My Details

Name: Ved Uttam Tamhanekar

Roll no : 1024031039

Project Link: https://github.com/Ved-1111/Battery-Swap-Appointment-Booking-System

Web Development Project Report

Project Title

Battery Swap Booking System Using Flask and MySQL

1. Introduction

The purpose of this project is to develop a web application that allows users to book battery swap appointments at designated stations. The system includes user authentication, appointment scheduling, and an admin panel for managing appointments. The backend is built with Flask (Python), and MySQL is used for database management.

2. Objective

- To develop a responsive web application for battery swap bookings.

- To implement user authentication (login, registration) and role-based access (user/admin).

- To enable users to book, view, and manage appointments.

- To provide admins with tools to update appointment statuses (e.g., booked, completed, cancelled).

3. Tools and Technologies Used

- **Programming Languages:** Python (Flask), HTML, CSS

- **Frameworks:** Flask (Backend)

- **Database:** MySQL

- **IDE:** Visual Studio Code

4. System Requirements

- Web Browser (Chrome, Firefox)

- Python 3.x and Flask installed

- MySQL Server for database management

5. Project Modules / Features

- **User Authentication:** Login, Registration, Logout

- **Appointment Booking:** Users can book appointments by selecting a station, date, and time.

- **Appointment Management:** Users can view their appointments.

- **Admin Panel:** Admins can view all appointments and update their statuses (e.g., booked, completed, cancelled).

- **Responsive UI:** Clean and user-friendly interface for both desktop and mobile devices.

6. Project Flow / Working

1. **User Registration/Login:** Users register or log in to access the booking system.

2. **Booking an Appointment:** Users select a station, date, and time to book an appointment.

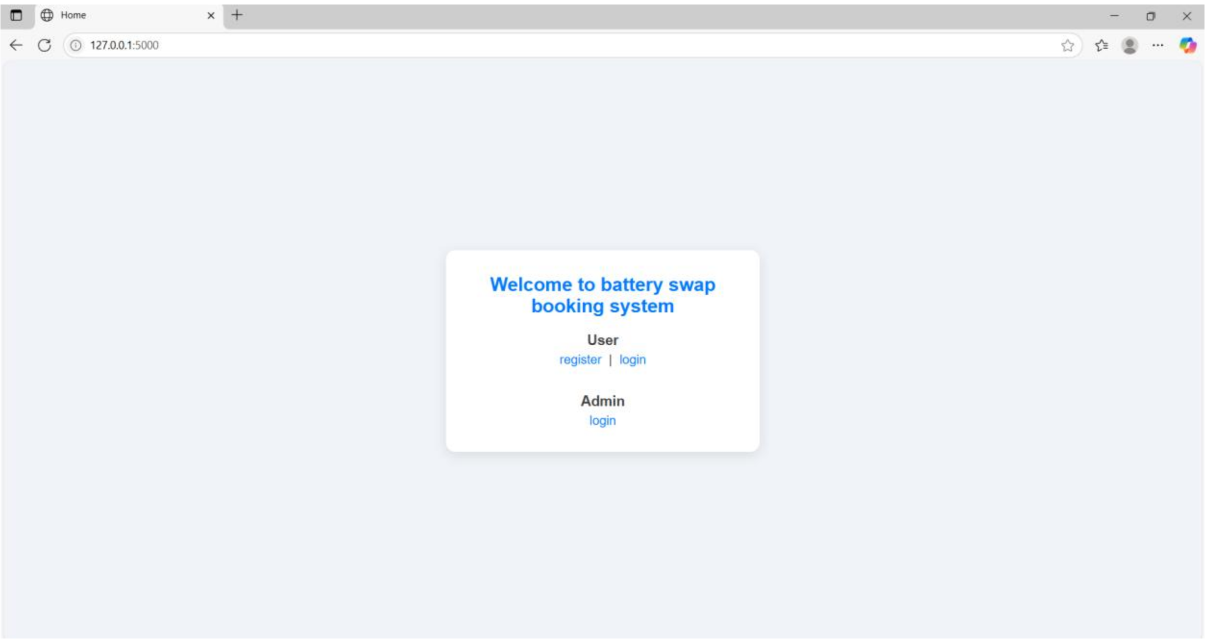
3. **Appointment Management:** Users view their appointments on the "My Appointments" page.

4. **Admin Access:** Admins log in to view all appointments and update their statuses.

