**Campus Connect -Task Management System**

**Project submitted to the**

**APSSDC**

**Bachelor of Technology**

**In**

**Computer Science and Engineering**

**Aditya College Of Engineering And Techology**

**Submitted By**

Babitha Esther Rani Anguluri (23P31A05A1)

Bhavana Shanmukhi Baswa (23P31A0596)

Shankara Mani Sai Nookalamanthi (23P31A05N5)

Mahammad Abubakar Siddiq (23P31A05M3)

Chedala Mohana Chinna Jagadeeshwara Reddy (23P31A05K3)



**Under the guidance of**

T.Srikanya

Revathi

**June 2025**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE NO** |
| 1 | Abstract | 3 |
| 2 | Introduction | 4-5 |
| 3 | Technical Setup & Prerequisites | 6 |
| 4 | System Design & Workflow Architecture | 7-9 |
| 5 | Analytical Tools & Visualization | 10-11 |
| 6 | Key Features & Benefits | 12 |
| 7 | Project Screenshots / Output Screens | 13-18 |
| 8 | Conclusion | 19 |
| 9 | Reference Materials & Learning Resources | 20-21 |

**ABSTRACT**

This study presents a detailed exploratory analysis of student task management and engagement using Python’s powerful web framework, Django, along with front-end tools like HTML, Bootstrap, and responsive design practices.

The project, titled CampusConnect, manages over 5,000 records of student activities from multiple academic departments, covering academic years from 2018 to 2024. It captures multiple aspects of student productivity, such as task creation and completion status, user roles (Admin, Staff, Student), deadlines, submission files, feedback, and user engagement.

**The primary objectives of this project are as follows:**

* **Role-Based Access:** CampusConnect provides custom dashboards for Admins, Staff, and Students. Each role has specific permissions for managing or viewing tasks.
* **Task Monitoring:** The platform allows real-time task assignment, progress tracking, and completion marking, ensuring efficient workflow among students and faculty.
* **Performance Highlights:** Using dashboard summaries and visual indicators, it identifies students who consistently meet deadlines and tracks departments with the highest engagement.
* **Submission Handling:** Students can upload files as proof of completion. Staff and Admins receive automated notifications when tasks are marked as done.
* **UI/UX Enhancement:** With the use of Bootstrap, the project offers a clean, visually appealing interface that improves user interaction and accessibility across devices.

Throughout this project, Django is used for model management, view handling, and user authentication, while HTML and Bootstrap contribute to building responsive and intuitive web pages. Additional features include email notifications, file upload support, and visual task tables.

The study’s outcomes contribute valuable insights for academic administrators, faculty, and student coordinators, enabling data-driven strategies to boost student productivity and accountability. Additionally, the integration of Python and Django proves to be a robust and scalable toolkit for academic management systems and can be extended to similar domains such as internship tracking, research management, and academic counseling.

**INTRODUCTION**

In an era where academic efficiency, digital collaboration, and student accountability are more important than ever, effective task management within educational institutions has become a critical need. Students often struggle with managing multiple assignments, meeting deadlines, and staying organized—challenges that can lead to stress, reduced performance, and missed academic opportunities. Institutions, on the other hand, face difficulties in tracking student progress, ensuring timely submissions, and maintaining consistent communication among students, staff, and administrators.

To address these challenges, **CampusConnect** was developed—a comprehensive web-based task management platform designed to streamline student responsibilities and academic workflows. Built using Python’s Django framework, the project leverages a robust backend combined with responsive front-end tools like HTML and Bootstrap to deliver a user-friendly experience for all stakeholders.

This project aims to provide a centralized solution for task allocation, progress monitoring, submission handling, and communication, covering academic years. The platform includes diverse functionality such as:

* Role-based access for **Admin**, **Staff**, and **Students**
* Task creation and assignment with deadlines and descriptions
* File upload capability for student submissions
* Automated email notifications upon task completion
* Task status tracking and feedback mechanisms
* Dashboard summaries to visualize task engagement metrics

Using Django’s ORM and templating system, the project begins by structuring models for users, tasks, and roles to ensure clean, scalable, and secure data handling. The analysis and insights include:

* Identifying students and departments with the highest task completion rates
* Monitoring engagement trends across academic years and roles
* Highlighting overdue or pending tasks for proactive intervention
* Correlating performance data with task submission patterns
* Visualizing usage statistics through dynamic and responsive dashboards

With the help of **Matplotlib** and **Seaborn**, the platform also supports backend-powered visual analytics—rendering line graphs, bar charts, and pie charts to represent progress and engagement levels. These visuals enable administrators and faculty to make informed academic decisions at a glance.

Through this initiative, CampusConnect provides a scalable and insightful solution for academic institutions, empowering them to improve student performance, foster accountability, and streamline internal workflows. Beyond student task management, this platform demonstrates the power of **Python, Django, and data visualization** in solving real-world educational challenges, and it lays the groundwork for future extensions such as research tracking, attendance systems, and project evaluation tools.

Ultimately, **CampusConnect** showcases the potential of data-driven platforms to drive academic success and institutional efficiency.

**Technical Setup & Prerequisites**

**Hardware:**

* Processor: Intel Core i3 or above
* RAM: 4 GB minimum
* Storage: 500 MB of available disk space
* Display: 13-inch monitor (min) with 1366×768 resolution

**Software:**

* OS: Windows/Linux/macOS
* Python 3.6 or above
* Django 3.2+
* SQLite
* Google Chrome
* Required Python Libraries:
  + Django
  + Sqlite3
  + Os
  + Email
  + Bootstrap

**System Design & Workflow Architecture**

The CampusConnect system follows the classic Django MVT (Model-View-Template) architecture, which separates the data (models), business logic (views), and presentation (templates). The system is designed to support role-based user interaction where each user type (Admin, Staff, Student) has access to features specific to their role.

**1. MVT Architecture in CampusConnect**

**Model**

* Defines the structure of the database.
* Custom user model CustomUser stores user details along with roles (Admin, Staff, Student).
* Task model handles task-related information like title, description, assigned user, due date, status, and attached files.
* Additional models like Submission or StudentProfile can be added if needed.

**View**

* Contains the logic to process requests and return responses.
* **Handles:**
  + User authentication and dashboard redirection based on role
  + Task creation and assignment (by Admin/Staff)
  + Task submission (by Students)
  + Status update (mark as done or marked)
  + Email notifications when a task is completed

**Template**

* HTML files styled with Bootstrap.
* Templates include: login.html, student\_dashboard.html, add\_task.html, admin\_dashboard.html, etc.
* Templates dynamically display data using Django template language ({{ }} and {% %}).

**2. Role-Based Workflow**

**Admin Workflow**

* Login using admin credentials.
* View dashboards for all users.
* Manage users (create, delete, or change roles).
* View all tasks and monitor overall system activity.

**Staff Workflow**

* Login and access staff dashboard.
* Create tasks and assign them to specific students.
* View submitted tasks.
* Mark tasks as reviewed or “marked”.

**Student Workflow**

* Login and access student dashboard.
* View assigned tasks and due dates.
* Upload files upon task completion.
* Submit and mark tasks as done.

