App Dev Project Report

1. Student Details

Name: Rishabh Maurya

Roll Number: 21f1000014

Email: 21f1000014@ds.study.iitm.ac.in

About Me:

I am working as a frontend developer (System Engineer) in TCS (Mumbai) from 1 year. My main work is in .Net Framework and C# for developing and maintaining the code for a desktop application used in banking sectors.

---

2. Project Details

Project Title: Vehicle Parking System

Problem Statement:

To design and build a web-based application that allows users to **book and release** their vehicles from a spot in a parking lot based on certain cost and location.

Approach:

1. The app was built using Flask as the backend framework with a modular structure and database creation using Flask-SQLAlchemy during build. API calls were tested using Thunder Client API in Visual Studio Code.

2. Frontend is built using Vue.js, where the routing is done in index.js and the main vue file is App.js . Axios was used to make API calls to backend app.py file where every function is required to have a JWT based token for authentication after login.

3. Backend Jobs were done in the end of the project
   a. Daily reminders for lot creation and user inactivity
   b. Monthly HTML reports to Users
   c. User-triggered csv reports from dashboard.

---

3. AI/LLM Declaration

I used **ChatGPT (GPT-5)** to assist in writing few SQLAlchemy codes and few for backend jobs.

The extent of AI/LLM usage is around **15–20%**, limited to **code suggestions and documentation**.

All final implementation logic, debugging, and integration were done manually.

4. Technologies and Frameworks Used

Technology / Library	Purpose / Usage
Flask	Web framework for routing, request handling, and API endpoints
Flask-RESTful	Simplifies building REST APIs
Flask-JWT-Extended	JWT-based authentication for users
SQLAlchemy	ORM for handling SQLite database and queries
SQLite	Lightweight database for storing users, parking lots, and reservations
Werkzeug	Password hashing (used by Flask and security utilities)
Celery	Asynchronous background jobs (daily reminders, monthly reports, etc.)
Redis	Backend broker for Celery tasks
smtplib, email.mime	Sending daily and monthly emails
Jinja2	Rendering HTML templates for email reports
datetime, timezone, timedelta	Working with IST time, timestamps, scheduling
csv, io.StringIO, pathlib.Path	File handling, CSV generation, path management
Vue.js	Building reactive user interfaces
axios	Sending requests to backend APIs
Chart.js	Displaying graphical data like user activity, parking statistics
Git	Source code versioning
npm / Node.js	Frontend dependency management and building
Python virtual environment (venv)	Isolated Python environment for installing dependencies

5. Database Schema / ER Diagram

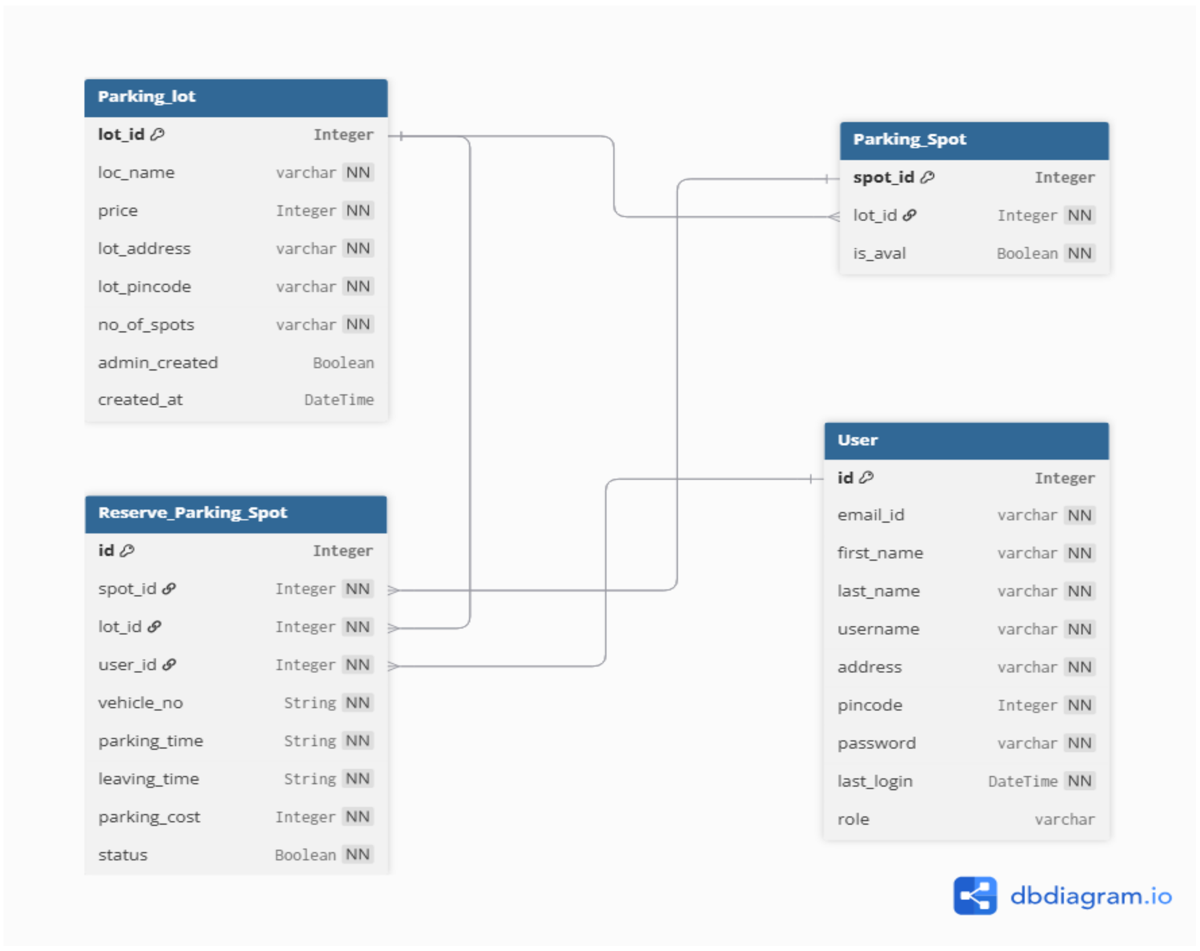
Tables:

