Author

Namit Gupta
Roll No.: 23F2003526
Email: 23f2003526@ds.study.iitm.ac.in
A recent psychology graduate trying to break into tech. Tinkerer.

Description

A full stack, multi-user vehicle parking app needs to created, which allows users to reserve and book parking spots from parking lots. It aims to simplify the hassle of trying to find a parking spot in otherwise busy and crowded parking lots.

AI/LLM Used percentage is about 15% for Styling/UI, ChartJS, and solving errors.

Technologies used

1. Celery – To handle background tasks like sending reminders and reports asynchronously.
2. Flask – Lightweight Python web framework to build the backend API.
3. Flask-Caching – To improve performance by caching frequently accessed data.
4. Flask-CORS – To enable secure cross-origin requests from the VueJS frontend.
5. Flask-RESTful – To structure the API with clean, resource-based endpoints.
6. Flask-SQLAlchemy – ORM integration to easily interact with the database.
7. Jinja2 – To render dynamic templates where required (e.g., emails).
8. Pyexcel – To handle Excel/CSV import and export functionality.
9. PyJWT – To securely create and verify JSON Web Tokens for authentication.
10. Redis – To manage Celery queues and serve as a fast in-memory cache.
11. SQLAlchemy – To provide ORM and database abstraction in Python.
12. SQLite – Lightweight relational database for local data storage.
13. VueJS (Vue CLI, Vite) – Progressive JavaScript framework used for building the frontend UI.
14. Pinia – State management library for Vue to manage user and app data globally.
15. Axios – To make HTTP requests from the frontend to the backend API.
16. Vue Toastification – To display clean, non-intrusive toast notifications to users.

DB Schema Design

1. Role Table (roles)

Column	Type	Constraints	Description
id	Integer (PK)	Primary Key	Unique identifier
name	String(50)	Unique, Not Null	Role name (admin/user)

- Relationship: One-to-many with users table.

2. User Table (users)

Column	Type	Constraints	Description
id	Integer (PK)	Primary Key	Unique identifier
public_id	String(50)	Unique	Public UUID for users
name	String(100)		Full name
email	String(70)	Unique	User email
password	String(80)		Password hash
role_id	Integer (FK)	Foreign Key → roles.id	Defines user role

- Relationship: One-to-many with vehicles.

3. Vehicle Table (vehicles)

Column	Type	Constraints	Description
id	Integer (PK)	Primary Key	Unique identifier
license_plate	String(20)	Unique, Not Null	Vehicle license plate
vehicle_type	String(20)	Not Null	Vehicle type (car/bike)
user_id	Integer (FK)	Foreign Key → users.id, Not Null	Owner of the vehicle

- Relationship: One-to-many with bookings and reservations.

4. ParkingLot Table (parking_lots)

Column	Type	Constraints	Description
id	Integer (PK)	Primary Key	Unique identifier
prime_location_name	String(100)	Not Null	Parking lot name
address	String(255)	Not Null	Parking lot address
price	Float	Not Null	Price per hour/day
pin_code	String(10)	Not Null	Area pin code
number_of_spots	Integer	Not Null	Total number of parking spots
created_at	DateTime	Default: current time	Record creation time

- Relationship: One-to-many with parking_spots.

5. ParkingSpot Table (parking_spots)

Column	Type	Constraints	Description
id	Integer (PK)	Primary Key	Unique identifier
spot_number	String(20)	Not Null	Unique identifier per lot
spot_type	String(20)		Spot type (compact, EV, etc.)

Column	Type	Constraints	Description
is_occupied	Boolean	Default False	Spot occupancy flag
lot_id	Integer (FK)	Foreign Key → parking_lots.id	Parent parking lot
status	String(20)	Default 'available'	Current spot status

- Relationship: One-to-many with bookings and reservations.