**3. Email Notification Flow**

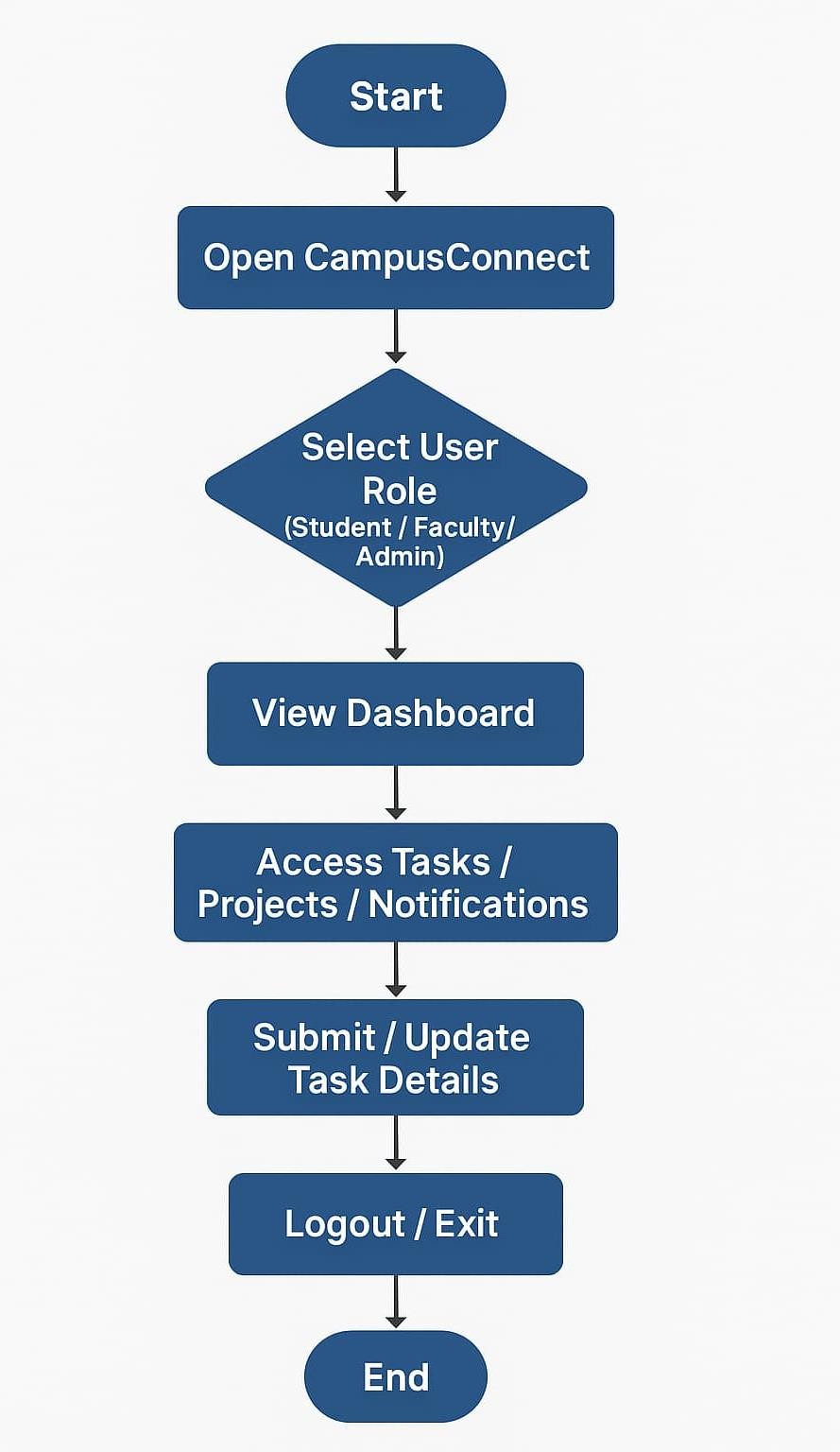
* Configured via Django's SMTP email backend.
* When a student submits a task:
  + An email notification is automatically sent to the assigned staff/admin.
* Ensures real-time communication and tracking.

**4. File Upload & Media Handling**

* Django handles file uploads through the FileField in the Task or Submission model.
* Uploaded files are stored in the /media/ directory.
* Files can be viewed/downloaded by staff/admin.

**5. Role-Based Functional Flow**

| **Role** | **Features** |
| --- | --- |
| Admin | Manage users, view all tasks, monitor overall progress |
| Staff | Create and assign tasks, view submissions, mark tasks as “Marked” |
| Student | View tasks, upload files, mark tasks as “Done” |



**Analytical Tools & Visualization**

**Analytical Tools & Libraries**

Tools and Libraries Used in CampusConnect

**1. Python**

* Core programming language used to build the backend logic of the project.
* Used for writing models, views, URL routing, and logic handling.

**2. Django**

* Main web framework used to develop the project.
* Provides built-in tools for:
  + MVT architecture (Model-View-Template)
  + User authentication
  + Admin panel
  + URL routing
  + Email sending (via SMTP)
  + File handling (uploads)

**3. SQLite**

* Default database used by Django for development.
* Stores data like users, tasks, roles, and submissions.

**4. HTML & Django Templates**

* Used for creating frontend UI pages like login.html, student\_dashboard.html, etc.
* Templates are dynamically rendered using Django Template Language ({{ }}, {% %}).

**5. Bootstrap (CSS Framework)**

* Used for styling the frontend pages.
* Ensures a clean, responsive, and user-friendly design.
* Features like:
  + Navigation bars
  + Tables
  + Buttons
  + Cards

**6. SMTP (Email Functionality)**

* You used Gmail’s SMTP to send email notifications when a student marks a task as done.
* Configured via Django settings (EMAIL\_HOST, EMAIL\_PORT, etc.).

**7. File Upload Handling**

* Used FileField in your Task model to let students upload files when submitting tasks.
* Media files handled using Django's MEDIA\_URL and MEDIA\_ROOT.

**Visualization**

**1. Task Status Overview**

* Pie chart or bar chart showing:
  + Completed vs Incomplete tasks
  + Marked vs Not Marked tasks

**2. Role-wise Task Distribution**

* Bar chart displaying how many tasks were assigned to each student/staff

**3. Monthly Task Trends**

* Line chart showing the number of tasks completed each month
* Helps identify student activity patterns

**4. Dashboard Summary (Textual/Visual)**

* Frontend dashboard contains:
  + Total tasks
  + Tasks completed
  + Tasks pending
  + User count (Admins, Staff, Students)

**Key Features & Benefits**

**1. Centralized Task Management**  
 All academic tasks are managed in one platform, reducing the confusion of scattered communication through emails or messages.

**2. Role-Based Access Control**  
 Students, staff, and admins have specific dashboards and permissions, ensuring a secure and organized user experience.

**3. User-Friendly Interface**  
 The use of HTML, CSS, and Bootstrap provides a clean, responsive, and intuitive UI that is easy to navigate for all users.

**4. Efficient Submission Handling**  
 Students can submit tasks quickly, and staff can review them without the need for manual paperwork or external systems.

