Parkly: A Parking Facility Management Application

Modern Application Development 2 Project Report

Submitted by

Mohit Raj Rathor

Roll No. 22f3003109

IITM Online BS Degree Program,

Indian Institute of Technology Madras, Chennai

Tamil Nadu, India, 60036

Author

Details:

- Name: Mohit Raj Rathor
- Roll number: 22F3003109
- Email: 22f3003109@ds.study.iitm.ac.in

About me:

I'm Mohit Raj Rathor, a Diploma student pursuing BS in Data Science & Applications at IIT Madras. I'm passionate about app development and web3, and have hands-on experience with Flask, Vue.js, SQLite, Redis, and Celery.
In this project, I built the backend with Flask, implemented Redis caching, async tasks with Celery, and developed the Vue.js frontend with PWA support and analytics. It gave me practical experience in building scalable, real-world, multi-user systems.

Description

Parkly is a responsive, multi user web application developed for seamless management of 4-wheeler parking lots. It provides essential features for both users and administrators, enabling real-time booking, slot management and analytics.

The backend is built with flask, exposing RESTful APIs to manage users, parking, slots, reviews and reservations. It uses Flask-SQLAlchemy as the ORM with SQLite for lightweight data storage. The system supports daily and scheduled jobs such as remainder emails and monthly reports using Celery and Redis.

On the frontend, Vue.js is used to cerate a dynamic and reactive user interface, styled with Bootstrap to ensure a clean, mobile-responsive design. The application includes both client-side and server-side validation for robust user input handling.

Technology used

This project leverages a modern full-stack development approach. The key technologies used are:

Backend:

- **Python – Flask** and it's extension libraries like lightweight WSGI web framework used to develop RESTful APIs and handle application logic.
- **SQLite –** Lightweight, relational database for storing application data.
- **Celery -** Distributed task queue used for background job processing.
- **Redis –** In memory database used as a Celery message broker and for caching frequent queries.

Frontend:

- **Vue.js –** Progressive JavaScript framework used to build dynamic, component based user interfaces.
- **Bootstrap 5 –** Frontend CSS framework for responsive design grid system and pre-styled UI components.
- **JavaScrip –** Client-side scripting for form validation and interactivity.

Communication:

- **Flask-Mail –** For sending automated emails (for - user Email verification, daily remainder, monthly reports and promotional mail)

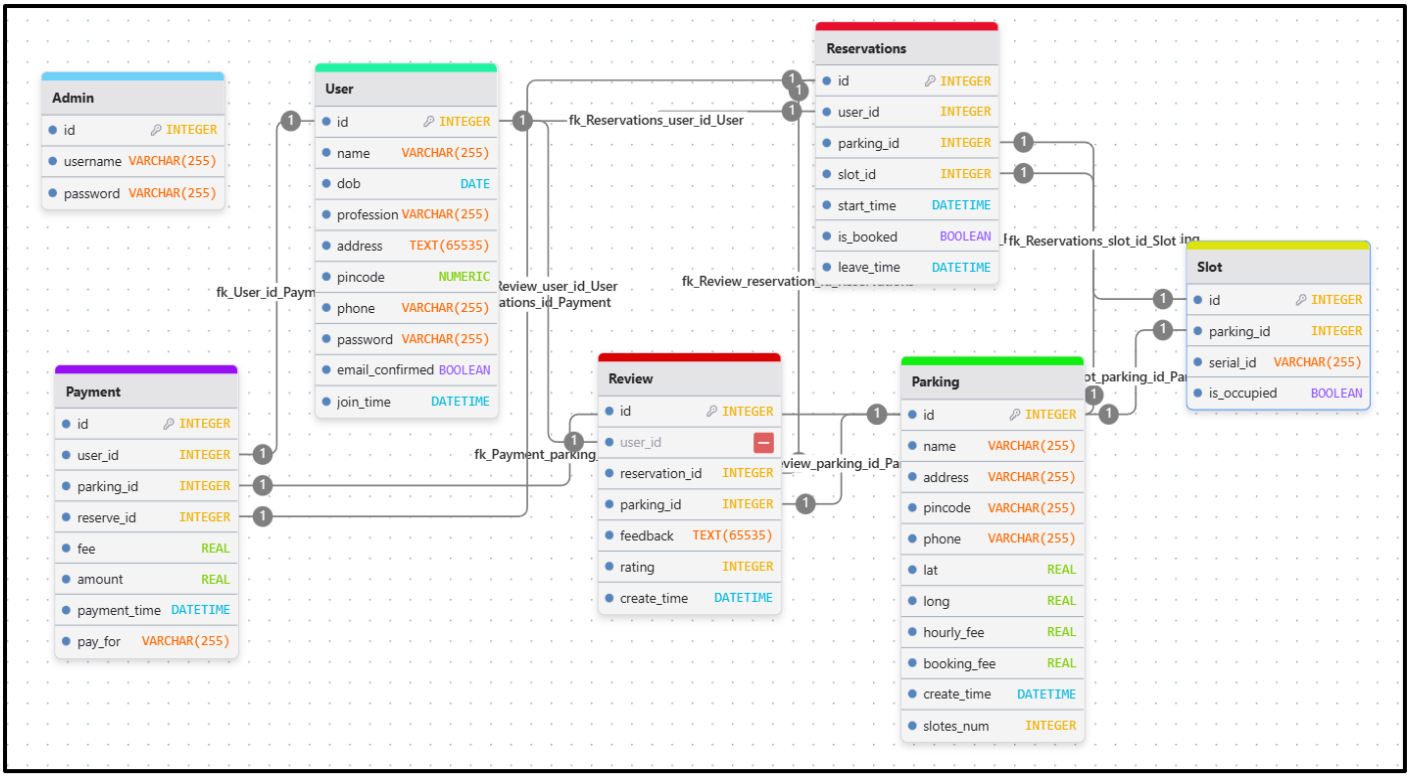
Tools:

- **Visual studio code –** Lightweight IDE for development, support multiple languages and open-sourced.
- **Bruno –** An open-source APIs documentation and testing to

DB Schema Design

The data base consists of 7 primary tables and relation between them as follow:

Table Name	Purpose
Admin	Stores login credentials of admin users.
User	Manages end-user profiles, including personal and authentication details.
Parking	Contains information about parking lots.
Slot	Manages individual slots within a parking area.
Reservations	Tracks reservations of slots by users.
Review	Captures user feedback and ratings per reservation.
Payment	Records payment transactions for reservations and related fees.



API Design

This project follows a RESTful-API structure using Flask, organised under the api_v1 module for modularity and version control. Following are the API sub-module overview:

- **auth.py**: User/admin login, signup, and secure jwt token-based auth.
- **user.py**: Profile updates, email verification, user-specific data access.
- **parking.py**: Admin CRUD for parking, search, and slot availability.
- **reservations.py**: Create, manage, and view reservations.
- **analytics.py**: Provides usage insights and monthly summaries.
- **task_routes.py**: Background task triggers and CSV export endpoints.
- **test.py**: Dev-only test routes.

API documentation Yaml file link: https://drive.google.com/file/d/1ik8yGPxGU2VJto-hAJOWTgODJ_VxY7OP/view?usp=sharing

Architecture and Feature

This project is a full-stack Vehicle Parking Management System built using Flask and Vue.js. It follows the MVC pattern, ensuring organized code structure and separation of concerns.

Backend (Flask):

- RESTful APIs for user registration (with email confirmation), login, parking lot CRUD, slot management, payments, and reviews.
- Slot reservation is real-time through REST API.
- Implemented data validation using Marshmallow and comprehensive error handling with try/except blocks.
- Used SQLite with Flask-SQLAlchemy ORM for data persistence.
- Integrated Redis for caching and Celery for asynchronous task execution (e.g., daily reminders, Monthly email reports).
- Secured routes using Flask-jwt-extended.

Frontend (Vue.js):

- Built a responsive UI with HTML5, CSS, and JavaScript (Vue.js).
- Client-side form validation using HTML5 and JavaScript.
- Served the Vue build through Flask.
- Notifications implemented on frontend.

Features:

- Admin dashboard for managing parking, users, and slots (add/delete/increment/decrement).
- Users can search, book, and review parking.
- Real-time parking display using Leaflet + OpenStreetMap API.
- Fetch user address using OpenMap API.
- Monthly analytics and CSV reports sent via email.
- Review and rating system for parking.

Video Presentation & Other files

Video presentation of this project can be found at:
https://drive.google.com/file/d/124EJYwMvpSyr8suiZPDalSBVIHLy5k0m/view?usp=sharing

All other project related files like code, project report copy and API documentation is present in following drive folder: https://drive.google.com/drive/folders/1_rg-diPTQkFOICdCCY6dW7V27LQeb-Zh?usp=sharing

References

- **Flask Documentation** – https://flask.palletsprojects.com/
- **Flask-SQLAlchemy Documentation** – https://flask-sqlalchemy.palletsprojects.com/
- **Flask-JWT-Extended Documentation** – https://flask-jwt-extended.readthedocs.io/
- **Flask-Smorest Documentation** – https://flask-smorest.readthedocs.io/
- **Celery Documentation** – https://docs.celeryq.dev/en/stable/
- **Redis Documentation –** https://redis.io/docs/
- **Vue.js Documentation –** https://vuejs.org/
- **Bootstrap Documentation –** https://getbootstrap.com/
- **Leaflet.js Documentation –** https://leafletjs.com/
- **OpenStreetMap API Docs** – https://wiki.openstreetmap.org/wiki/API
- **Vue-Chart.js Documentation** – https://vue-chartjs.org/
- **Swagger/OpenAPI Specification –** https://swagger.io/specification/
- **GitHub repository of the project –** https://github.com/mohitrajrathor/parking_app_v2