Parking Management System – Project Report

Author
Agnibhu Mandal
Roll Number: 21F1001209
Email: 21f1001209@ds.study.onlinedegree.iitm.ac.in

I'm currently pursuing the Programming and Data Science program from IIT Madras. I enjoy building full-stack apps and exploring the intersection of system design, software engineering, and practical AI integration.

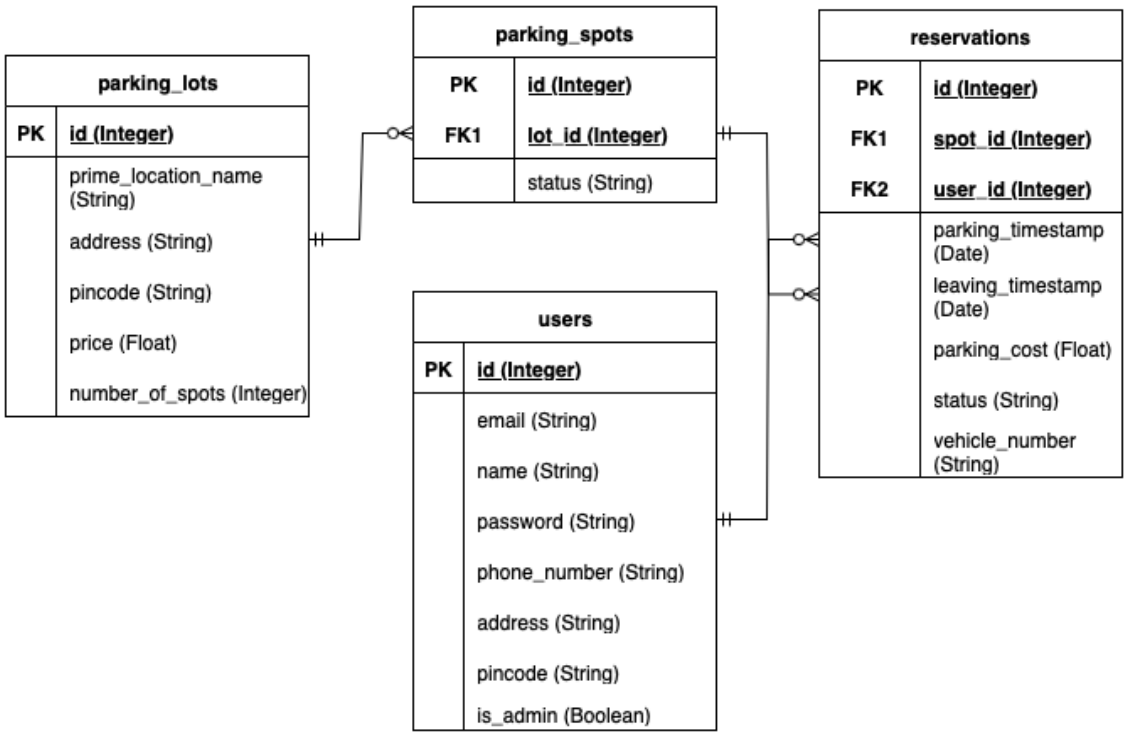
Description
This project implements a parking reservation management system with both admin and user features. It allows authenticated users to book, edit, export, and manage parking reservations while administrators can oversee user data.

AI/LLM Use:
Only used for minor assistance in UI rendering logic and debugging Celery issues. All core functionality was implemented manually.

Technologies Used
Frontend: Vue 3, Bootstrap 5, Axios, Pinia, Chart.JS
Backend: Flask, Flask-Login, SQLAlchemy, Flask-CORS, Flask-Mail
Task Queue: Celery
Message Broker: Redis
API Docs: Swagger UI (via YAML/OpenAPI)
Database: SQLite
Others: bcrypt (password hashing), dotenv (env variables), fpdf2 (pdf generation), honcho (automating shell commands)

Purpose:
Flask was chosen for its simplicity and modularity, Vue 3 for reactive frontend handling, and Celery+Redis for handling background email and export jobs.

Database Schema Design



API Design

APIs were created for the following features:
- Users: register, login, get user info, edit/delete users, admin search
- Reservations: book, view, cancel, list active reservations
- Exports: trigger CSV export via Celery background task
- Parking Lots and Spots: CRUD operations on parking lots and spots
- Docs: Swagger UI (/docs) built from OpenAPI YAML

YAML file submitted separately.

Architecture and Features

The project is structured as follows:

Backend: All routes are modular using Flask Blueprints (users_bp, reservations_bp, etc.). Business logic is handled in the routes and database models.
Celery Tasks: Defined in tasks.py, includes async email sending and CSV export.
Swagger Docs: YAML-based API spec exposed using Flask route and Swagger UI.
Vue Frontend: Organized into views and components. Axios used to call APIs. Navigation bar provides access to user profile, export, docs, and logout.

Implemented Features:
- Secure login using Flask-Login
- Admin vs. User views

- View/Edit profile
- Export reservations as CSV via Celery
- API docs accessible via browser
- Bootstrap-based responsive UI
- Redis caching

Additional Features:
- Swagger integration
- CSV export and time triggered email notifications
- Role-based access control for delete/edit

Video

https://drive.google.com/file/d/1huTDW_WnboSbt8NDMDRs-hjH5pHiT-3K/view?usp=sharing