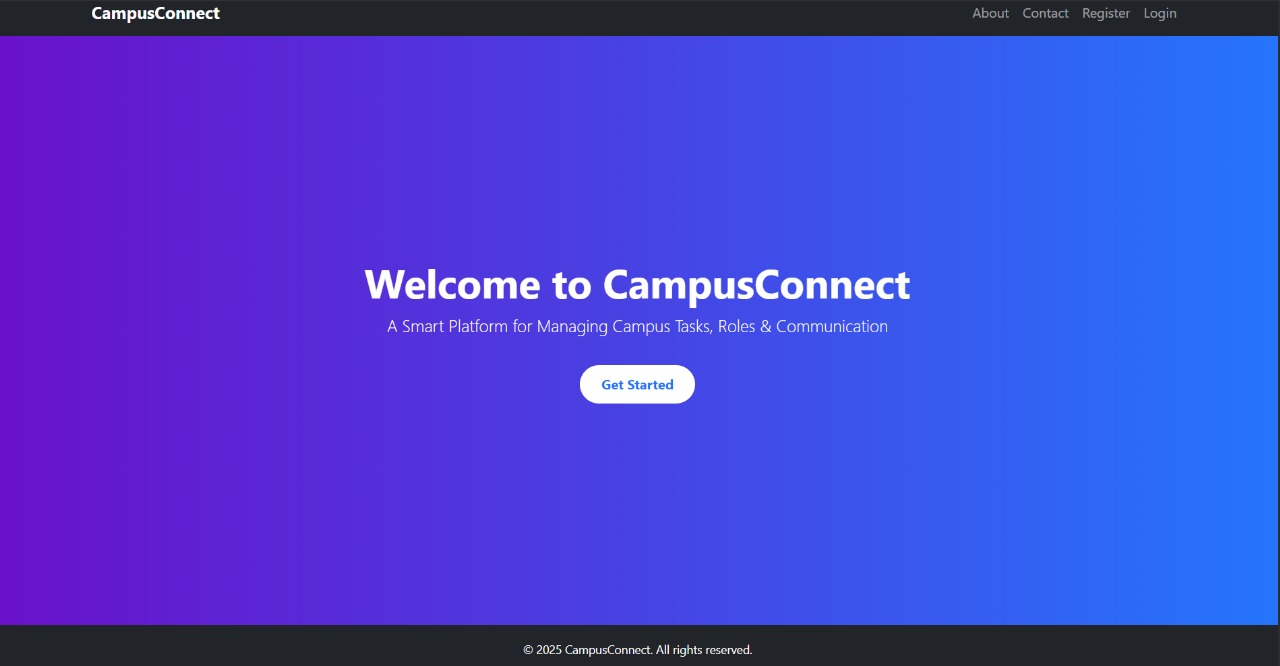
**5. Database-Backed Storage**  
 All data is securely stored using Django ORM and SQLite, ensuring reliability, scalability, and ease of maintenance.

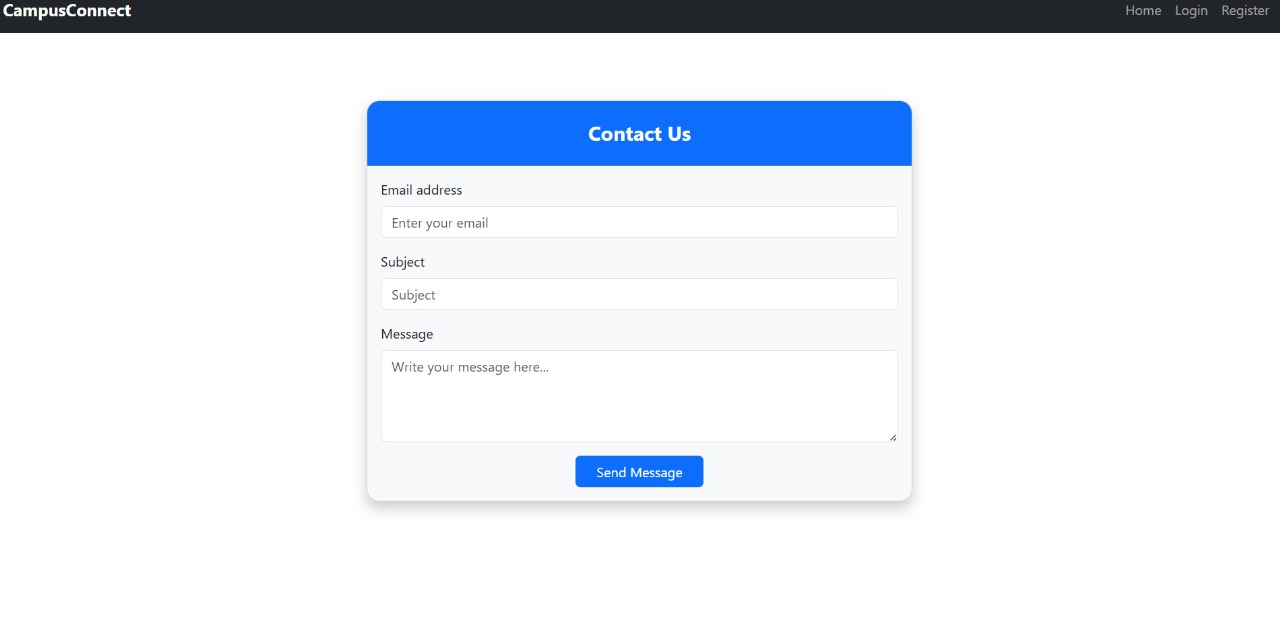
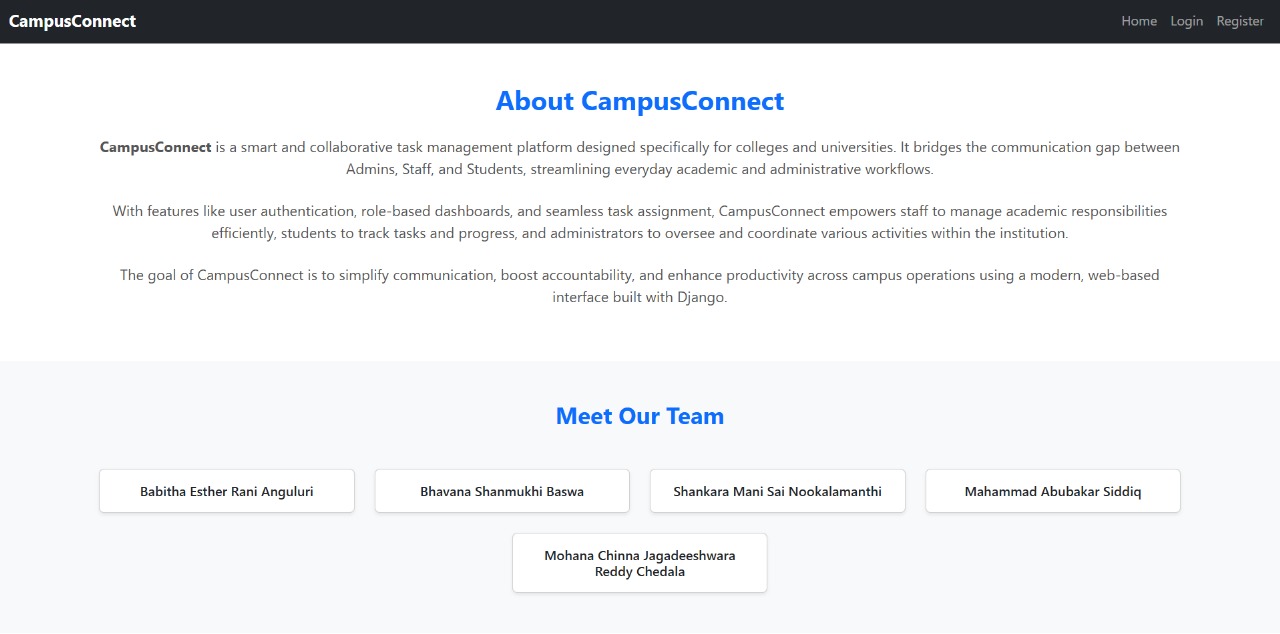
**6. Transparent Workflow**  
 The system provides clear tracking of task status (submitted/pending), enhancing accountability and academic discipline.

