App Dev 1 Project Report

Student Details

Name: **Golu kumar**
Roll Number: 23f2003761
Email: 23f2003761@ds.study.iitm.ac.in
About Me: Diploma level student at IIT Madras BS in Data Science and Application, with a focus on full-stack development and building robust, data-driven web applications.

Project Details

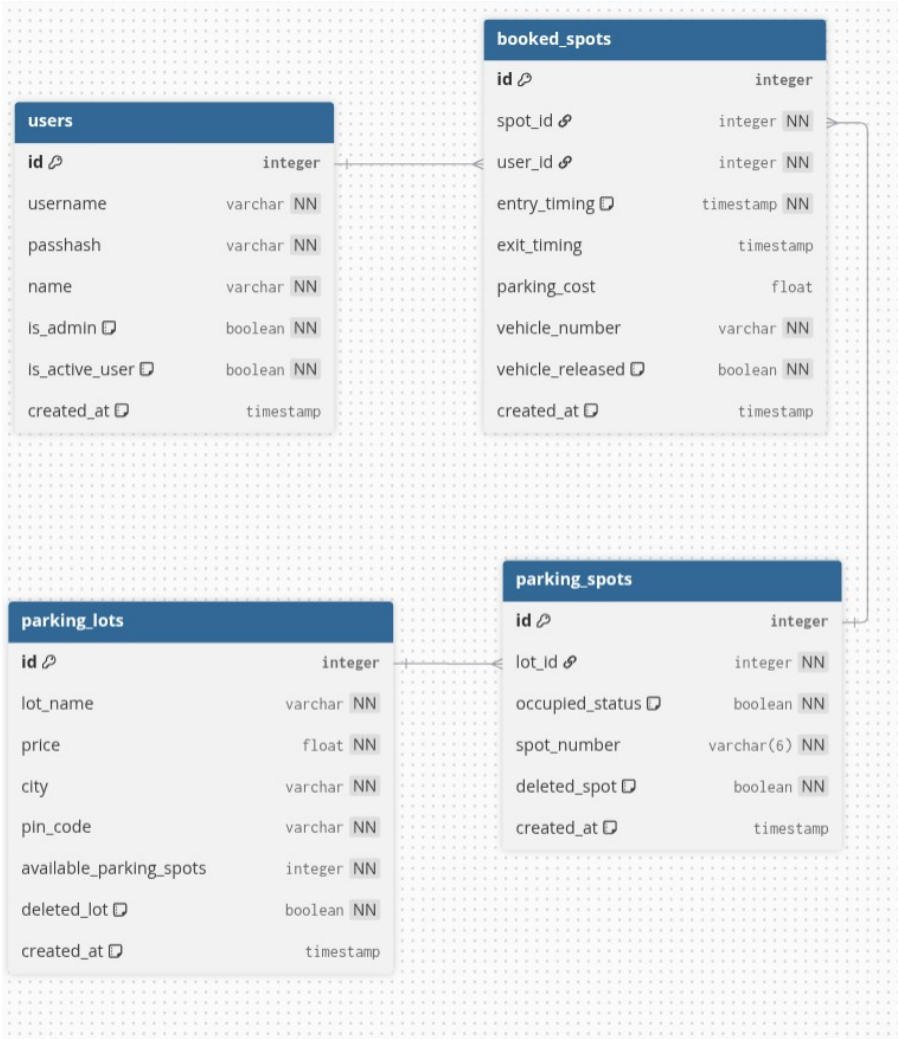
Project Title: Parking Lot Booking System
Problem Statement: To design a two-sided (User/Admin) web application for real-time tracking, booking, and management of vehicle parking spots, eliminating user uncertainty and providing administrators with key utilization and revenue data.
Approach: Built on the Flask micro-framework using an MVC-like structure. It leverages SQLAlchemy for data persistence and implements secure role-based access. Parking costs are automatically calculated upon release, rounded up to the nearest hour for billing.

Technologies and Frameworks Used

- **Flask:** Core backend web framework.
- **Flask-SQLAlchemy:** Object Relational Mapper for database.
- **SQLite:** Lightweight local database.
- **Jinja2 / Bootstrap 5:** Template rendering and responsive styling.
- **Chart.js:** Visualization for analytical dashboards.
- **Werkzeug Security:** Password hashing and verification.
- **Flask Session:** User authentication and session management.
- **Python `math`:** Used for `ceil()` function for billing.

Database Schema / Key Tables & Relationships / ER Diagram

- **User** (id, username, passhash): Stores credentials and role (is_admin, is_active_user).
- **ParkingLot** (id, lot_name, price): Stores location, pricing, and overall spot count.
- **ParkingSpot** (id, lot_id, spot_number): Represents individual spots and their occupancy status.
- **Bookedspot** (id, user_id, spot_id, entry_timing, exit_timing): Stores transactional booking history and calculated parking_cost.
- **Relationships:** One-to-Many connections between User to Bookedspot and ParkingLot to ParkingSpot.



Useful Routes/ Endpoints

Endpoint/Routes	Method	Role	Description
/login, /register	POST	Public	User authentication and account creation.
/book_this_spot/<int:spot_id>	POST	User	Finalizes booking, marks spot as occupied, records entry_timing.
/release_spot/<int:book_id>/<int:spot_id>	GET	User	Completes transaction, calculates cost (rounded up hours), and updates spot status.
/create_lot	POST	Admin	Adds a new lot and bulk-creates associated ParkingSpot records.
/deactivate_this_spot/<int:sid>	POST	Admin	Toggles the availability (deleted_spot) of a parking spot.
/admin_summary	GET	Admin	Retrieves data for parking utilization (pie chart) and monthly revenue (bar chart).
/user_bookings_summary	GET	User	Retrieves data for user's monthly booking counts and total expenditure visualization.

Architecture and Features

Implemented Features:

1. **Dual Role Access:** Enforced access control for Standard Users and Administrators via custom Flask decorators.
2. **Secure Authentication:** User registration with password hashing (Werkzeug), session management, and admin capability to block user accounts.
3. **Lot & Spot Management (Admin):** Comprehensive CRUD for lots, including dynamic creation/recreation of spots and individual spot status toggling (deleted_spot).
4. **Dynamic Booking:** Real-time spot availability checking, prevention of double-booking, and instant occupancy status updates.
5. **Billing Automation:** Accurate parking cost calculation on vehicle release, rounding up duration to the nearest whole hour (math.ceil).
6. **Analytical Dashboards:** Chart.js visualizations provide Admins with utilization/revenue trends and Users with expenditure/booking frequency summaries.

Video Presentation

Drive Link
video link