Vehicle Parking App — Project Report (MAD-1)

1. Student Details

- **Name:** Varun Singh
- **Roll Number:** 24F2008180
- **Email:** 24f2008180@ds.study.iitm.ac.in

2. Project Details

Project Title: Vehicle Parking App (MAD-1)

Problem Statement:
Develop a multi-user web application to manage vehicle parking lots and spots. The system should support two roles: Admin and User. The admin manages parking lots, creates them with automatic spot generation, edits, and deletes lots, and views comprehensive dashboard data. Users can register, login, choose parking lots (without selecting specific spots), book and release spots, and monitor their reservations and expenses.

Approach:

- Used **Flask** as the backend web framework.
- Designed the database using **SQLAlchemy** ORM with SQLite as the storage engine.
- Implemented session-based authentication to differentiate **Admin** and **User roles**.
- Automated spot creation tied to lot creation and adjustment.
- Enforced booking rules such as automatic spot assignment based on availability and matching vehicle type.
- Employed client-side UI built with **Bootstrap** and **Jinja2** templating for dynamic pages.
- Integrated data visualization using **Chart.js** to render parking utilization and user expenses.
- Added user profile management, password reset functionality, and feedback system.

3. Frameworks and Libraries Used

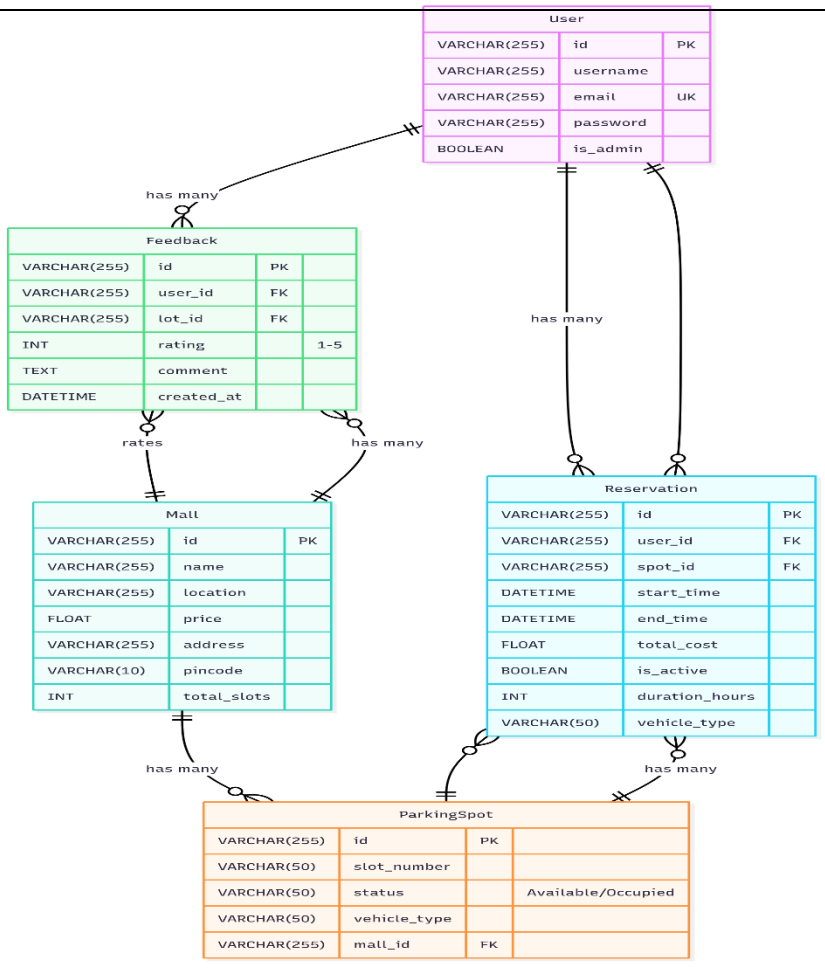
Technology	Purpose
Python	Programming language
Flask	Web framework
Flask-SQLAlchemy	ORM to interact with SQLite database
SQLite	Relational database
Jinja2	Templating engine for dynamic HTML pages
Bootstrap 5	Frontend CSS framework for styling & layout
Werkzeug	Security utilities like password hashing
Chart.js	Data visualization graphs

Entity relationships:

4. Database Design (ER Diagram)

The application database consists of these main entities:

- **User:** Stores user data, including username, email, password hash, and role (Admin/User).
- **Mall (Parking Lot):** Describes each parking location with name, address, location details, price per hour, and total slots.
- **ParkingSpot:** Represents individual spots in each lot, tracks availability and vehicle type compatibility.
- **Reservation:** Records bookings, linking users to spots, with timestamps, pricing, and status.
- **Feedback:** Allows users to rate and comment on parking lots.



5. API Endpoints

The application provides two simple API endpoints to supply real-time data for dashboards and charts. These endpoints return JSON-formatted data and are used internally by the admin and user dashboards to visualize key statistics:

- **/admin/chart_data (GET):**
  Returns parking lot utilization data—each lot's name, total spots, and number of occupied spots.
  This data is used by the admin dashboard to show bar charts of lot usage.
- **/user/chart_data (GET):**
  Returns the logged-in user's personal parking cost history.
  This enables individual users to see visualizations (e.g., line graphs) of their past parking expenses.

6. Presentation Video: Click_Here

7. Use of AI or Language Models

This project involved limited use of AI language models primarily to:

- Perplexity generates example code snippets for External and Inline CSS.
- Perplexity to assist in structuring the project and documentation.

All core design, customization, and final implementations were authored and verified independently by the student.

Summary and Conclusion

This Vehicle Parking App demonstrates a robust understanding of full-stack development principles using the Flask framework. The app encapsulates essential functionalities for real-world parking management including role-based access, dynamic resource creation, secure user authentication, reservation management, interactive dashboards, and user experience enhancements such as QR codes and feedback systems. The modular design lays a solid foundation for future expansion, such as adding notification systems, mapping features, or integrating payment gateways.