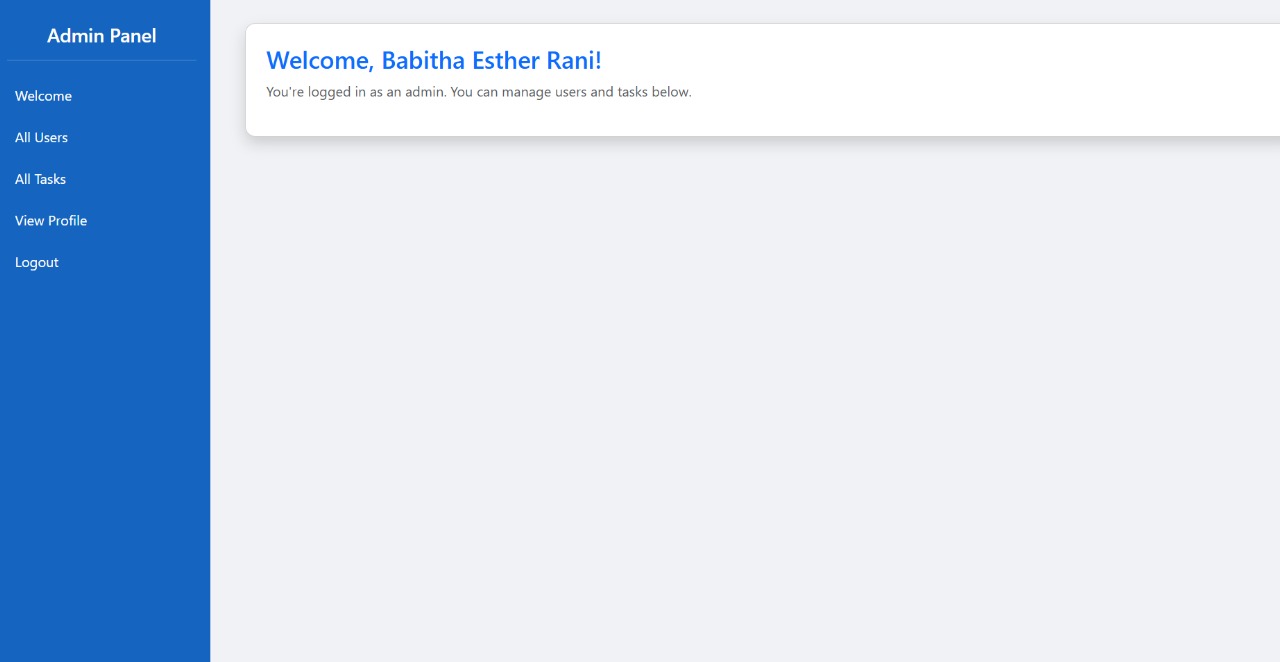
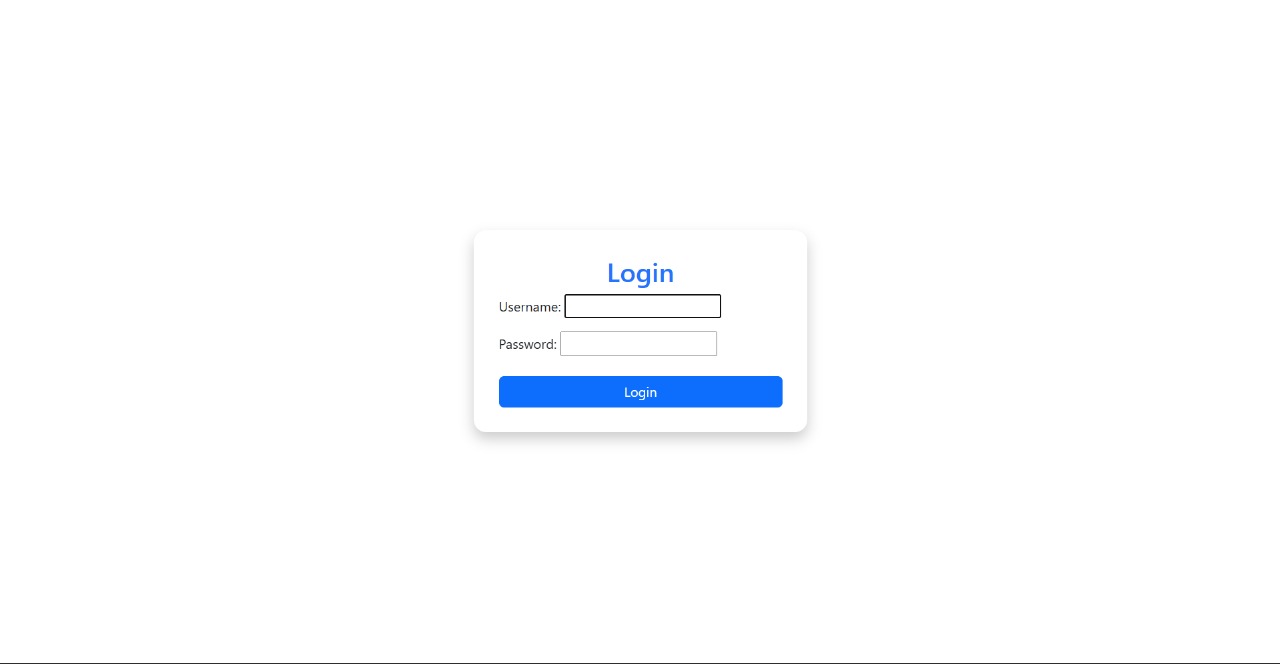
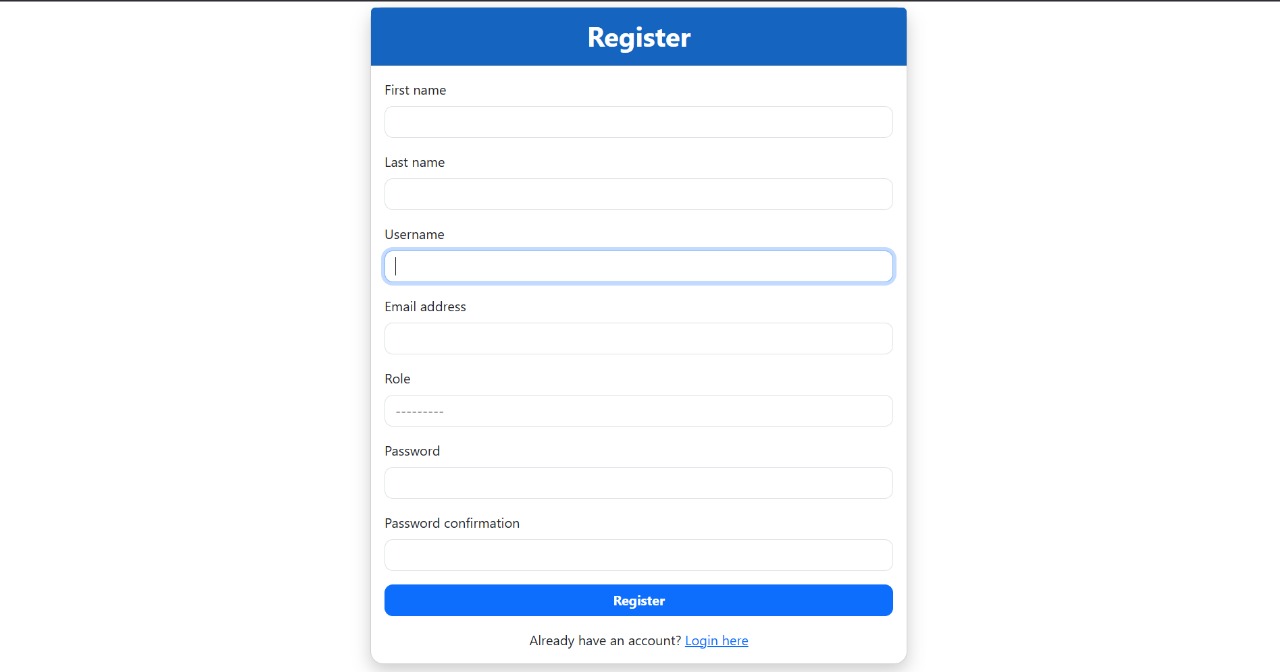
**7. Reduces Manual Workload**  
 Automates the process of task distribution, collection, and tracking — saving time for both staff and students.

