Parking App

30.11.2025

Presented By : Kshitij Pandey

Roll No. : 23f2000345

Institute : Indian Institute of Technology, Madras.

About Me

Name : Kshitij Pandey

Roll No. : 23f2000345

Student Mail : 23f2000345@ds.study.iitm.ac.in

Project Overview

Project Title: Vehicle Parking App – V2

Problem Statement: To design and build a multi-user vehicle parking management application that enables administrators to manage parking lots, parking spots, and pricing—while allowing users to reserve, occupy, and release parking spaces.

The system must support real-time status updates, role-based access control, activity tracking, dashboards, caching, and asynchronous batch jobs.

Approach

I built the project using a modular Flask REST API as the backend and Vue.js 3 for the frontend, keeping a clean separation between API, UI, caching, and background jobs. Authentication & Authorization: Implemented JWT-based authentication and role-based authorization with separate flows for Admin and User accounts.

Admin features: Admin can create, edit, and delete parking lots; automatically generate parking spots; view real-time spot availability; monitor all users and parking activities; access comprehensive analytics with charts showing revenue trends and usage patterns.

User features: Users can register/login, view available parking lots, book parking spots with vehicle number tracking, automatically get the first free spot, check-out with automatic cost calculation, and view their complete reservation history with personal analytics.

Caching: Used Redis caching for frequently accessed data such as lot listings, spot statuses, and analytics to improve performance with automatic cache invalidation on data updates.

Background jobs: Implemented Celery background jobs for:

1. Daily reminder notifications to users with active reservations
2. Monthly activity report generation and emailing to admins
3. Asynchronous CSV export of reservation history with email delivery
4. Analytics and Dashboards: Integrated Chart.js to display comprehensive reports including revenue trends, occupancy rates, popular parking lots, and user-specific analytics.

AI/LLM

I primarily used **ChatGPT 5.1** and **Claude Sonnet** for UI improvement suggestions, debugging support (particularly with Redis and Celery), documentation refinement, and code readability enhancements.

Overall AI involvement was around **20%**, limited to suggestions and troubleshooting.

All core development—including **database and API design, backend architecture, validations, Redis/Celery integration, and full frontend implementation**—was completed **manually** by me.

API Endpoints

Authentication

- POST /api/auth/register
- POST /api/auth/login
- GET /api/auth/me
- POST /api/auth/logout

Admin

- GET /api/admin/lots
- POST /api/admin/lots
- GET /api/admin/lots/{lot_id}
- PUT /api/admin/lots/{lot_id}
- DELETE /api/admin/lots/{lot_id}
- GET /api/admin/users
- GET /api/admin/analytics/overview

User Parking

- GET /api/lots
- GET /api/lots/{lot_id}
- POST /api/reservations
- PUT /api/reservations/{id}/checkout
- GET /api/reservations/my
- GET /api/analytics/my

Export (Async Jobs)

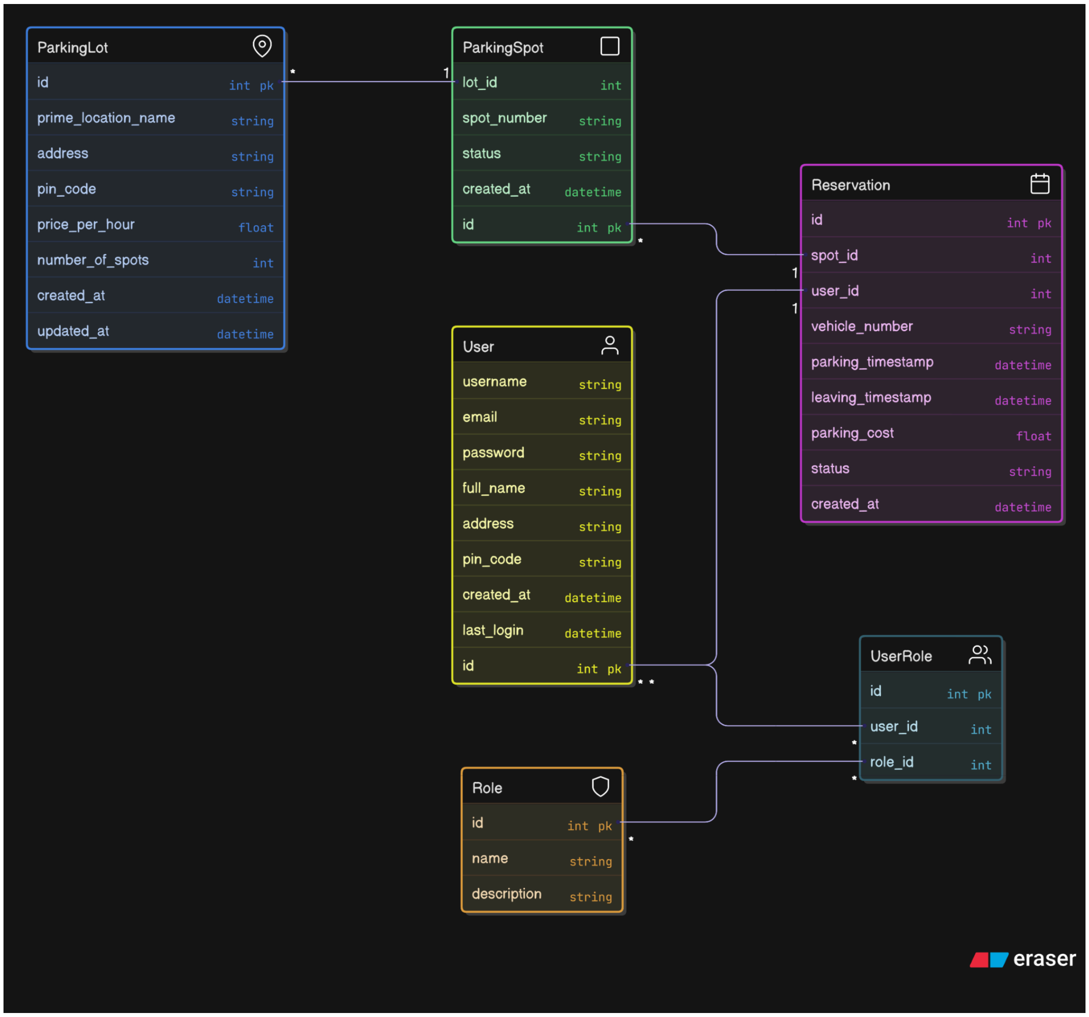
- POST /api/export/reservations
- GET /api/export/status/{task_id}

Technologies Used

- Flask: A lightweight Python web framework for building the RESTful backend API with JWT authentication.
- Vue.js 3: A progressive JavaScript framework for building dynamic and reactive user interfaces with Composition API.
- SQLAlchemy: Used for database management, providing an ORM for interacting with the database with relationship modeling.

- **SQLite: A lightweight disk-based database for storing application data including users, parking lots, spots, and reservations.**
- **Redis: Used for caching frequently accessed data and as a message broker for Celery.**
- **Celery: Used for background tasks for sending automated emails, daily reminders, and CSV exports.**
- **Chart.js: A JavaScript library for creating interactive charts and data visualizations.**
- **Pinia: State management library for Vue.js 3 for managing authentication and global state.**
- **Bootstrap 5: CSS framework for responsive and modern UI design.**
- **Pytest: Testing framework for comprehensive unit and integration tests.**

DB Schema



THANK-YOU

Gdrive Link:- Video Link