5. **Real-Time Updates:** Changes in appointment statuses are reflected immediately in the database and UI.

7. Output Screenshots

- Home Page



Welcome to battery swap booking system

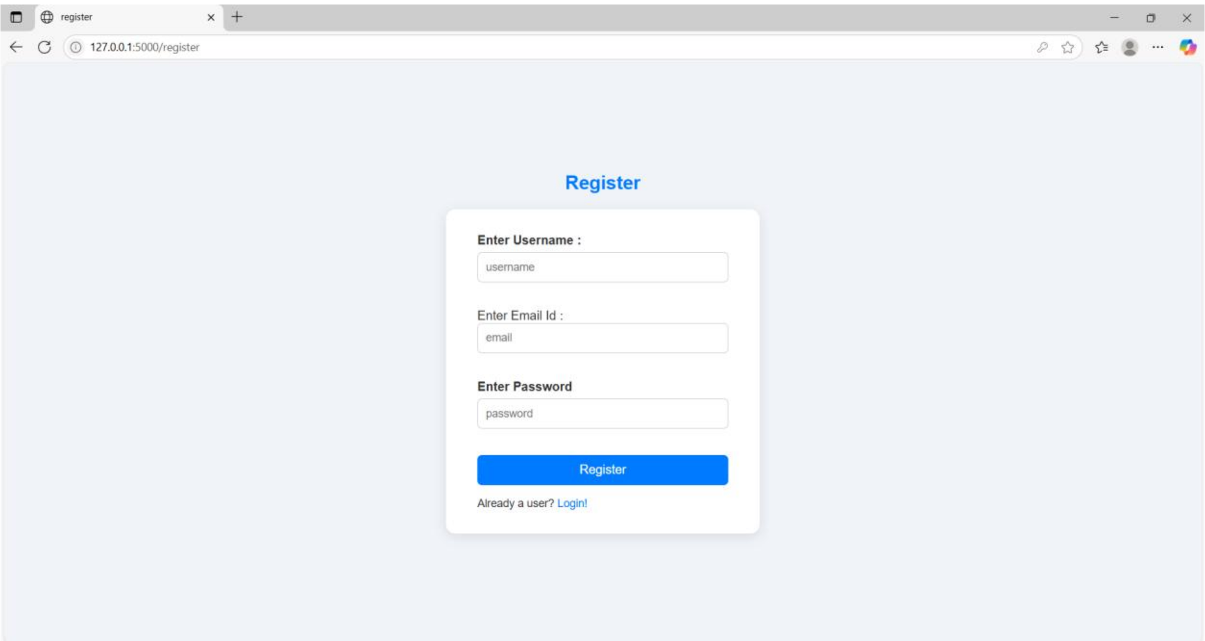
User

register | login

Admin

login

- Login/Registration Pages



Register

Enter Username :

username

Enter Email Id :