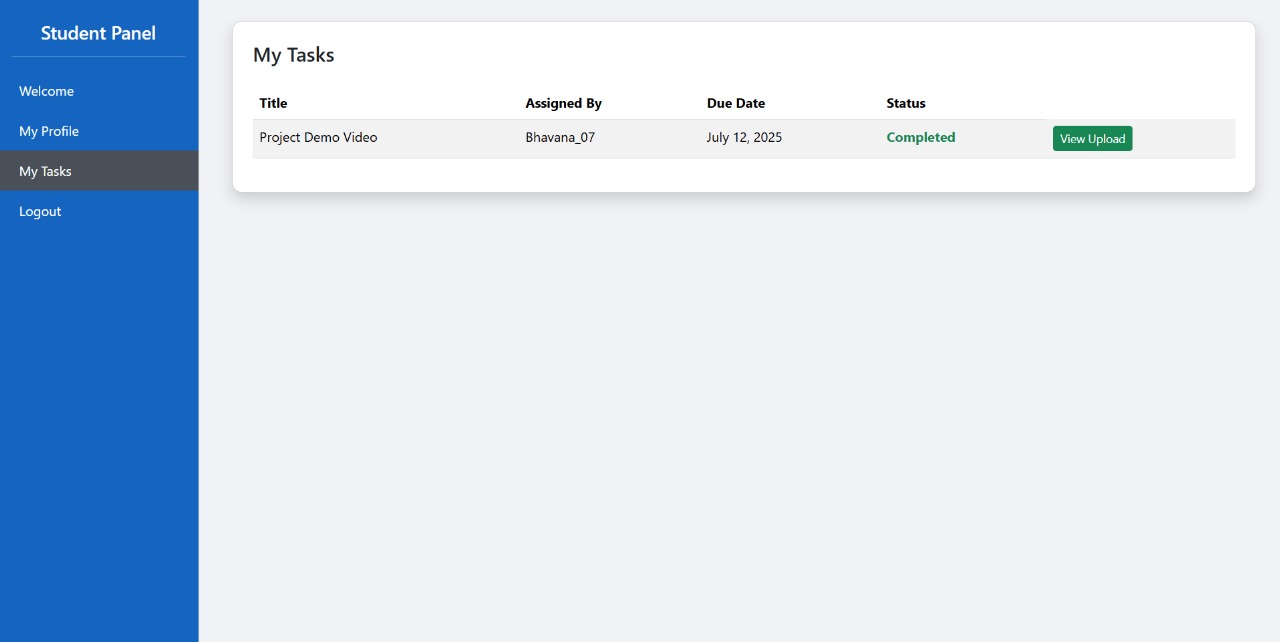
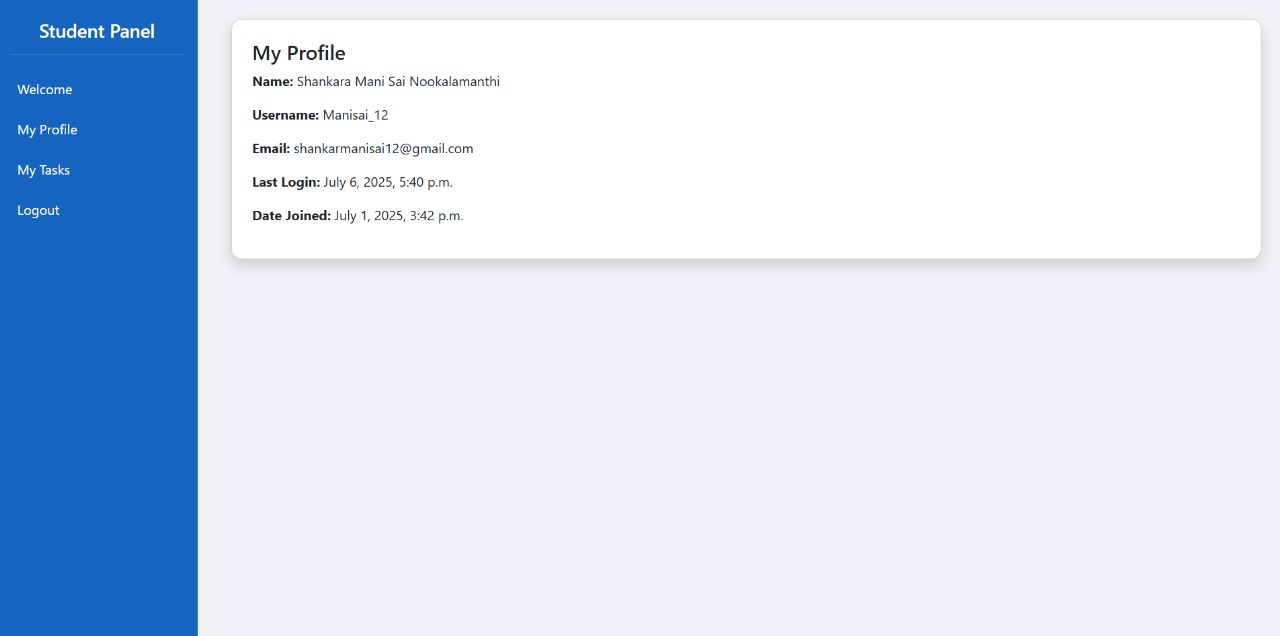
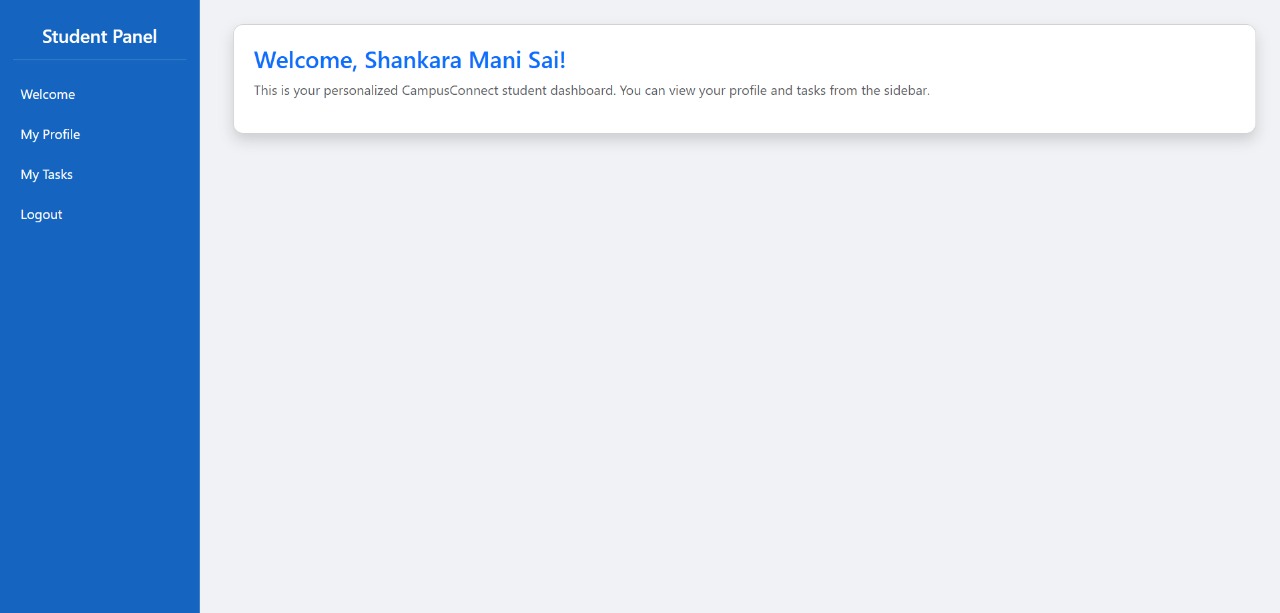
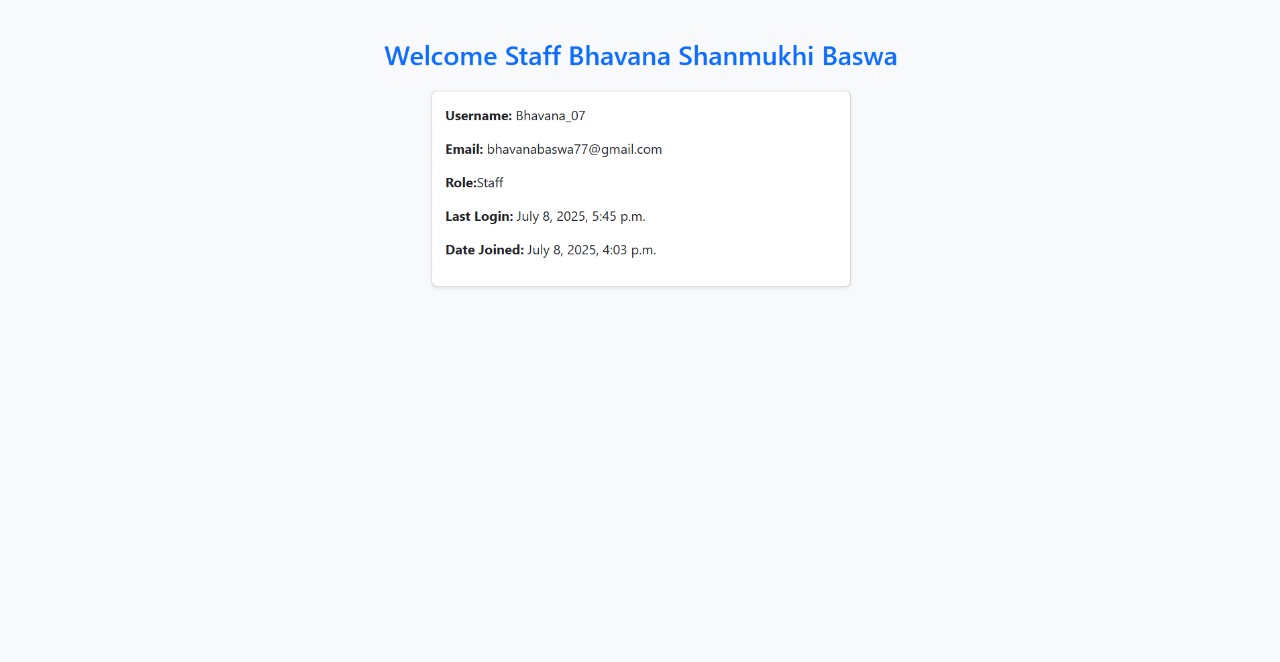
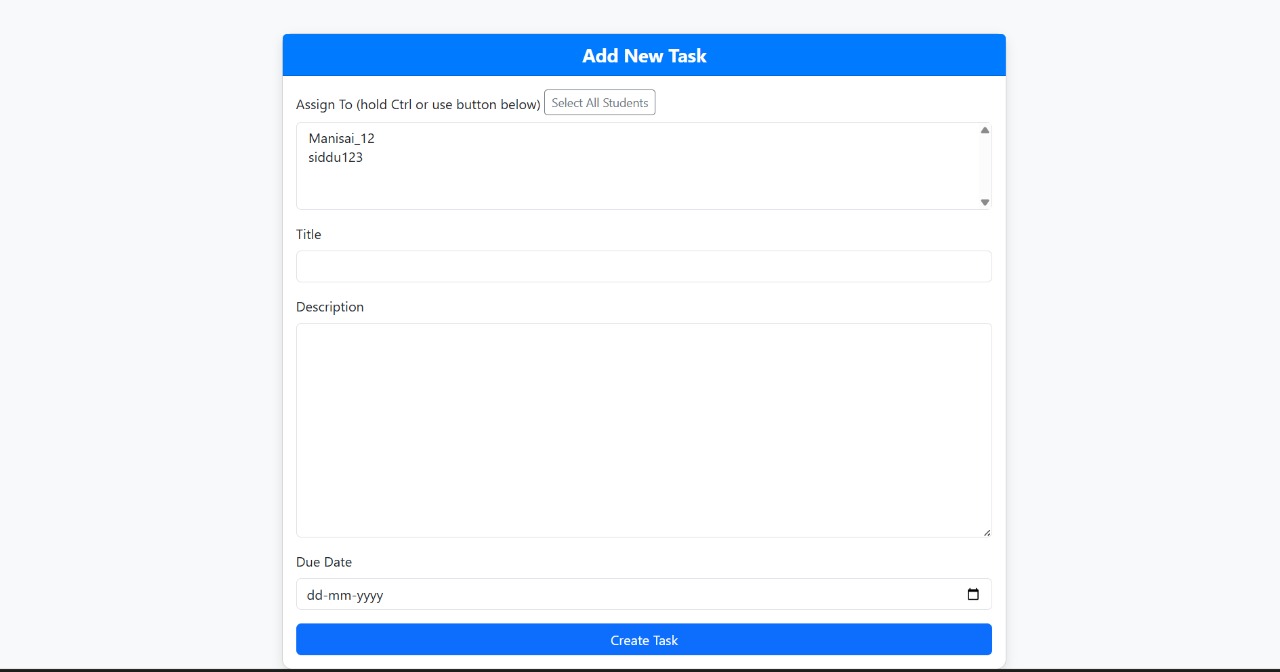