6. Booking Table (booking)

Column	Type	Constraints	Description
id	Integer (PK)	Primary Key	Unique identifier
start_time	DateTime	Not Null	Booking start time
end_time	DateTime		Booking end time
vehicle_id	Integer (FK)	Foreign Key → vehicles.id, Not Null	Booked vehicle
spot_id	Integer (FK)	Foreign Key → parking_spots.id (SET NULL)	Booked spot
created_at	DateTime	Default: current time	Record creation time

7. Reservation Table (reservation)

Column	Type	Constraints	Description
id	Integer (PK)	Primary Key	Unique identifier
vehicle_id	Integer (FK)	Foreign Key → vehicles.id, Not Null	Vehicle that reserved spot
spot_id	Integer (FK)	Foreign Key → parking_spots.id, Not Null	Reserved spot
created_at	DateTime		Time reservation created
expires_at	DateTime		Expiry timestamp

API Design

Authentication and User Management:

- /signup, /login, /logout allow users to register, log in, and log out.
- /whoami returns the profile of the authenticated user.
- /admin/users enables admins to manage user accounts.

Parking Lot Management (Admin):

- /admin/lots (create and list parking lots)
- /admin/lots/<lot_id> (update or delete a lot)
- /admin/lots/<lot_id>/spots/<spot_number> (manage specific parking spots)
- /admin/lots/<lot_id>/spots/<spot_number>/active-booking and /active-reservation for viewing current bookings and reservations.

User Parking Operations:
- /lots (list available lots for users)
- /lots/<lot_id>/summary (view summary of a specific lot)

Vehicles:
- /vehicles to register or view a user's vehicles.

Bookings and Reservations:
- /bookings (create and view bookings)
- /bookings/<booking_id>/release (release a booking)
- /reservations (create and manage reservations)
- /admin/bookings (admin view of all bookings)

Architecture and Features

Backend (backend/)
This folder contains **Python Flask** backend.
- **Controllers / Route Handlers:**
  - Found in routes.py and partially in auth.py, api.py.
- **Models:**
  - Defined in models.py.
- **Configuration & Utilities:**
  - config.py handles settings.
  - utils/ contains helper functions for batch jobs.
  - parsers.py is used for request/response parsing or validation.
- **Celery Support:**
  - celery/ contains background tasks which are managed using Celery.

Frontend (frontend/)
This is a Vue.js-based frontend organized under the src/ directory.
- **Views:**
  - Located in src/views/, these .vue files represent actual route pages like LoginView.vue, DashboardView.vue, UsersView.vue.
- **Components:**
  - Reusable components like Navbar.vue, ParkingLot.vue, and modals are in src/components/.
- **Routing:**
  - Defined in src/router/index.js.

- **State Management:**
    - Pinia initialized in src/stores/user.js.

Project Root

- **Database:** instance/Database.db is a SQLite database.
- **Tasks:** celerybeat-schedule and migrations/ contain scheduled tasks and database migration setup.
- **Entry Point:** main.py for Flask.

Core Features Implemented

Authentication

- Role-based login with separate forms for admin and users.
- A default admin is auto-created.
- Roles are managed via user model and Flask session-based or JWT authentication.

Admin Dashboard

- Admin can create/edit/delete parking lots (only deletable if empty).
- Parking spots are auto-generated based on lot capacity.
- Admin can view parking statuses, vehicle details, all users, and parking summary charts.

User Dashboard

- Users select a lot; first available spot is auto-assigned.
- Users can occupy/release a spot, and timestamps are recorded.
- Summary charts display parking history.

Backend Jobs

Scheduled Tasks

- **Daily Reminders:** Email sent to inactive users or newly created lot users, prompting booking.
- **Monthly Reports:** HTML report emailed monthly with stats like most used lot, bookings, cost, etc.

User-Triggered Task

Export as CSV: User can trigger a batch job to export their parking data (spot IDs, timestamps, cost, remarks).

Performance & Caching

- Caching implemented to optimize API performance.
- Cache expiry is configured for stale data removal.

Video

https://drive.google.com/file/d/1szLK7FZB-J6D19BWubvgfKlsSwf7irf4/view?usp=sharing