Project Report: Vehicle Parking App - V1

Author Details:
Name: HIRUTHIK SUDHAKAR
Roll No: 24F2001283
Email: 24f2001283@ds.study.iitm.ac.in

1. Setup Instructions

1. Setup Virtual Environment:
python -m venv venv

2. Activate the Environment:
- Windows: venv\Scripts\activate
- macOS/Linux: source venv/bin/activate

3. Install Dependencies:
pip install -r requirements.txt

4. Run the Application:
flask run

5. Access in Browser:
http://127.0.0.1:5000

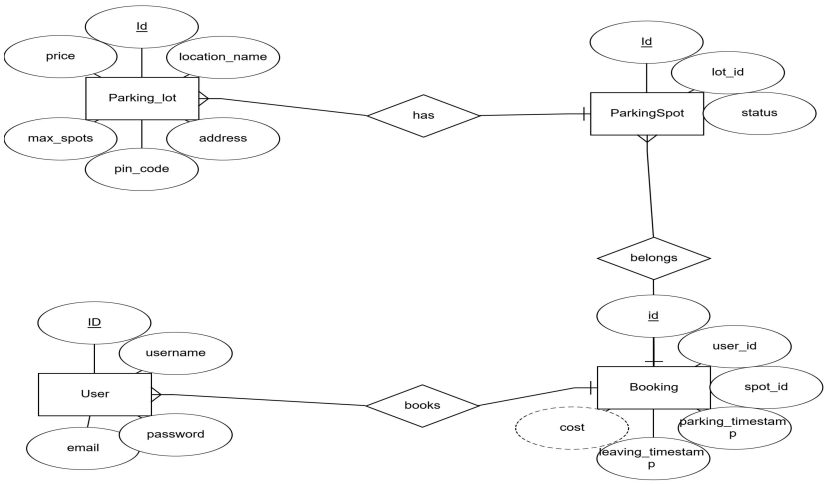
2. Project Statement

This is a multi-user web application built using Flask, designed to manage 4-wheeler parking in multiple lots. The application supports admin control over parking lots, pricing, and spot status. Users can reserve, occupy, and vacate spots with timestamp tracking. Visual summary charts are provided for both admin and user dashboards.

3. Approach and Execution

Step 1: Requirement Analysis
- Role separation: Admin (superuser), User (registration-based)
- CRUD operations for lots and automated creation of spots
- Spot status updates and parking logs
- Visual summary of bookings and occupancy
Step 2: Module-by-Module Development
- Authentication: Admin has fixed credentials; users register/login.
- Admin Dashboard: View/edit lots, auto-create parking spots, view spot usage.
- Parking Logic: Automatic assignment of the first available spot.
- Booking Management: Tracks parking in/out timestamps and costs.
- Charts: Matplotlib-based bar and pie charts for data insights.
- Template Rendering: Jinja2 templates for responsive UI.

4. Database Schema



5. API Endpoints (Controller)

Route	Methods	Description
/	GET, POST	Login page for users and admin
/admin	GET	Admin dashboard
/register	GET, POST	User registration form
/parking	GET, POST	User dashboard showing available parking lots
/add_parking	GET, POST	Admin creates a new parking lot
/edit_parking/<int:parking_id>	GET, POST	Admin edits parking lot details
/delete_parking/<int:parking_id>	GET, POST	Admin deletes a parking lot
/delete_spot/<int:spot_id>	POST	Admin deletes a parking spot
/user	GET, POST	User dashboard home
/booking/<int:spot_id>	GET, POST	User books a parking spot
/release_booking/<int:booking_id>	GET, POST	User releases a booking and calculates cost
/history	GET, POST	User views their parking history
/summary	GET	Displays summary charts for user
/admin/search	GET	Admin searches users/bookings/parking info
/admin_user	GET, POST	Admin manages all non-admin users
/edit_user/<int:user_id>	GET, POST	Admin edits user details
/delete_user/<int:user_id>	GET	Admin deletes a user
/admin/summary	GET, POST	Admin view with charts and parking lot summary
/admin/booking/<int:spot_id>	GET, POST	Admin books a spot for any user

6. Frameworks & Libraries Used

- Backend: Flask (Python)
- Frontend: HTML5, CSS, Bootstrap, Jinja2
- Database: SQLite (auto-created using SQLAlchemy)
- Libraries: Flask-SQLAlchemy, Flask-WTF, Matplotlib

7. Video Link:

https://drive.google.com/file/d/1sflHPS0V6E5TXlUPXPjgMx8Y4oelKaWs/view