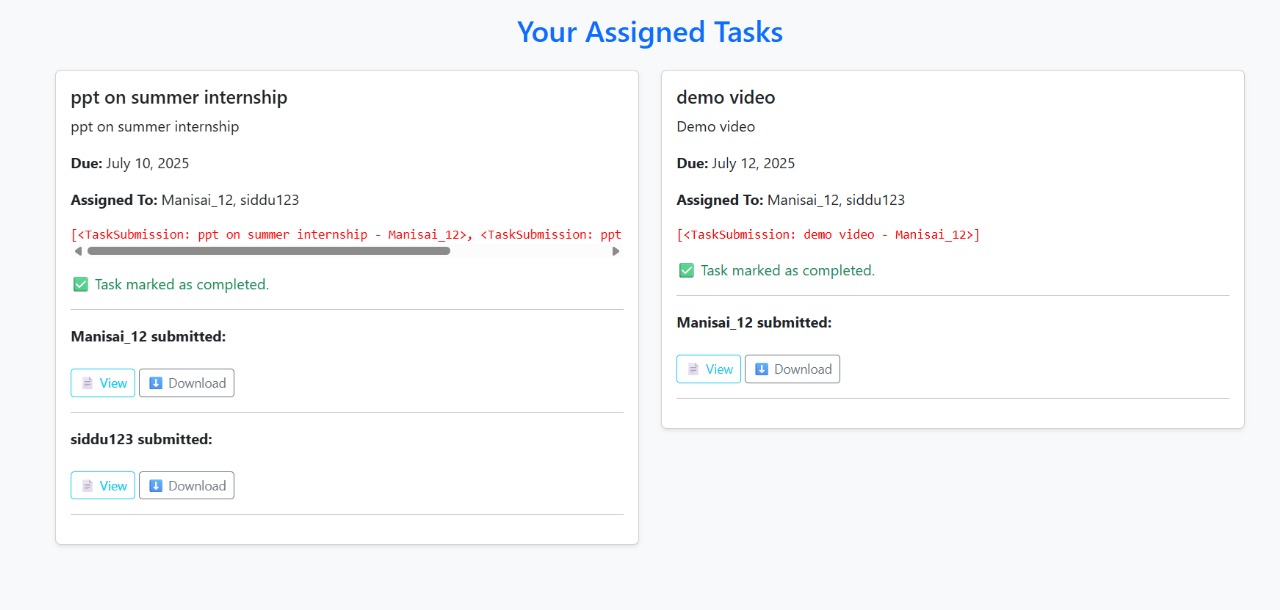
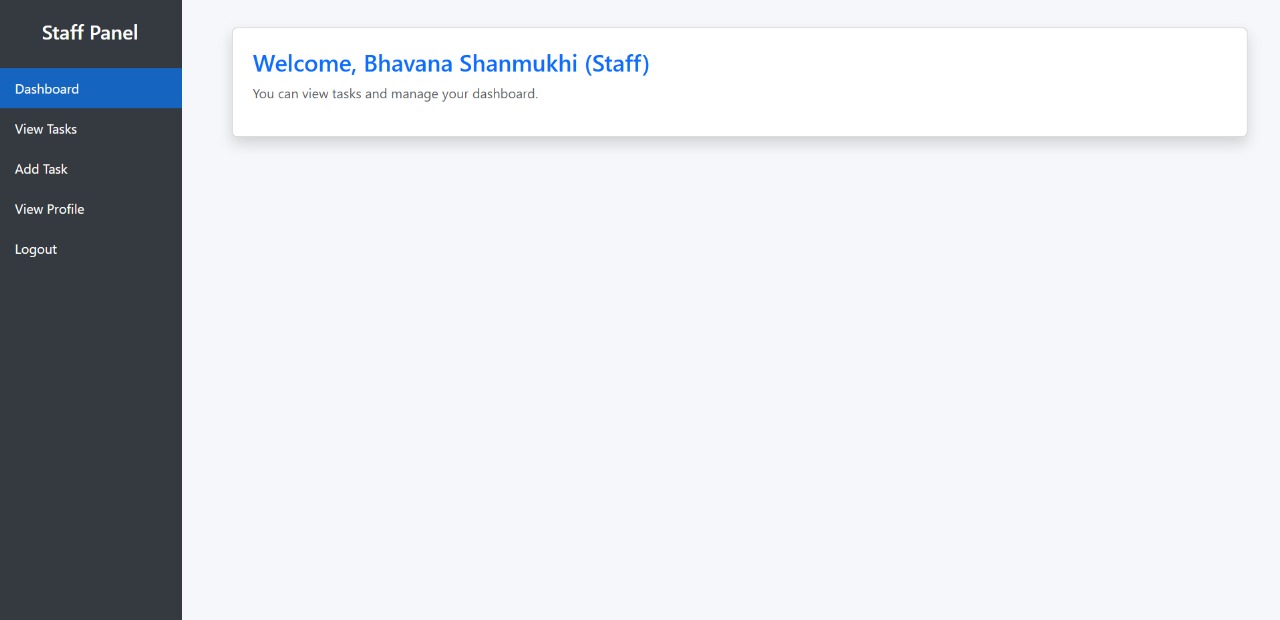
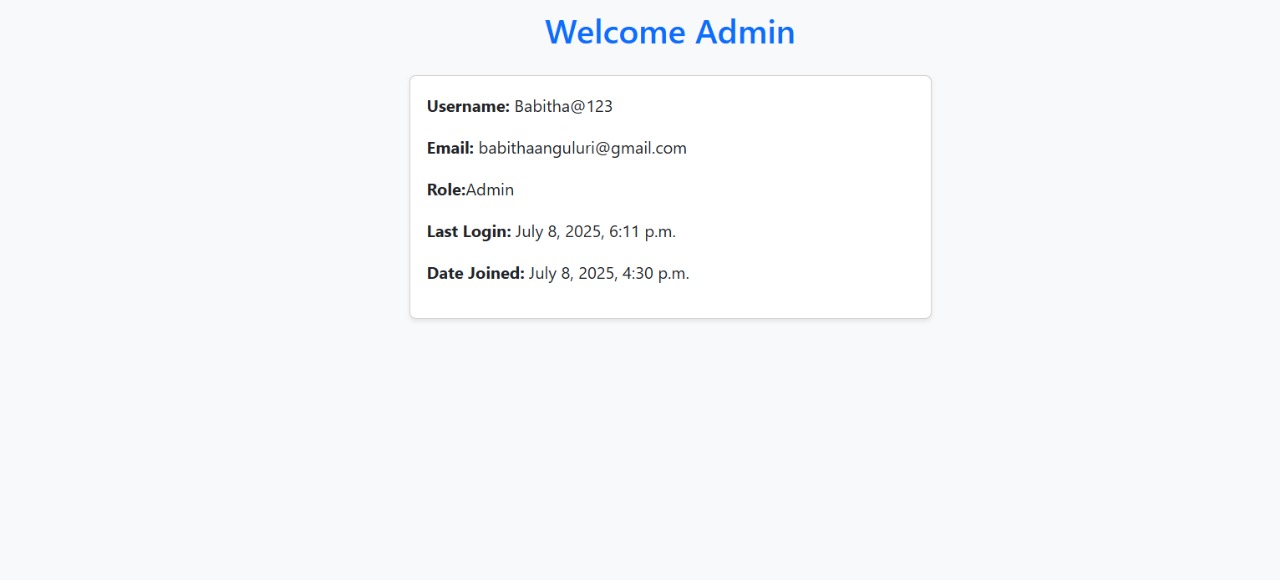
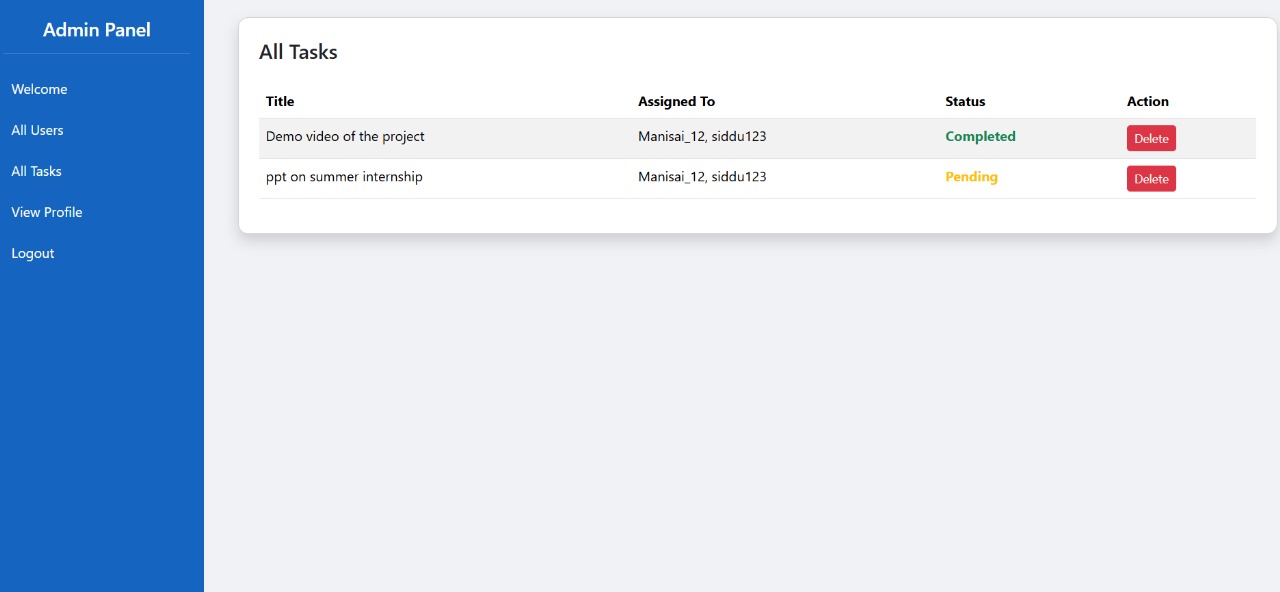
