Vehicle Parking App - V1

Author

Name: Harshad

Roll Number: 23f3000591

Student Email: 23f3000591@ds.study.iitm.ac.in

About me: I am a standalone B.S. student in Data Science and Applications at IIT Madras. Currently, I am in the Diploma level of this program. This program has strengthened my skills in programming, statistics, and analytical thinking.

Description

To build a Flask-based Vehicle Parking Web Application that allows users to find, book, and release parking spots in real-time. Admins should be able to manage parking lots, monitor spot usage, and user activity. The goal is to create a responsive, user-friendly interface with proper authentication, data handling, and dashboard functionalities.

Technologies used

Backend: Flask, SQLAlchemy, Flask-Login, Flask-Flash, Jinja2

Frontend: HTML, CSS, Bootstrap (Frontend framework used for building responsive, mobile-friendly layouts with ready-to-use components.)

Database: SQLite

DB Schema Design

Users: id, username, password, role, balance

Admin: id, username, password, user_type

ParkingLots: id, name, address, pin code, price

ParkingSpots: id, lot_id, is_reserved

Reservations: id, user_id, spot_id, parking_time, leaving_time, cost

Rationale:

The database is designed with normalization in mind to avoid redundancy and ensure data integrity. Separate tables for users, parking lots, spots, and reservations allow clear relationships and easy scalability. Role-based access is handled via a `role` field in the user table for simplicity and flexibility.

API Design

Implemented RESTful APIs (`/user_routes.py`) with route-based logic and dynamic rendering.

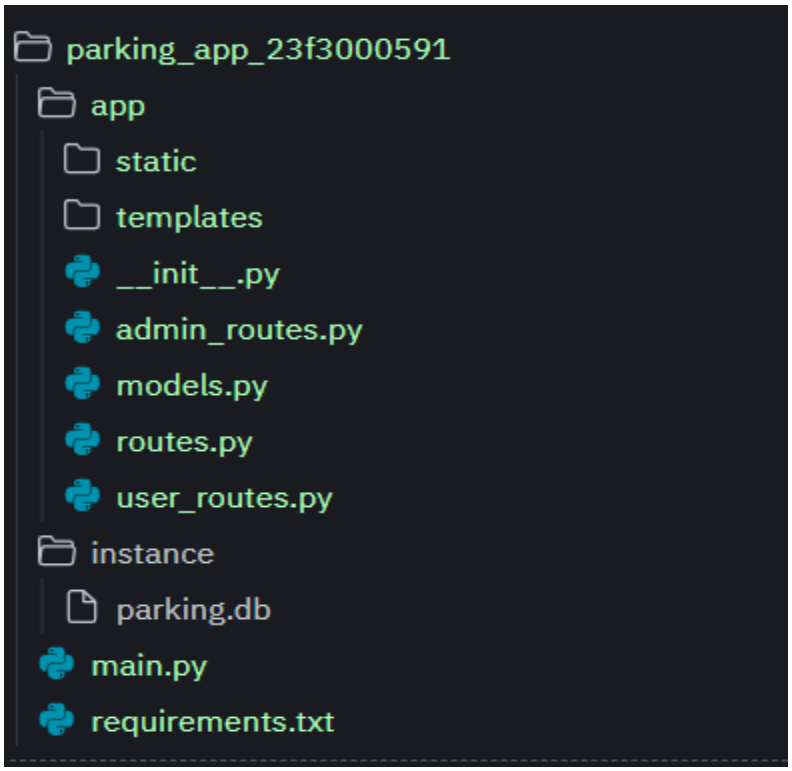
Endpoints:

- `GET /dashboard` → User dashboard with lots and reservation history
- `GET /reserve/<lot_id>` → Reserve spot in selected lot
- `GET /release` → Show user's active reservations
- `GET /release/<reservation_id>` → Release selected reservation
- `POST /confirm_payment` → Confirm and deduct payment
- `POST /delete_history` → Delete past reservations
- `GET /account` / `POST /account` → View & update user profile

Standard Response: Flash messages + HTML rendering

Architecture & Features

Structure:



The project follows a modular Flask MVC (Model-View-Controller) architecture. All route controllers are organized in `user_routes.py` and `admin_routes.py`, handling both user and admin functionalities respectively. HTML templates are stored in the `templates/` directory, styled using Bootstrap for responsiveness. Database models such as `User`, `ParkingLot`, `ParkingSpot`, and `Reservation` are defined in `models.py` using SQLAlchemy ORM. Static assets like CSS is placed in the `static/` folder.

Features Implemented:

- User Authentication (Login/Register) using Flask-Login
- Dynamic Spot Reservation per selected lot
- Spot Release with selection when multiple reservations exist
- Billing & Cost Calculation based on duration
- Balance Management & Payment Confirmation
- Admin Dashboard for managing lots, spots, and viewing all reservations
- Reservation History with deletion and total spending calculation
- Time Handling and error feedback using flash messages
  Additional enhancements include automatic status updates, formatted UI, and input validations.

Screen Recording 2025-07-31 114445.mp4