Author

Adwait Keshari
Roll No. 23f2000979
Student Email id: 23f2000979@ds.study.iitm.ac.in
Hi my name is Adwait Keshari, I am from Bhopal Madhya Pradesh and currently I am in my diploma of this IITM BS degree program.

Description

This is a vehicle parking management web application built with Flask that allows users to book and release parking spots while providing administrators with tools to manage parking lots, view user data, and generate revenue reports.

Summary of AI/LLM used percentage: In this application I have used nearly 15-20% of AI for making the complete project.

Technologies used

Flask: Core web framework for building the application with routing, templating, and session management
Flask-SQLAlchemy : Database ORM for Python that simplifies database operations and provides model relationships
SQLite: Lightweight, serverless database for storing user data, parking lots, spots, and reservations
HTML/CSS: Frontend technologies for creating responsive user interfaces and interactive dashboards
Bootstrap: CSS framework for styling and responsive design components

Purpose: Flask was chosen for its simplicity and flexibility in building web applications quickly. SQLite provides a lightweight database solution perfect for this scale of application. Flask-SQLAlchemy abstracts complex database operations into simple Python objects, making the code more maintainable.

DB Schema Design

1. Users Table
id: Integer, Primary Key
email_id: String(120), Unique, Not Null
password: String(120), Not Null
full_name: String(120), Not Null
address: String(200), Not Null
pin_code: String(10), Not Null
role: String(20), Not Null, Default = "user"

2. Parking Lots Table
id: Integer, Primary Key
prime_location_name: String(120), Unique, Not Null
price_per_hour: Float, Not Null
address: String(200), Not Null
pin_code: String(10), Not Null
maximum_number_of_spots: Integer, Not Null

3. Parking Spots Table
id: Integer, Primary Key
lot_id: Integer, Foreign Key → Parking Lots(id), Not Null
spot_number: Integer, Not Null
status: String(1), Not Null, Default = "A" (A = Available, O = Occupied)
Constraint: Unique(lot_id, spot_number)

4. Reserved Spots Table
id: Integer, Primary Key
spot_id: Integer, Foreign Key → Parking Spots(id), Not Null
user_id: Integer, Foreign Key → Users(id), Not Null
vehicle_number: String(20), Not Null
parking_timestamp: DateTime, Not Null, Default = now
leaving_timestamp: DateTime, Nullable
total_cost: Float, Nullable

The schema follows a hierarchical structure where parking lots contain multiple spots, and spots can have multiple reservations over time. The unique constraints prevent data integrity issues, while nullable fields allow for flexible booking states. Foreign keys maintain referential integrity, and the status field enables real-time availability tracking.

API Design

The application implements a RESTful web API using Flask routes that handle user authentication, parking lot management, spot booking/release, and data retrieval. Key API endpoints include user registration/login, admin dashboard operations, parking spot booking with vehicle registration, cost calculation upon spot release, and comprehensive reporting for administrators. The API uses session-based authentication, form data processing, and JSON responses for health checks. All endpoints follow consistent error handling patterns with flash messages for user feedback.

Architecture and Features

The application follows a simple MVC-like pattern where Flask routes in app.py act as controllers handling HTTP requests and responses. Database models are separated into model.py for clean code organization. HTML templates are stored in the templates/

directory with a base template providing consistent layout. Static assets like CSS are in the static/ folder. The application uses SQLite database stored in the instance/ directory.

The core features include user registration and authentication with role-based access control (user/admin), real-time parking spot availability tracking with search functionality, dynamic parking lot management for administrators including CRUD operations, automated cost calculation based on parking duration, comprehensive user booking history with detailed reports, and an admin dashboard with revenue analytics and occupancy statistics. Additional features include responsive web design using Bootstrap, session management for secure user experience, and data validation to prevent booking conflicts and ensure data integrity. The system automatically creates parking spots when lots are added and prevents deletion of lots with occupied spots.

Video

https://drive.google.com/file/d/15Js2RCeDubEAwk9RL9loCCCq80jQ4JMD/view?usp=sharing