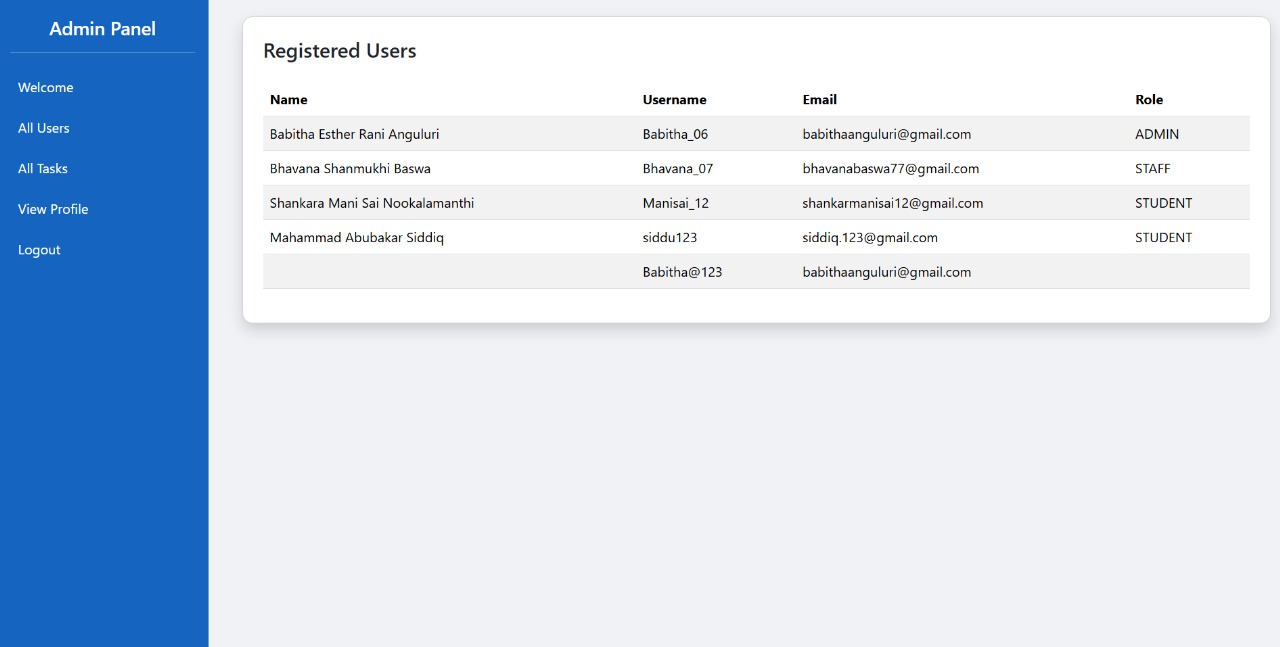
**8. Open-Source & Cost-Effective**  
 Built entirely with open-source technologies like Django, Python, and Bootstrap, making it free to use and maintain.

