Vehicle Parking App - V1

Project Report

Author

Name: Sarang Gajanan Rao
Roll Number: 24F2000232
Email: 24f2000232@ds.study.iitm.ac.in

About Me:
I'm a Diploma student at IIT Madras BS in Data Science, with strong interests in computer vision, machine learning, and AI. This project helped me apply my web development skills to solve a real-world problem while gaining experience with backend technologies.

Project Description

The objective is to build a Flask-based Vehicle Parking Web Application that enables users to seamlessly find, book, and manage parking spots, while allowing admins to oversee parking lots, monitor spot usage, manage user data, and analyze revenue and performance through an interactive dashboard.

Technologies Used

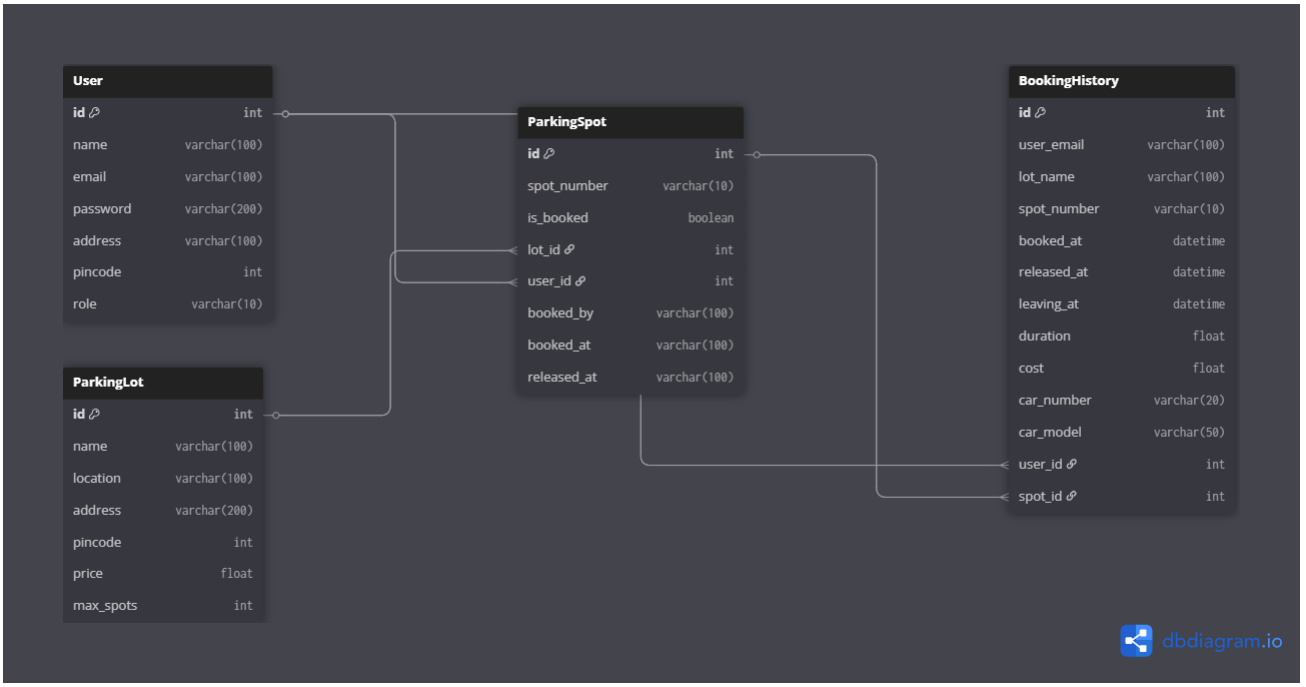
- **Backend:** Flask (web application framework)
  SQLAlchemy (ORM for database communication)
  Werkzeug (secure password hashing and utilities)
- **Frontend**:
  Bootstrap 5.3 (responsive UI)
  Chart.js (visual analytics)
- **Database**:
  - **SQLite** (lightweight, file-based database)

DB Schema Design

- **Users:**
  id, name, email, password (hashed), address, pincode, role
- **ParkingLots:**
  id, name, location, address, pincode, price, max_spots

- **ParkingSpots:**
  id, lot_id (FK), spot_number, is_booked, booked_by, booked_at, released_at
- **BookingHistory:**
  id, user_email, lot_name, spot_number, booked_at, leaving_at, duration, cost, car_number, car_model

Design Rationale:
Clear role separation, referential integrity via foreign keys, automatic spot creation based on lot capacity, and tracking of booking history ensures smooth data flow and minimal redundancy.