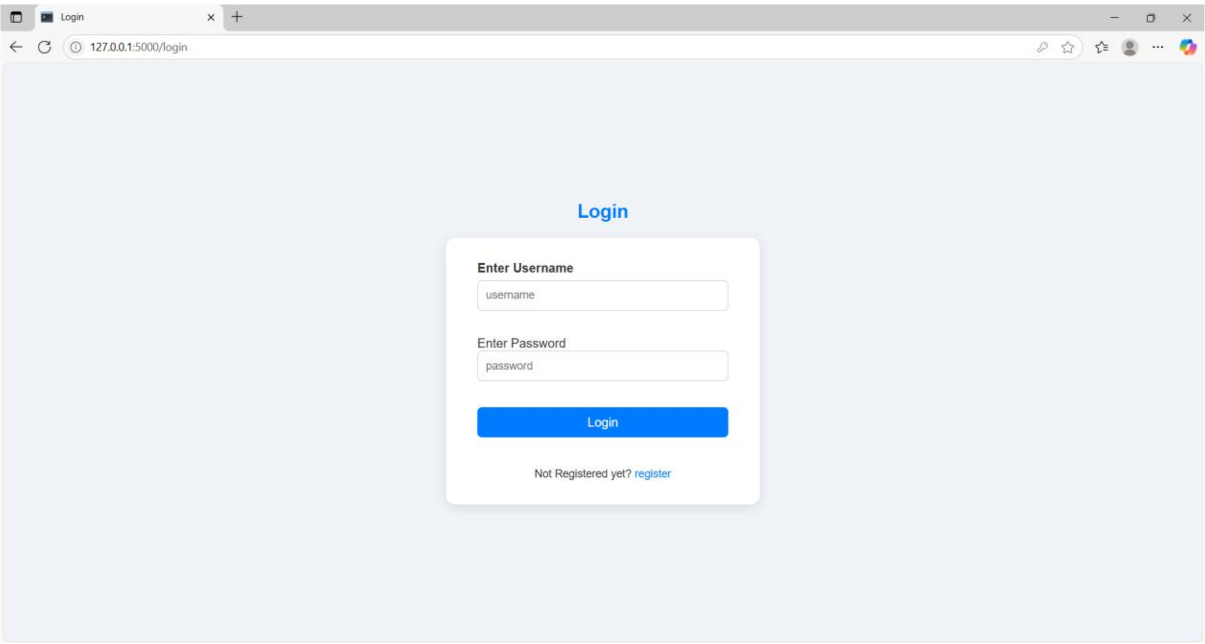
email

Enter Password

password

Register

Already a user? Login!



