App Dev Project Report

1. Student Details

Name: Alan B George

Roll Number: 21f2000348

Email: 21f2000348@ds.study.iitm.ac.in

About Me: I am currently pursuing the Programming and Data Science degree from IIT Madras. I work as a data analyst in a microfinance company and I am interested in backend development and full-stack applications. This project helped me understand Flask, Vue.js, Redis and Celery by building a complete end-to-end web application.

2. Project Details

Project Title: Vehicle Parking System – MAD-2 Project

The Vehicle Parking System is a web application that allows users to search, book and manage parking slots in different parking lots. The backend is built using Flask with SQLite for data storage, while the frontend is implemented using Vue.js and Bootstrap. Redis and Celery are used for caching and background jobs such as CSV export, daily reminders and monthly activity reports. The application supports role-based access where normal users can book and view parking slots, and admins can manage parking lots, view registered users and trigger batch jobs.

3. Tech Stack and Library Distribution

Overall Tech Stack:

- Backend: Flask, SQLite, SQLAlchemy, Jinja2 (for basic templates), Python.
- Frontend: Vue.js 3, Vue Router, Axios, Bootstrap.
- Background Jobs & Caching: Celery, Redis.
- Email Testing / Tools: MailHog (local SMTP testing).

4. Key Functionalities Implemented

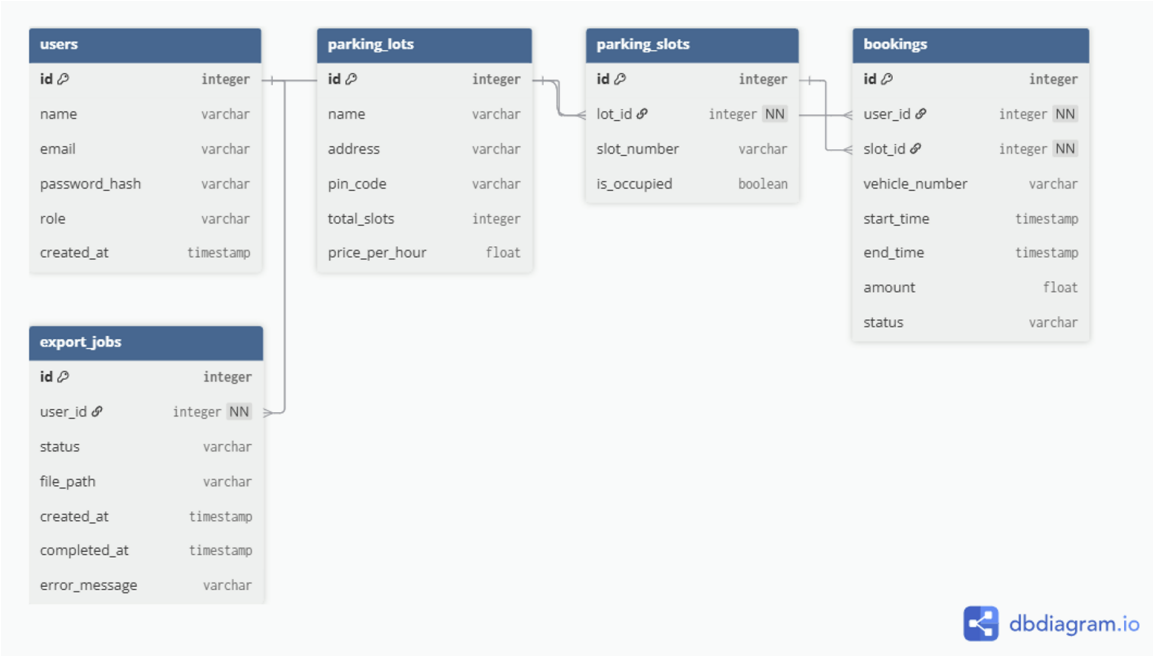
1. User registration and login with role-based access control (USER and ADMIN).
2. Admin dashboard to create, update and delete parking lots with configurable price per hour.
3. Automatic creation of parking slots for each parking lot and tracking of free / occupied status.
4. User dashboard to search parking lots by pin code, view free slots and navigate to slot booking.
5. Booking flow for selecting a slot, entering vehicle number and creating an ACTIVE booking.
6. Slot release that marks the booking as COMPLETED, frees the slot and calculates the final amount based on duration and price per hour.
7. Booking history page showing all past bookings, including amount and status, with an option to release active bookings.
8. CSV export of user bookings implemented as a background Celery job, with download link once the file is generated.
9. Admin view of registered users with an indicator showing whether the user currently has any active booking (Active / Inactive status).
10. Background batch jobs: cleanup of stale ACTIVE bookings, daily reminder emails for overdue bookings and monthly HTML activity reports.
11. Caching of parking lot search results in Redis to reduce repeated database queries for the same pin code.

5. Database Schema / ER Diagram

The Vehicle Parking System is built using a clean and modular relational database schema. The schema supports user authentication, parking lot creation, slot management, booking workflows, and background tasks such as CSV exports and automated reminders.

5.1 ER Diagram

The following diagram illustrates the relationship between the core tables in the database:



5.2 Table Descriptions

User: Stores user credentials and roles (ADMIN / USER).
ParkingLot: Represents a parking location with address, pin code, pricing and slot count.
ParkingSlot: Represents individual parking slots associated with a parking lot.
Booking: Tracks every booking including start/end time, amount and status.
ExportJob: Stores records of user-triggered CSV export jobs for booking history.

5.3 Relationships

One ParkingLot has many ParkingSlots (1 → N).
One ParkingSlot is referenced in many bookings over time (1 → N).
One User can have many Bookings (1 → N).
One User can trigger many ExportJobs (1 → N).

6. AI / LLM Usage Declaration

I used ChatGPT (LLM) for approximately 20–25% of the overall work in this project. The model was mainly used for the following activities:

- - Clarifying concepts in Flask, Vue.js, Redis and Celery when I was stuck.
- - Suggesting example code snippets for axios calls, route handlers and Celery tasks.
- - Helping refactor some functions to make the code cleaner and easier to understand.
- - Generating initial drafts for documentation text such as this report and the README file.

All critical implementation work such as wiring the routes, configuring the environment, fixing integration bugs and testing the complete flow (login, booking, release, CSV export and batch jobs) was done by me. I always reviewed, modified and tested any code suggested by the LLM to ensure that I fully understood what it was doing.

7. Key Learnings and Reflections

- I learned how to design a small but realistic backend using Flask blueprints, an application factory and SQLAlchemy models.
- I became more comfortable with Vue 3, especially reactive state, routing and composing multiple views into a single-page application.
- I understood the practical use of Redis and Celery for handling background jobs and caching in a web application instead of doing everything in the main request.
- I gained experience in running multiple services together in WSL (Flask server, Vite dev server, Redis, Celery worker and MailHog) and debugging issues when one of them failed.
- Overall, this project increased my confidence to build and debug full-stack applications on my own.

Video Presentation

Drive Link : https://drive.google.com/file/d/1yK4OClt3Qn72yvryuu-hLsw9x-sPs5tt/view