API Design

Implemented RESTful APIs (/api/*) with JSON responses.

Endpoints:

- ***GET /api/lots*** → *Fetch all parking lots*
- ***GET /api/spots*** → *Fetch all parking spots*
- ***GET /api/spots/<lot_id>*** → *Fetch parking spots for a specific lot*
- ***GET /api/user_bookings/<email>*** → *Fetch active bookings of a user*
- ***GET /api/user_history/<email>*** → *Fetch booking history of a user*
- ***GET /api/users*** → *Fetch list of all registered users*

Standard Response:
{ status, message, data }

Note:

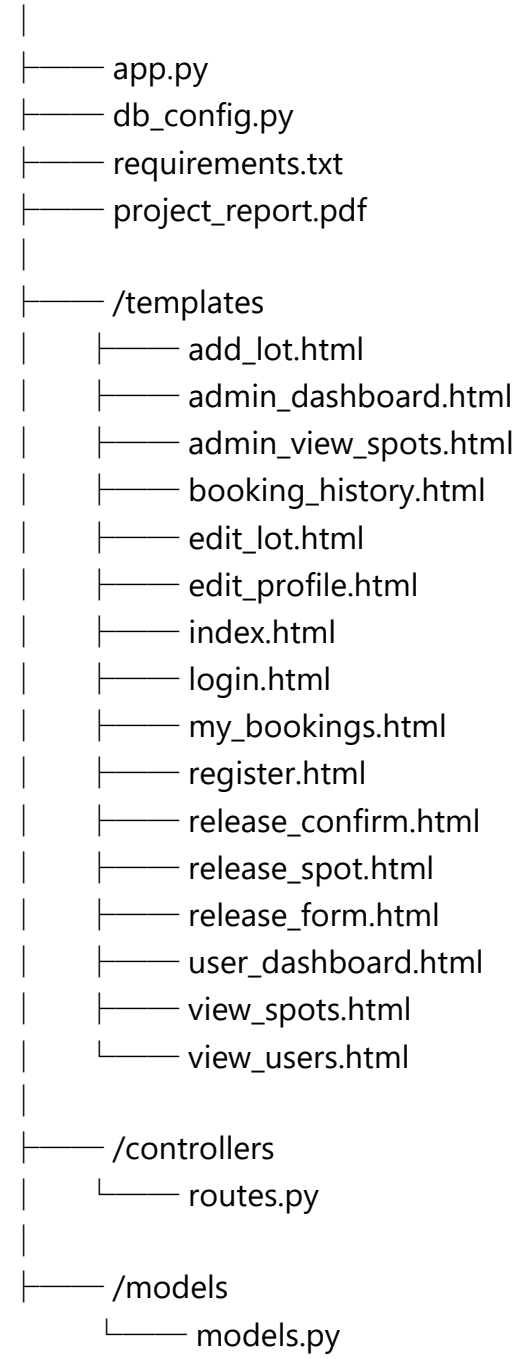
Booking and releasing actions are handled via HTML form submissions, not through API POST routes in this version.

Note on Use of AI

AI tools (LLMs) were used during development for debugging, error resolution, and implementation brainstorming. Tried LLM's to learn new things about this. Learnt bootstrap and frontend enhancing features from LLMs.

Architecture & Features

Structure:

parking_app_24f2000232 (This is your project root folder)


Features Overview

- Manual session management using Flask's session dictionary (Flask-Login not used)
- Role-based access control (Admin/User) via custom logic
- Passwords hashed using Werkzeug
- Bootstrap 5 based UI with basic alerts
- Chart.js analytics for admin revenue and usage stats
- Booking actions performed via form submissions (page reloads)

Key Functionality

User Dashboard:

- View available parking slots with status (Free/Booked).
- Book and release parking slots (via form; no live timers).
- View booking history (basic table format).
- Update profile information (name, address, pincode).
- No double-booking enforced using server-side check when booking.

Admin Panel:

- Add, edit, delete parking lots (including automatic spot creation when adding lots).
- View all parking spots in each lot (read-only; no individual spot edit/delete).
- View all registered users and their booking history.
- View spot status: green (available), red (booked).
- View simple revenue and usage analytics using Chart.js graphs.

Video

Video Link: https://drive.google.com/file/d/1HcNYcSoluTjk5DsQ0oN5-POwj8y6pVGt/view?usp=sharing