Login

Enter Username

username

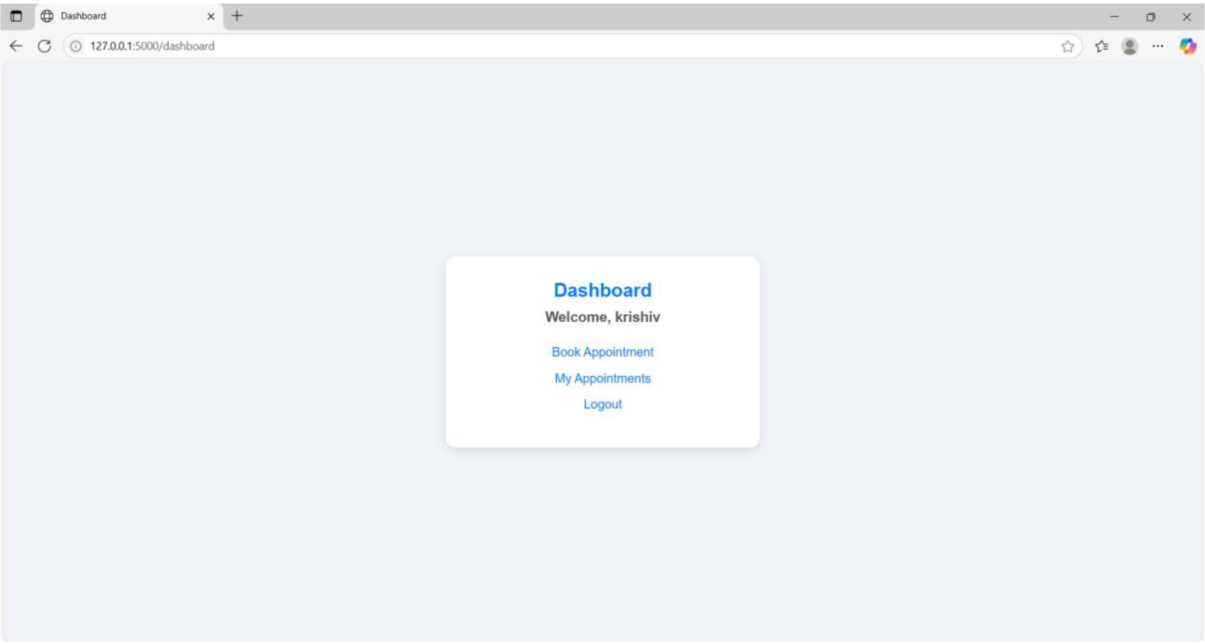
Enter Password

password

Login

Not Registered yet? register

- User Dashboard Page



Dashboard

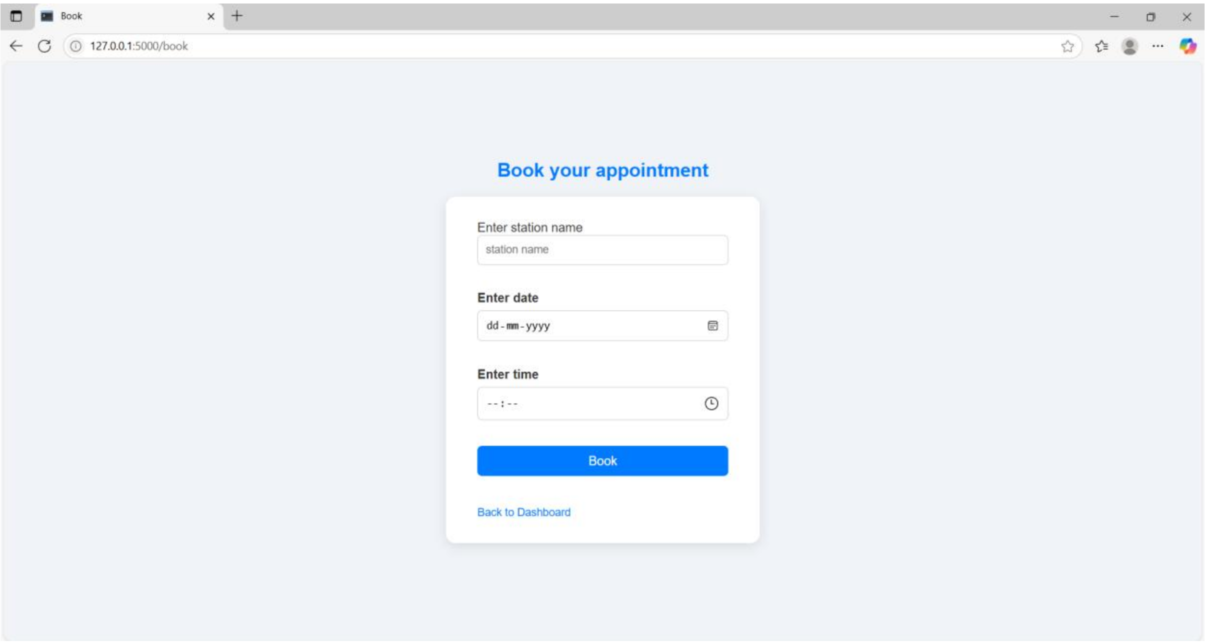
Welcome, krishiv

Book Appointment

My Appointments

Logout

- Booking Page



Book your appointment

Enter station name

station name

Enter date

dd-mm-yyyy

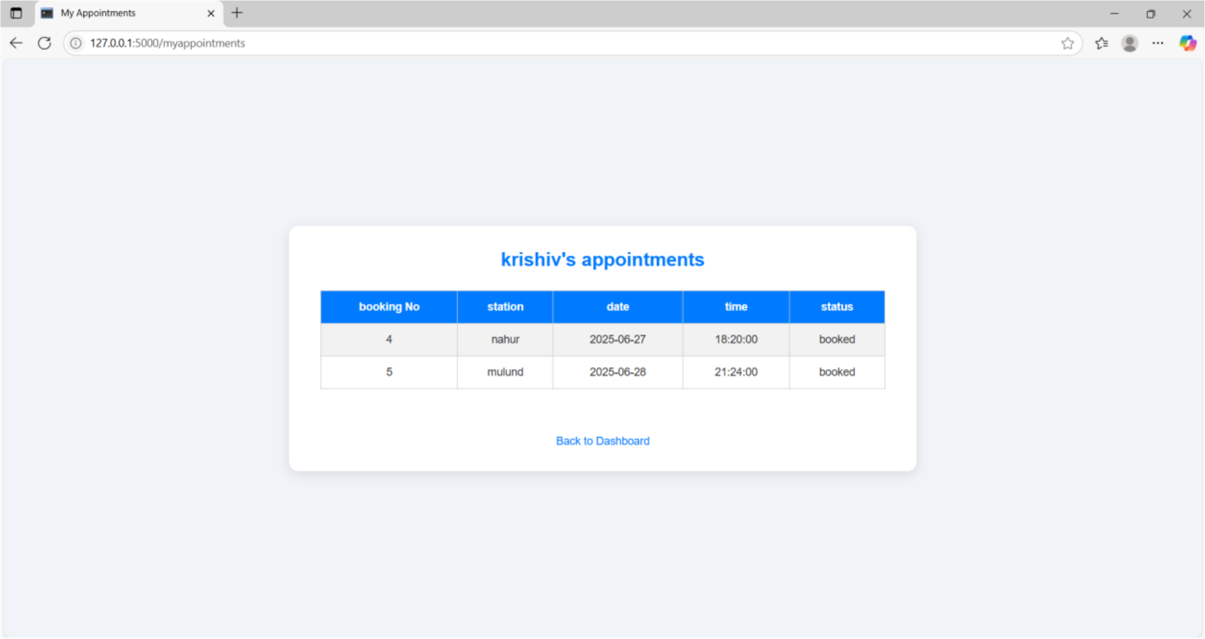
Enter time

--:--

Book

Back to Dashboard

- My Appointments Page

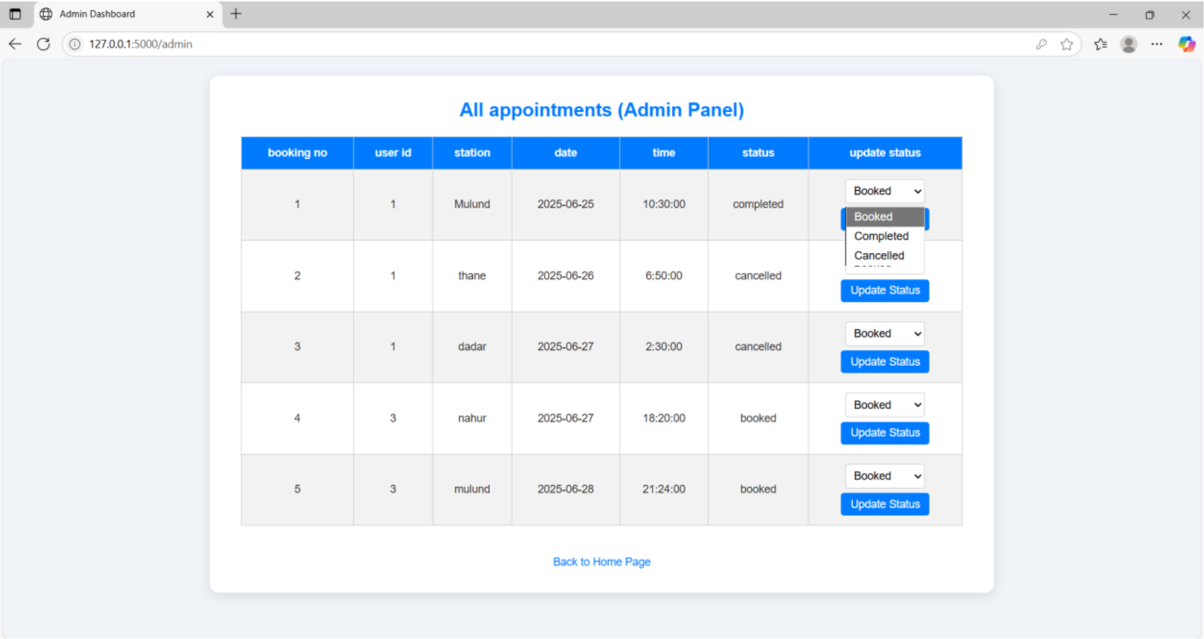


krishiv's appointments

booking No	station	date	time	status
4	nahur	2025-06-27	18:20:00	booked
5	mulund	2025-06-28	21:24:00	booked

Back to Dashboard

- Admin Panel



All appointments (Admin Panel)

booking no	user id	station	date	time	status	update status
1	1	Mulund	2025-06-25	10:30:00	completed	Booked Booked Completed Cancelled
2	1	thane	2025-06-26	6:50:00	cancelled	Update Status
3	1	dadar	2025-06-27	2:30:00	cancelled	Booked Update Status
4	3	nahur	2025-06-27	18:20:00	booked	Booked Update Status
5	3	mulund	2025-06-28	21:24:00	booked	Booked Update Status

Back to Home Page

8. Testing and Evaluation

- **Manual Testing:**
  - Verified user authentication (login, registration).
  - Tested appointment booking and status updates.
  - Validated admin functionalities (viewing and updating appointments).
- **Cross-Browser Testing:** Ensured compatibility with Chrome and Firefox.
- **Responsiveness:** Confirmed the UI adapts to different screen sizes.

9. Conclusion

This project enhanced my skills in full-stack web development, including Flask backend development, MySQL database integration, and frontend design using HTML/CSS. I learned to implement role-based access control, handle form submissions, and manage real-time data updates.

10. References

- Flask Documentation

- MySQL Documentation

- MDN Web Docs (HTML/CSS/JavaScript)

- StackOverflow