Project Report: Vehicle Parking App - V1

Author

Patel Srushti Niranjan
23f1002007
23f1002007@ds.study.iitm.ac.in
I am a dual-degree student and this is my first full-stack Flask project. I enjoyed learning backend development and database integration through it

Description

The objective of the project is to develop a multi-user vehicle parking management app. The app allows admins to manage parking lots and users to book/release parking spots. The focus is on time-stamped booking and charge calculation.

Technologies Used

- Python 3.10
- Flask (backend)
- Jinja2 (templating)
- SQLite (database)
- HTML/CSS/Bootstrap (frontend)
- JavaScript + Chart.js (for charts)
- Flask extensions: Flask-SQLAlchemy, Flask-Login, Flask-WTF

Purpose: These were chosen to simplify app development using Python and support modular development with templates and database models.

DB Schema Design

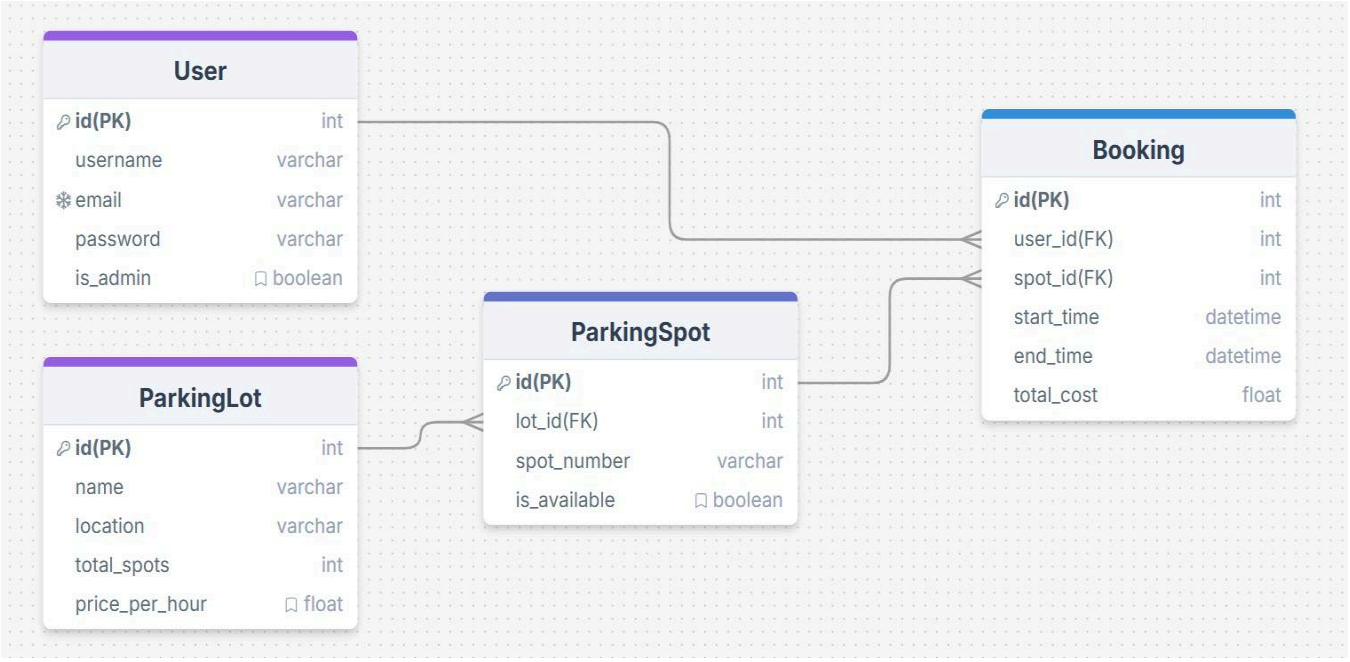
Our database consists of four main entities: User, ParkingLot, ParkingSpot, and Booking.

- **User** table stores authentication details. **email** is unique, and **is_admin** identifies admin access.
- **ParkingLot** represents each parking area with attributes like **location**, **total_spots**, and **price_per_hour**.
- **ParkingSpot** is linked to a ParkingLot via **lot_id (FK)**, with each spot having a **spot_number** and **is_available** flag.
- **Booking** logs user parking activity, linking users and spots with **start_time**, **end_time**, and **total_cost**.

All tables use an **id** as the primary key. Foreign keys ensure referential integrity across tables.

Design Rationale:

- Separation of User and Booking allows tracking multiple bookings per user.
- Splitting ParkingLot and ParkingSpot allows flexible management of spots within lots.
- Storing start_time and end_time in Booking enables precise cost calculation and parking duration analysis.



API Design

This project does not use full REST APIs. However, a minimal internal JSON API endpoint (/charts/data) was implemented to dynamically fetch chart data for admin and user dashboards using Chart.js. Data exchange was handled using fetch() in JavaScript, with backend response via Flask's jsonify().

Architecture and Features

Project Architecture

The Vehicle Parking App – V1 uses a modular architecture built on the Flask web framework. The core logic resides in the app/ directory with clear separation of concerns.

- Controllers (Routes): Found in app/routes/ as admin.py, user.py, and auth.py.
  - admin.py handles admin tasks like lot creation, user management, and charts.
  - user.py manages user features such as booking, releasing, and viewing history.
  - auth.py handles authentication, including login, logout, and role redirection.
- Models: Defined in app/models.py using SQLAlchemy, covering Users, Parking Lots, Spots, and Booking with proper relationships.
- Templates (Frontend): Located in app/templates/, organized into admin/ and user/ folders. Shared layout via base.html uses Bootstrap and Jinja2.
- Static Files: CSS and JavaScript (e.g., Chart.js) are stored in static/.
- Database Setup: create_db.py initializes the database schema.
- App Entry: app.py starts the app; __init__.py configures and registers routes.

Implemented Features

Core Features:

- **User Registration & Login**: Includes validation and password hashing. Role-based redirection sends users/admins to their respective dashboards.
- **Admin Authentication**: Predefined admin (no registration) secured via @admin_required decorator.
- **Parking Lot Management (Admin)**: Admins can create, edit, delete lots. Spots auto-generated based on lot capacity.
- **User Management (Admin)**: View all registered users, their bookings, and spot details.
- **Spot Status View**: Admin can see real-time availability of spots per lot.

User Features:

- **Book/Release Spot**: Users can book available spots and release them. Cost is calculated using timestamps and shown before confirmation.
- **Booking History**: Users can view active and past reservations.
- **Charts (User)**: Visualizes user activity – active vs. released spots and past booking costs.

Admin Analytics:

- **Charts (Admin)**: Bar chart showing bookings per lot.
- **Earnings Summary**: Displays total earnings from all released sessions.

Additional Features:

Includes role-based access, validations, auto spot generation, earnings tracker, and dynamic charts via Chart.js.

Video

<<Link to your online video of not more than 3 minutes length>>