1. **User** — stores user profile details (id, name, email, password)
2. **Parking_lot** — logs admin activities (lot_creation, lot_id, spot_id, address, location, pincode, no. of spots,creation time, admin_created,etc)
3. **Parking_Spot** — stores information about availability of spot inside a lot
4. **Reserve_Parking_Spot** — Stores user information (vehicle no, parking_time, leaving_time, cost, lot_id, spot_id, etc)

Relationships:

- One-to-Many → User → Reserve_Parking_Spot
- One-to-Many → Parking_lot → Reserve_Parking_Spot
- One-to-Many → Parking_lot → Parking_Spot
- One-to-One → Parking_Spot → Reserve_Parking_Spot

E-R Diagram:



5. API Resource Endpoints

Endpoint	Method	Description
/api/signup	POST	Add a new User
/api/signin	POST	Authenticate User and generate session
/api/logout	POST	To end user session
/api/userInfo	GET	Returns all users data
/api/UserParkingSpots	GET	Returns User based parking data
/api/parkingLotInfo	GET	Returns Spot and it's status
/api/parkingLot	GET	Returns Parking Lot details
	POST	Parking Lot creation
	PUT	To edit a Parking Lot
	DELETE	To delete a Parking Lot
/api/parkingSpot	GET	Returns particular Spot and it's status
	DELETE	To delete a Spot
/api/availableSpot	GET	For Parking Spot availability
/api/reserveParkingSpotAdmin	GET	Returns Reserved Parking Spot details for showing admin
/api/reserveParkingSpot	GET	Returns Reserve Parking Spot details for a particular user
	DELETE	To release a vehicle from a Parking Spot
/api/Search	GET	Returns lots based on selection and search by admin
/api/ParkingHistory	GET	User Based Parking history
/api/UserSearch	GET	Returns spots based on selection and search by user
/api/BookSpot	GET	Returns spot data selected
	POST	To book a Parking Spot
/api/userChart	GET	Returns parking spot status of a User
/api/adminChart	GET	Returns parking lot status

/api/userCsv	GET	Returns csv response

---

6. Architecture and Features (optional)

Architecture Overview:

- **app.py** – main Flask application entry point
- **models**.py – Flask SQLAlchemy Models
- **/components** – constains VUE files
- **/instance** – database
- **/router/index.js** – Activity routes
- **/templates** – Backend Jobs email template

Implementation Summary:

- **Authentication:** JWT-based login for Admin and Users with secure password hashing.
- **Admin Features:** Create/edit/delete parking lots, auto-generate spots, view users, track parking status, and view analytics.
- **User Features:** Auto-allocated parking spot, park/release workflow with timestamps, personal usage summary, and charts.
- **Dashboards:** Chart.js visualizations for both Admin and User analytics.

Automated Jobs:
- Daily reminder emails
- Monthly HTML activity report
- User-triggered CSV export of parking history

- **Backend:** Flask REST API, SQLAlchemy ORM, Jinja2 templates, Celery + Redis for scheduling, SMTP email service.
- **Frontend:** Vue.js with Bootstrap-based responsive UI.
- **Database:** SQLite (created programmatically; no manual DB tools).

Additional Features:
- **Monthly Activity Reports:** Professionally formatted HTML-based reports summarizing user parking activity.
- **Charts & Visualizations:** Integrated Chart.js for interactive and responsive analytics dashboards.
- **Backend Validation:** Implemented robust input validation within all API endpoints for secure operations.
- **Responsive UI:** Enhanced user experience through clean, responsive styling using Bootstrap and custom CSS.

- **Secure Login System:** Added role-based authentication using JWT to prevent unauthorized access.

---

8. Video Presentation
Drive Link:
https://drive.google.com/file/d/1MwUfczmXl1j09B8G697okUbDvw74srNg/view?usp=sharing

---