Documentation for Vehicle Parking Management System

Author

Name: Krishna Kumar B J
Roll Number: 22F3001224
Email: 22f3001224@ds.study.iitm.ac.in

About Me:
I am currently pursuing a degree in Information Technology at Sri Sairam Institute of Technology. I am passionate about full-stack development, backend systems, and working with databases and APIs. This project reflects my interest in building real-world applications using Flask and SQLAlchemy.

Description: This project is a web-based Parking Management System where users can register, book parking spots, and view their booking details. Admins can manage parking lots, monitor spot occupancy, and generate visualizations. The system ensures data integrity and real-time status updates for parking availability.

Technologies Used

- **Python** – Core backend logic
- **Flask** – Web framework
- **Flask-Login** – For authentication and user session management
- **Flask-WTF** – For form handling
- **Flask-Bootstrap / Bootstrap 5** – For responsive frontend design
- **SQLAlchemy** – ORM for database models and queries
- **SQLite** – Lightweight database for local development
- **Matplotlib** – For admin dashboard visualizations
- **Jinja2** – For rendering dynamic templates

DB Schema Design

users: id, username, email, password, role

parking_lots: id, prime_location_name, address, pin_code, price, max_spots, available_slots

parking_spots: id, lot_id (FK), is_occupied

reservations: id, user_id (FK), spot_id (FK), vehicle_number, booking_

API Design

The project includes internal routes for admin and user functionalities. APIs are implemented using Flask routes.

APIs for booking, viewing lots, searching users/vehicles, and admin controls.

Implemented using Flask routes.

RESTFUL API

Architecture and Features

The project follows a modular **MVC-style architecture**:

- **Controllers**: Flask Blueprints (admin_bp, user_bp) handle route logic.
- **Models**: Defined in models.py using SQLAlchemy ORM.
- **Templates**: Stored under templates/ directory using Jinja2.
- **Static Files**: Bootstrap and custom CSS under static/.
- **Database**: parking.db using SQLite.

Features Implemented:

- User registration and login
- Admin and user role-based access
- Parking lot creation, editing (only if slots are unoccupied), and deletion
- Auto-generation of parking spots
- Reservation with vehicle number tracking
- Dashboard visualizations (bar charts using Matplotlib)
- Search functionality by user or vehicle number

Additional Features:

- Dynamic UI with Bootstrap
- Flash messaging for feedback
- Form validation with Flask-WTF
- Prevent editing or deleting active lots/spots

Video

https://drive.google.com/file/d/1ubx6huMVtwjuewIy6Qe4KNiAIrpMshM8/view?usp=sharing