Project Report
Flask-Based Vehicle Parking Application

Author
Name - Abhishek Dutta
Roll No. - 22f3001838
Email - 22f3001838@ds.study.iitm.ac.in
Course - Modern Application Development 1

Description
Vehicle Parking System is an online application that streamlines parking lot management for users and administrators. Developed with Flask and SQLAlchemy, the system provides a straightforward interface to reserve and monitor parking spaces in different places.

Technologies used
Flask: Lightweight web framework for building the application.
Werkzeug Security: For password hashing and secure authentication.
Flask-Login: For User authentication.
Jinja2: Templating engine for rendering dynamic HTML pages.
SQLite: Lightweight, file-based database for easy setup and local development.
Bootstrap: For responsive, modern, and mobile-friendly UI design.
HTML5 and CSS3: For structuring and styling frontend pages.

DB Schema Design
1. Admin:

- id: Integer, Primary Key
- username: String (max 80), Unique, Not Null
- password hash: String (max 256), Not Null

2. User:

- id: Integer, Primary Key
- username: String (max 120), Unique, Not Null
- password_hash: String (max 256), Not Null
- full_name: String (max 200), Nullable
- email: String (max 120), Nullable
- confirm_email: String (max 120), Nullable
- vehicle_number: String (max 50), Nullable
- phone: String (max 20), Nullable
- gender: String (max 20), Nullable
- postcode: String (max 20), Nullable
- communication_mode: String (max 20), Nullable
- age: Integer, Nullable
- city_id: Integer, Foreign Key to City.id, Nullable
- car_park_id: Integer, Foreign Key to ParkingLot.id, Nullable
- address: String (max 200), Nullable
- pin_code: String (max 10), Nullable
- **Relationships**: One-to-many with ReserveParkingSpot

3. City:

- id: Integer, Primary Key
- name: String (max 100), Unique, Not Null
- **Relationships**: One-to-many with ParkingLot

4. ParkingLot

- id: Integer, Primary Key
- city_id: Integer, Foreign Key to City.id, Not Null

- prime_location_name: String (max 100), Not Null
- price: Float, Not Null
- address: String (max 200), Not Null
- pin_code: String (max 10), Not Null
- maximum_number_of_spots: Integer, Not Null
- **Relationships**: One-to-many with ParkingSpot

5. ParkingSpot
   - id: Integer, Primary Key
   - lot_id: Integer, Foreign Key to ParkingLot.id, Not Null
   - status: String (1 character), Default 'A', Not Null (Values: 'A' = Available, 'O' = Occupied)
   - vehicle_number: String (max 20), Nullable

6. ReserveParkingSpot
   - id: Integer, Primary Key
   - spot_id: Integer, Foreign Key to ParkingSpot.id, Not Null
   - user_id: Integer, Foreign Key to User.id, Not Null
   - parking_timestamp: DateTime, Not Null
   - leaving_timestamp: DateTime, Nullable
   - parking_cost_per_unit: Float, Not Null
   - total_cost: Float, Nullable

API Design

The Vehicle Parking System implements a RESTful API architecture with one primary endpoint for real-time parking availability data. The API was implemented using Flask's routing system and returns JSON responses.

Features:

- Role-Based Authentication System - Secure user registration and login with separate access levels for regular users and administrators.
- Real-Time Parking Management - Users can book and release parking spots in real-time with automatic cost calculation based on duration, while administrators can manage spot availability and pricing.
- Administrative Dashboard - Complete management interface for administrators to handle user accounts, parking spot allocation, location management, and system analytics with occupancy rates and revenue tracking.
- Multi-City Location Support - Comprehensive location hierarchy with 8 cities and 9 parking lots, allowing users to select preferred locations during registration and search for available spots by area.

The project folder has app.py which is the actual flask application containing the routes, model.py which contains the database schema related definitions. The templates folder contain the html pages files and the static folder contains the css bootstrap file. The api-documentation.yaml file contains the api documentation. The requirements.txt file contains the required libraries to be installed on the virtual environment and the README.md file contains the description and instructions about how to start the flask application.

Video:

MAD1.mp4