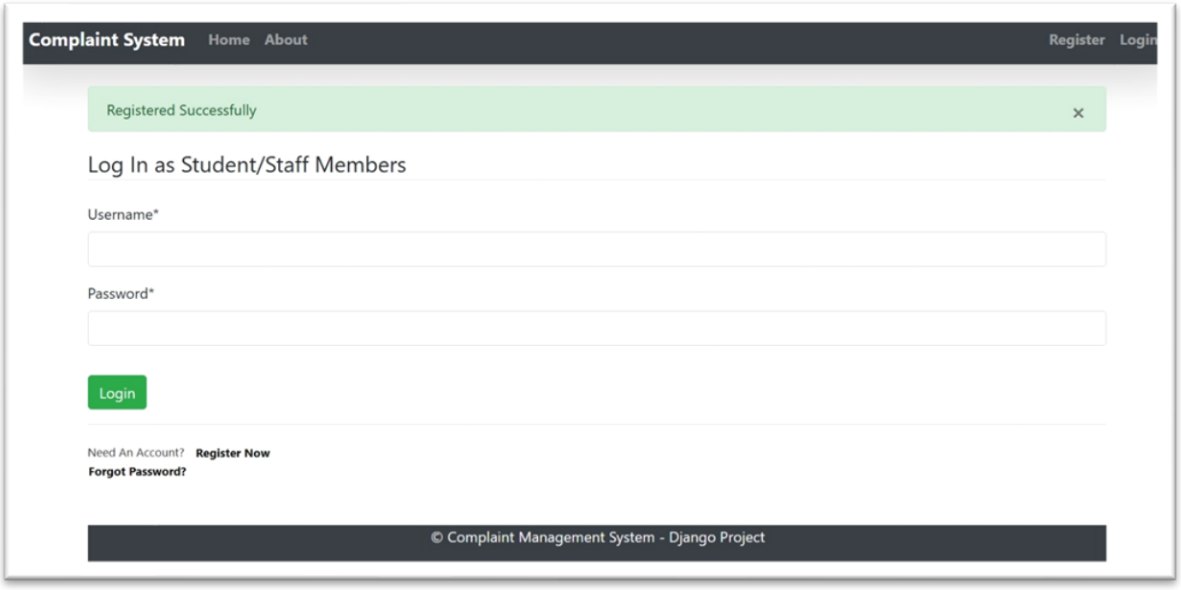


Registration_Page:

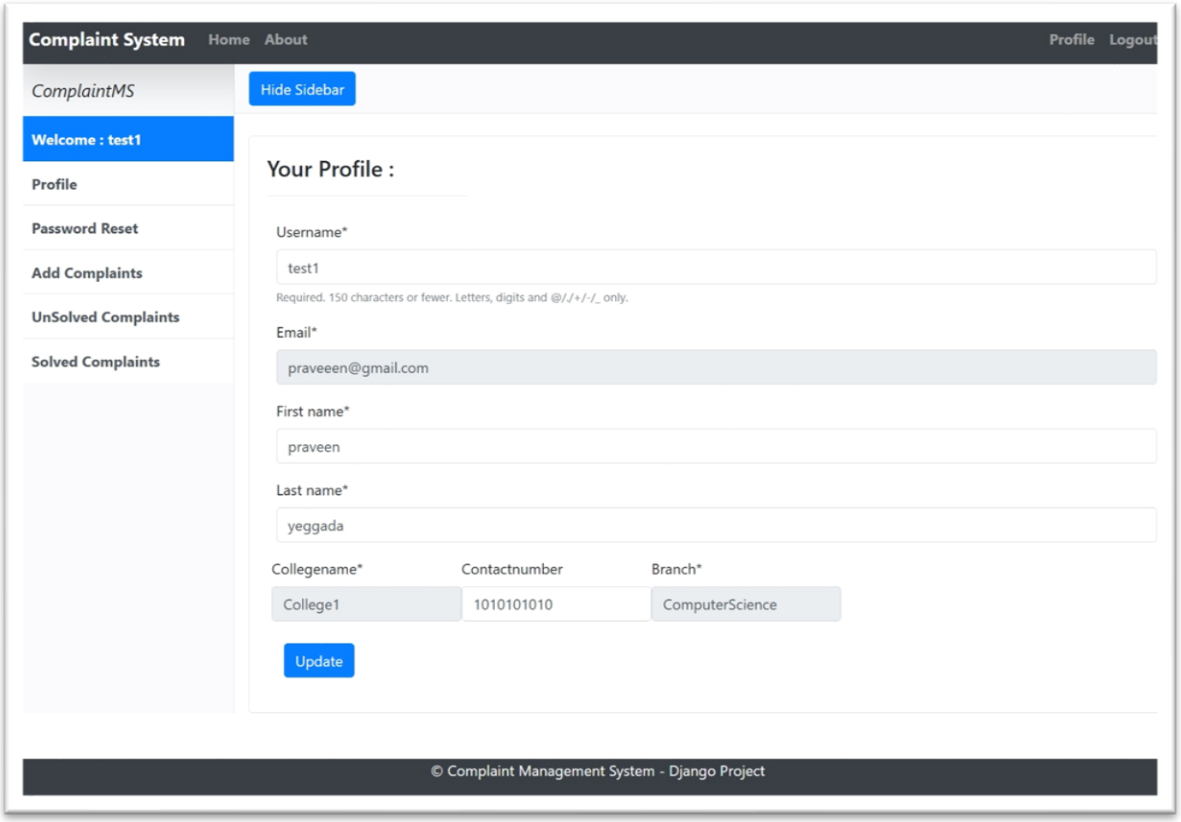


Login_Page:

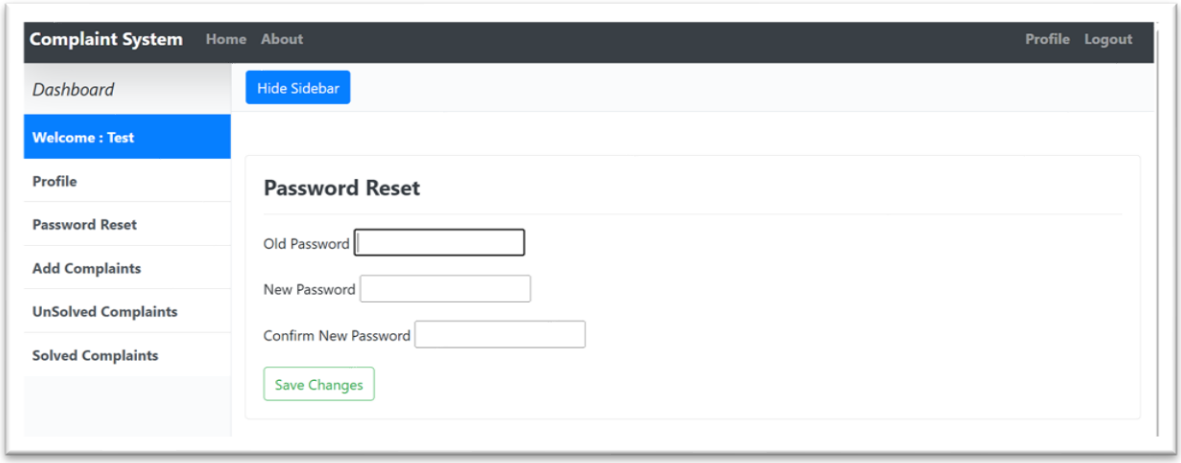
User_Login_Page:



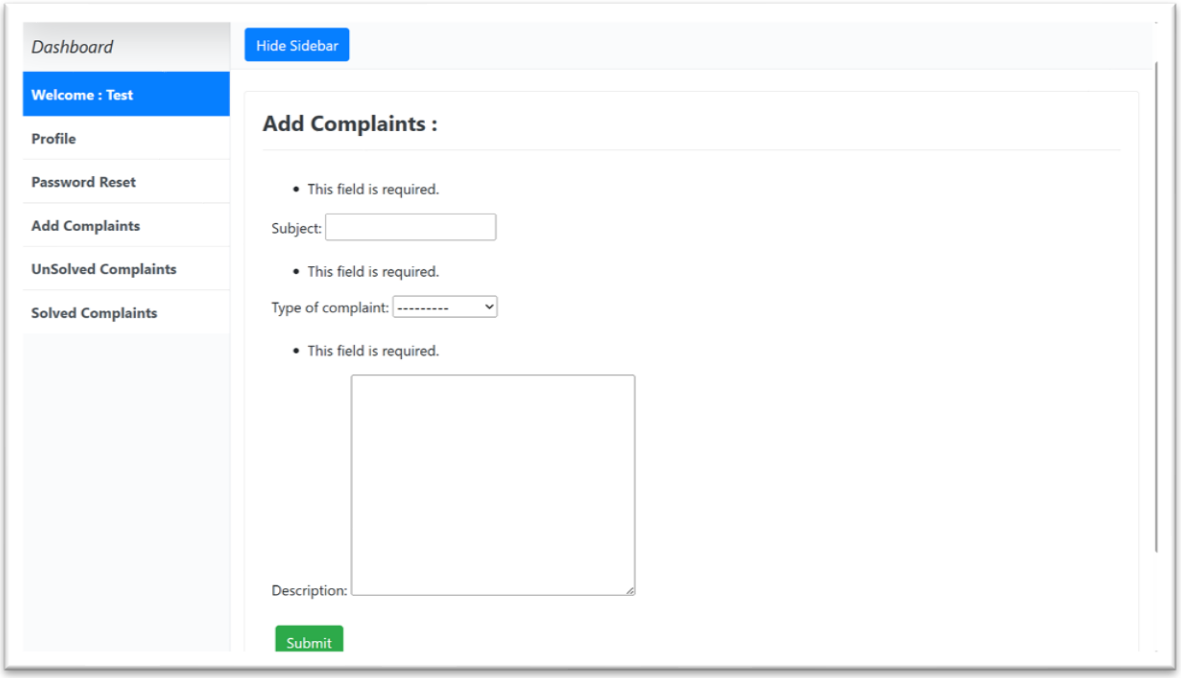
User_Profile_Page:



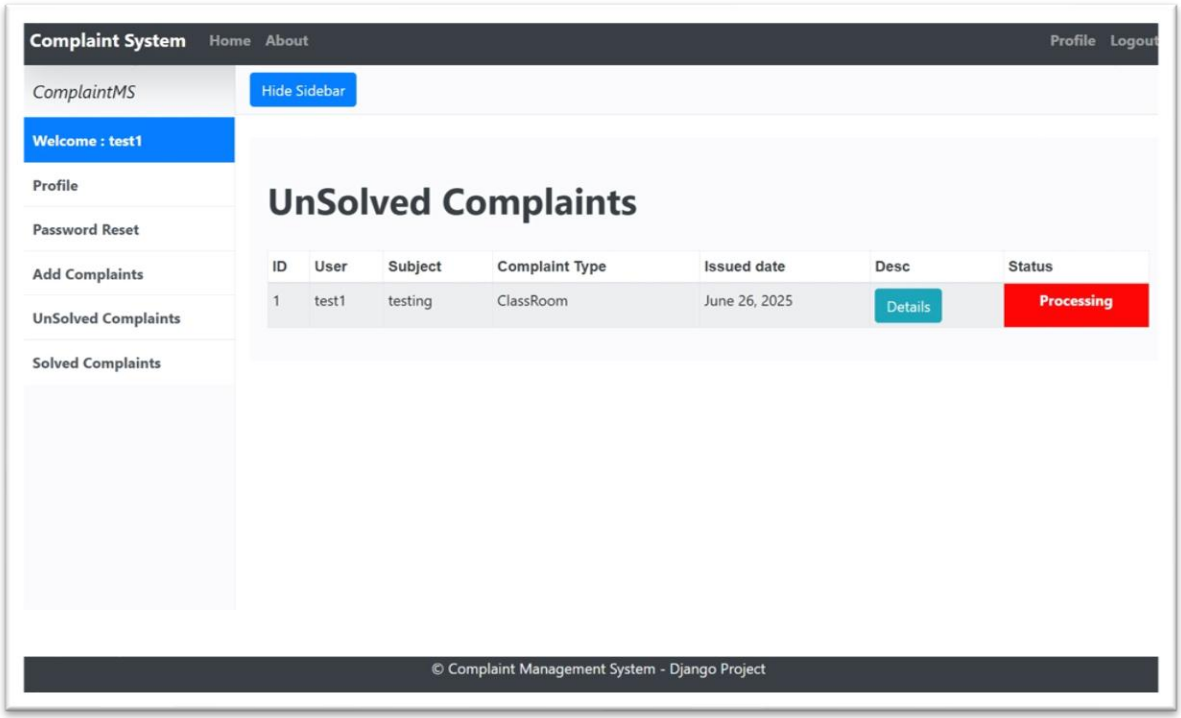
Password_Reset_Page:



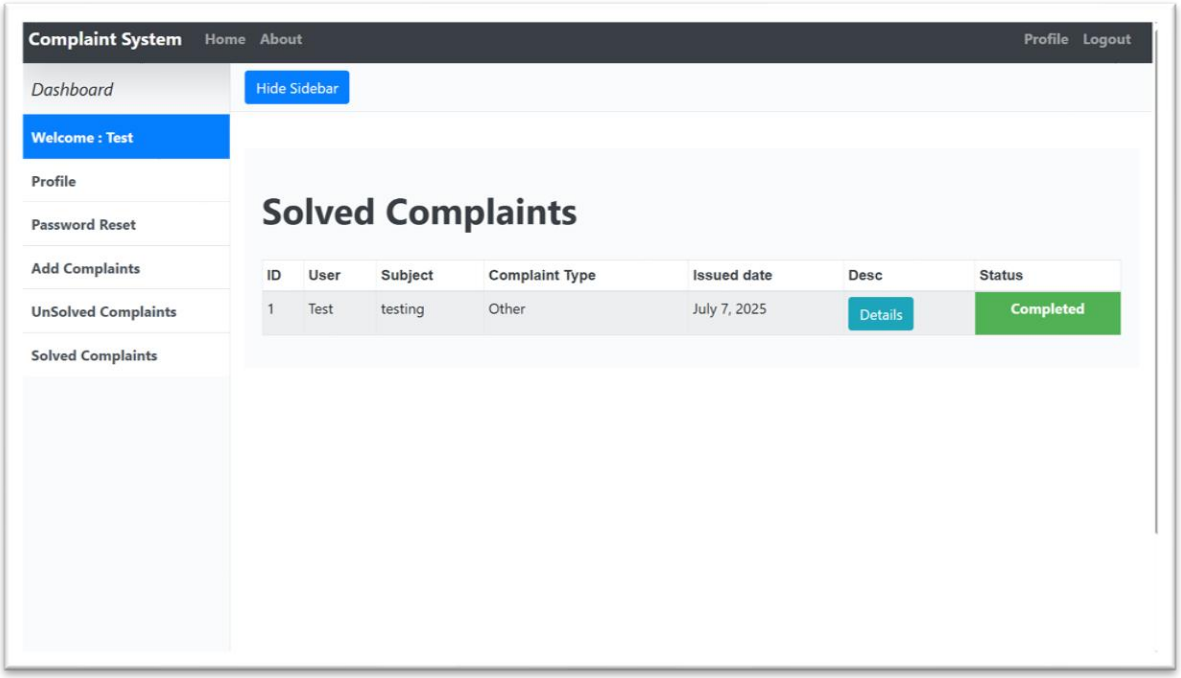
Complaint_Submit_Page:



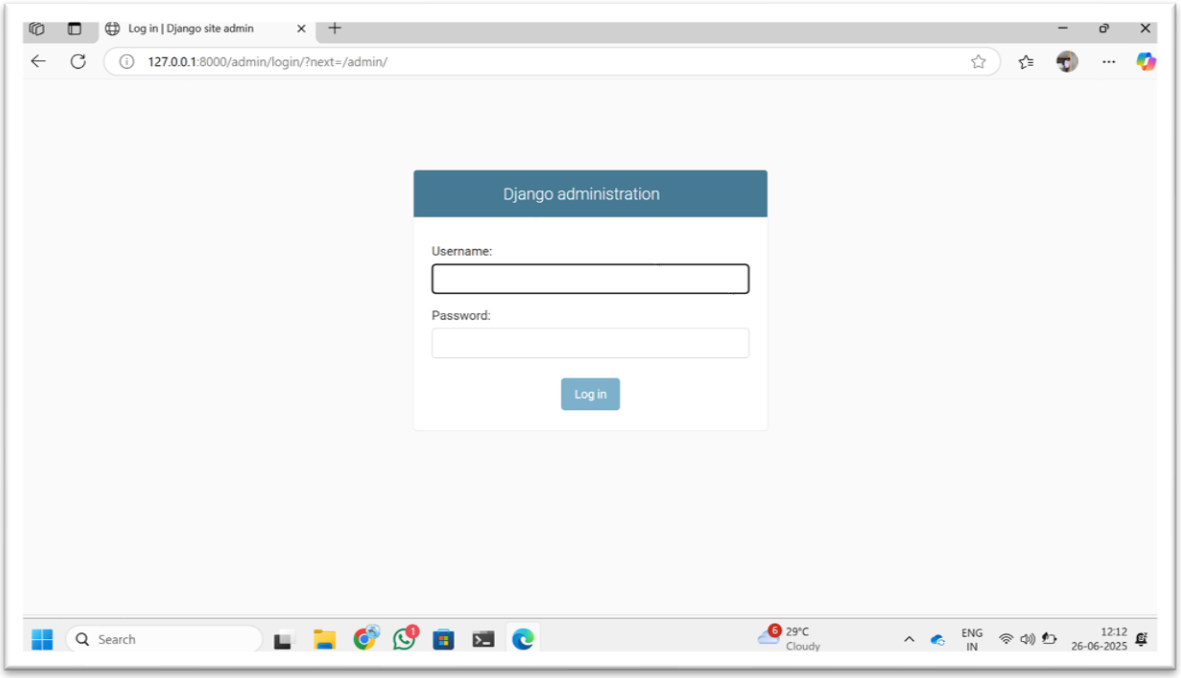
Unsolved_Complaints_Page:



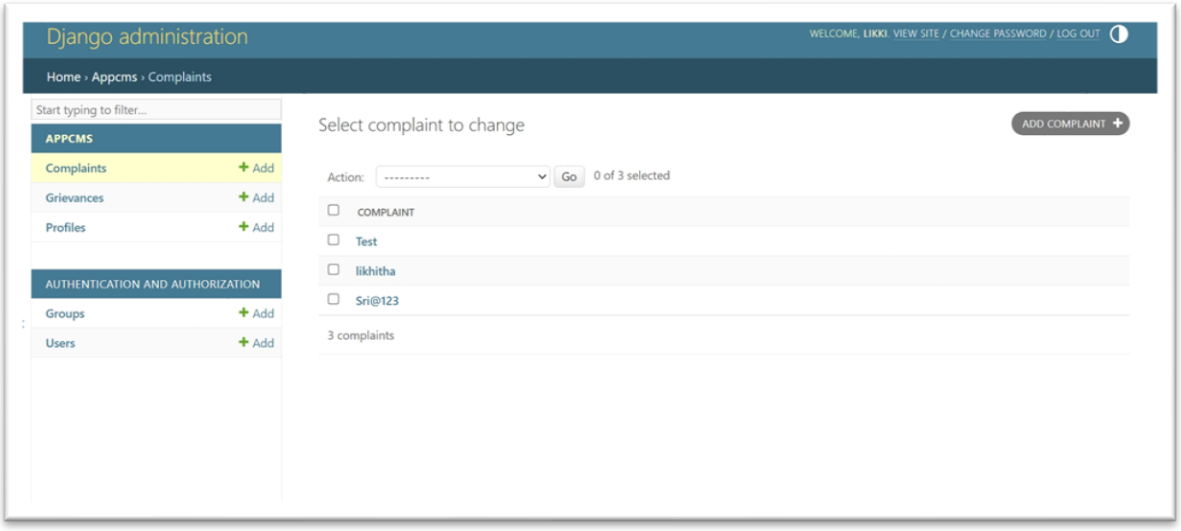
Solved_Complaints_Page:



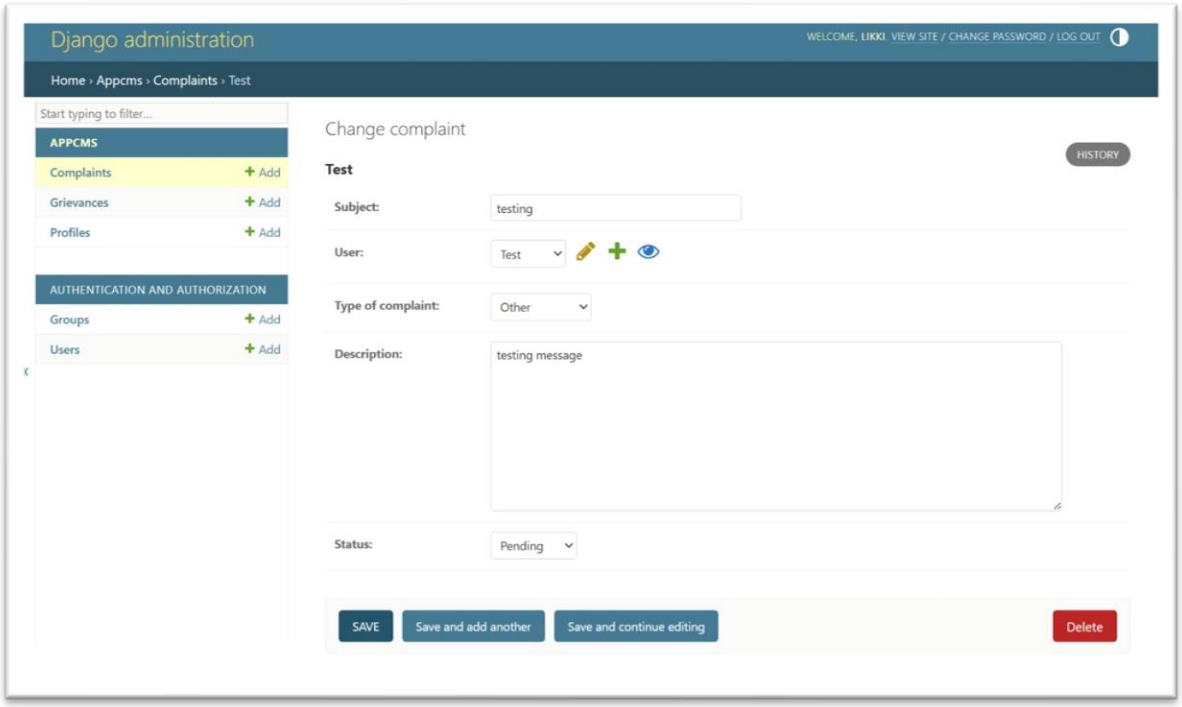
Admin_Login_Page:



Complaints_Page:



Complaints_Update_Page:



CONCLUSION

The Complaint Management System has successfully fulfilled its intended goal of transforming a manual, inefficient complaint handling process into a modern, streamlined digital platform. By automating user registration, complaint submission, and administrative review, the system improves productivity, enhances accountability, and ensures no complaint goes unnoticed. The modular architecture of the platform allows for easy updates and maintenance, making it a future-ready solution for scalable deployment.

The platform not only reduces workload for administrators but also increases transparency, traceability, and responsiveness, which are critical for user trust and operational efficiency. It provides a centralized view of all complaints, making it easier to track trends, identify recurring issues, and implement solutions proactively.

Looking ahead, several enhancements can be implemented to extend the system's capabilities and improve user engagement:

- Real-time Chat Support: Integrating a chat module for users to interact directly with admin staff regarding their complaints.

- File Upload Support: Allowing users to attach relevant documents or images for better context and evidence.

- Email and SMS Notifications: Notifying users about complaint status updates or administrative responses instantly.

- Role-Based Access Control (RBAC): Introducing fine-grained user roles like department admins, superusers, and general staff.

- Data Analytics Dashboard: Displaying complaint trends, resolution times, department-wise distribution, and other metrics using graphical tools.

- Search and Filter Capabilities: Helping users and admins quickly locate specific complaints or users.

By implementing these features, the CMS can evolve into a more comprehensive, intelligent, and user-centric platform. The flexibility of the Django framework and the scalability of its design make it well-suited for these future developments. The project stands as a strong foundation for building robust issue management solutions in both academic and enterprise environments.

REFERENCES

1. Bootstrap Framework – https://getbootstrap.com/

   - <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.7/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-LN+7fdVzj6u52u30Kp6M/trliBMCMKTyK833zpbD+pXdCLuTusPj697FH4R/5mcr" crossorigin="anonymous">
   - <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.7/dist/js/bootstrap.bundle.min.js" integrity="sha384-ndDqU0Gzau9qJ1lfW4pNLlhNTkCfHzAVBReH9diLvGRem5+R9g2FzA8ZGN954O5Q" crossorigin="anonymous"></script>

2. Django Documentation – https://docs.djangoproject.com/en/4.0/

3. SQLite Documentation – https://www.sqlite.org/docs.html

4. Icons from Flaticon – https://www.flaticon.com/