**Project Screenshots / Output Screens**









**CONCLUSION**

This project successfully developed a web-based academic task management system — Campus Connect — using the Django framework and open-source technologies like Python, HTML, CSS, and Bootstrap. By implementing features such as role-based dashboards, secure user authentication, and centralized task handling, the system streamlines how students submit tasks and how staff monitor them.

Through efficient task distribution and submission tracking, the platform addresses key issues in academic environments, such as miscommunication, deadline management, and workload organization. The system promotes transparency by displaying submission statuses and storing data securely using Django ORM and SQLite. These functionalities help both students and staff stay accountable and organized throughout the academic process.

Campus Connect enhances digital learning by reducing dependency on physical paperwork and manual tracking methods. Its scalable, modular design allows for future improvements like grading, deadline alerts, and analytics dashboards. The project demonstrates how a simple, well-structured digital tool can significantly improve academic productivity and communication within institutions.

By leveraging open-source tools and a clean architectural approach, this project not only supports educational efficiency but also offers a replicable, cost-effective solution for schools, colleges, and universities. With further development, Campus Connect has the potential to evolve into a complete academic management system tailored to the needs of modern education.

**Reference Materials & Learning Resources**

1. Django Project Documentation (v4.2–2024) — Offers comprehensive guidance on building web applications with Django, including models, views, URL routing, authentication, and admin configurations; extensively used throughout the backend development of the Campus Connect platform.  
 [https://docs.djangoproject.com](https://docs.djangoproject.com/)

2. Bootstrap 5 Documentation — Provided detailed resources for implementing responsive design elements such as grids, cards, forms, and buttons; helped enhance the user experience across student, staff, and admin dashboards.  
 [https://getbootstrap.com](https://getbootstrap.com/)

3. GeeksforGeeks – Django & Python Web Tutorials — Served as a major reference for implementing role-based authentication, form handling, file uploads, and user session management in Django views and templates.  
 <https://www.geeksforgeeks.org/django-tutorial/>

4. W3Schools – Front-End Development — Used for structuring and styling the interface with HTML5 and CSS3; also supported template development and layout design using Django templating language.  
 <https://www.w3schools.com/>

5. Stack Overflow – Technical Issue Resolution — Helped address project-specific challenges such as form validation errors, static file loading, redirect issues, and Django URL configuration problems through real-world developer solutions.  
 <https://stackoverflow.com/>

6. SQLite and Django ORM Documentation — Consulted to design models, establish foreign key relationships, and manage data storage and retrieval securely within the system.  
 <https://www.sqlite.org/docs.html>

7. YouTube – Django Full Stack Projects — Practical video tutorials guided the step-by-step implementation of modules like user login, task submission, and dashboard routing.  
 [https://www.youtube.com/+3](https://www.youtube.com/@Telusko)

8. GitHub – Open-Source Django Repositories — Explored to understand standard project structures, database modeling strategies, and submission handling workflows in task management systems.  
 <https://github.com/>

9. Python Official Documentation — Referenced for implementing custom logic in views, file operations, and secure handling of user inputs within Django’s backend.  
 <https://docs.python.org/3/>