

App Dev-II Project Report

1. Student Details

Name: Patel Srushti Niranjana

Roll Number: 23f1002007

Email: 23f1002007@ds.study.iitm.ac.in

About Me: I enjoy building structured, scalable web applications and exploring backend automation. My interests lie in Flask-based systems, UI frameworks, and cloud-style workflows.

2. Project Details

Project Title: Vehicle Parking Management App – V2

Problem Statement:

To develop a multi-user parking management system that handles parking lots, parking spots, and reservation of 4-wheeler slots. The app must support an admin role with full control over lots/spots and normal users who can book, release, and view their parking history.

Approach:

The system is developed using Flask for backend APIs and Vue.js for frontend UI. The backend follows a modular controller-based architecture, integrated with Redis caching and Celery for scheduled/background jobs. Users can register, book the first available spot, release it, and view charts and reports. Admin can manage lots, spots, users, and view dashboards.

3. AI/LLM Declaration

I used **ChatGPT (GPT-5.1)** only for **solving specific errors** during development and for minor improvements in documentation wording.

The extent of AI usage is approximately **8–10%**

4. Technologies & Frameworks Used

Technology / Library	Purpose
Flask	-Backend API framework
SQLite	-Lightweight local database for storing data
Vue.js (CLI)	-Frontend UI framework
Bootstrap 5	-Styling and responsive layout
SQLAlchemy	-ORM for database modeling
Redis	-Caching and Celery backend (broker & result backend)
Celery	-Background/scheduled task execution (daily reminders, monthly reports, CSV exports)
Chart.js	-Rendering admin and user charts
Flask-JWT / Flask-Login	-Authentication and role handling
MailHog	-Local SMTP server to catch and view all emails sent via send_email (daily reminders & monthly reports)

5. Database Schema / ER Diagram

The application uses a relational database with the following main tables:

roles			
id	int		
name	varchar(80)	NN	
description	varchar(255)		

users									
id	int								
username	varchar(100)	NN							
email	varchar(255)	NN							
password	varchar(255)	NN							
active	boolean	NN							
fs_uniquifier	varchar(255)	NN							
fs_token_uniquifier	varchar(255)	NN							
created_at	datetime								
updated_at	datetime								

user_roles			
id	int		
user_id	int	NN	
role_id	int	NN	

parking_spots									
id	int								
lot_id	int	NN							
spot_number	varchar(50)	NN							
status	varchar(1)	NN							
vehicle_number	varchar(20)								
reserved_at	datetime								
created_at	datetime								
updated_at	datetime								

parking_lots									
id	int								
prime_location_name	varchar(255)	NN							
address	varchar(255)	NN							
pin_code	varchar(10)	NN							
price_per_hour	float	NN							
number_of_spots	int	NN							
notes	varchar(512)								
created_at	datetime								
updated_at	datetime								

reservations									
id	int								
user_id	int	NN							
spot_id	int	NN							
parking_timestamp	datetime	NN							
leaving_timestamp	datetime								
parking_cost	float								
vehicle_number	varchar(20)	NN							
remarks	varchar(512)								
created_at	datetime								
updated_at	datetime								

6. API Resource Endpoints

The Vehicle Parking App provides RESTful APIs to manage users, parking lots, spots, and bookings. Endpoints were implemented for **user registration and login**, **admin dashboard and role management**, **booking/reservation of parking spots**, **releasing spots**, and **fetching user/admin-specific chart data**. Additional APIs were created for **exporting CSV reports** and **retrieving parking spot and lot details**. The APIs were built using **Flask-RESTful**, with **Redis caching** for frequently accessed data and **Celery** for asynchronous and scheduled tasks such as daily reminders and monthly reports. All endpoints, request/response details, and parameters are defined in the submitted api.yaml file.

7. Architecture and Features (optional)

Backend

The backend is built with Flask and organized as follows: `app.py` as the entry point, `model.py` for SQLAlchemy models, `cntrlrs/` for API controllers, `tasks/` for Celery jobs, `cache.py` for Redis caching, `celery_app.py` for task configuration, and `mail.py` for email utilities.

Frontend

The frontend uses Vue.js with a component-based structure. `views/` contains Admin and User pages, `components/` holds reusable UI elements, `api/axios.js` manages API calls, and `stores/` manages state with Pinia.

Features

- **Admin:** Manage parking lots, auto-generate spots, view users, dashboard with charts, monitor spot status.
- **User:** Register/Login, reserve/release spots, view booking history, personalized dashboard.
- **Background Tasks:** Celery + Redis for daily reminders, monthly HTML reports via email, and async CSV export.
- **Other:** Caching with expiry for frequently accessed APIs, full frontend & backend validation, responsive UI with Bootstrap.

8. Video Presentation

Video Link:

https://drive.google.com/file/d/1RsdwPmNyIFrD50-V_ZLD3GjZN8fZpif4/view?usp=sharing