MODERN APPLICATION DEVELOPMENT – I

VEHICLE PARKING SYSTEM- V1

AUTHOR

NAME: PRIYANSHU MITTAL

ROLL NO: 23F2002327

STUDENT EMAIL: 23f2002327@ds.study.iitm.ac.in

I am currently a student pursuing my B.Tech in DSAI from Indian Institute of Information Technology Dharwad. I have just completed my 1st year and am enthusiast about learning programming, coding, data science and artificial intelligence. This project has enhanced my development skills and has taught me new techniques which would be helpful for me in future.

DESCRIPTION

Project Statement:

Vehicle Parking App - V1

It is a multi-user app (one requires an administrator and other users) that manages different parking lots, parking spots and parked vehicles. Assume that this parking app is for 4-wheeler parking.

MY APPROACH:

As per my understanding, this project requires to design an application which ensures proper management of vehicles in parking. It should allow user to reserve the spot for his 4-wheeler, showing all his previous bookings, automatically tracking the fare and ensuring single reservation per user. For admin, it should allow to create and edit the lots each containing spots and also tracking all the users and reservations and showing all the details clearly to admin. I performed all the tasks according to milestones listed which simplified and gave direction to my project.

I have used AI/LLM in my code to implement some high-level logics, advanced python code which I am not familiar with and sometimes to enhance my app 's functionality and styling. Exact percentages are: Backend-35%, Frontend-20%, Auth+CRUD-24%, Styling+UI-6% for my application.

TECHNOLOGIES USED

Core Technologies: Python, Flask, SQLAlchemy, SQLite Flask Extensions: Flask- Bootstrap

Frontend Technologies: HTML, CSS, Jinja2, Bootstrap

Flask Bootstrap is used to manage and enhance the UI and presentation of the app. It is added so that the app looks more user-friendly.

DB SCHEMA DESIGN

USERS:

Columns	Type	Constraints
Id	Integer	Primary Key
Email	String	Unique, Not Null
Password	String	Not Null
Full_Name	String	Not Null
Address	String	
Phone	String	
Pin_Code	String(10)	

ADMINS:

Columns	Type	Constraints
Id	Integer	Primary Key
email	String	Unique, Not Null
password	String	Not Null
Full_name	String	Not Null

PARKING_LOTS:

Columns	Type	Constraints
Id	Integer	Primary Key
Name	String	Not Null
Address_Line_1	String	Not Null
Address_Line_2	String	
Address_Line_3	String	
Pin_Code	String(10)	Not Null
Price_Per_Hour	Numeric(10,2)	Not Null
Number_Of_Spots	Integer	Not Null

PARKING_SPOTS:

Columns	Type	Constraints
Id	Integer	Primary Key
Spot_Number	String	Not Null
Status	Enum	Not Null, Default=”AVAILABLE”
Parking_Lot_Id	Integer	Foreign Key → Parking_Lots.Id, Not Null

RESERVATIONS:

Columns	Type	Constraints
Id	Integer	Primary Key
User_Id	Integer	Foreign Key → Users.Id, Not Null
Parking_Spot_Id	Integer	Foreign Key → Parking_Spots.Id, Not Null
Vehicle_Number	String	Not Null
Start_Time	Datetime	Not Null, Default=Now
Occupy_Time	Datetime	
End_Time	Datetime	

Each entity is modeled as a separate table to minimize redundancy and improve data integrity. Automated spot management ensures that spot records reflect the current capacity, preventing orphaned or excess records. Use of foreign keys and enumerations prevents invalid references and enforces valid status values.

ARCHITECTURE AND FEATURES

The project follows a clear and standard structure. The main application logic resides in app.py file. All controllers are implemented in app.py. Here, each route corresponds to a specific controller function that manages user authentication, parking lot management, reservations, administrative operations, and analytics. The data models are defined in the models.py file under the models directory. This file contains ORM classes for entities such as User, Admin, ParkingLot, ParkingSpot, and Reservation. The HTML templates are organized in a templates directory, which is referenced by the Flask application's render_template function. Static assets such as CSS is stored in a static folder.

DEFAULT FEATURES

- **User Registration & Authentication**: Customers and administrators can register and log in using secure credentials. Role-based access control restricts features according to user type.
- **Parking Facility Management**: Administrators can add, edit, and delete parking lots. Each lot maintains details such as name, address, capacity, and pricing. Automated creation and management of parking spots within each lot.

- **Parking Spot Management**: Real-time tracking of each spot's status: available, reserved, or occupied. Automated adjustment of spots when lot capacity changes, ensuring spot records accurately reflect facility capacity.
- **Reservation System**: Users can view available parking lots and reserve spots. The system prevents multiple active reservations per user. Reservation status transitions from reserved to occupied to completed, with timestamps for each stage.
- **Session Tracking & Billing**: Tracks session start, occupation, and end times. Calculates parking fees based on duration and lot-specific hourly rates. Enforces a minimum billing duration and real-time cost updates.
- **Dashboards**: User dashboard displays current reservations, costs, and parking history. Admin dashboard provides system-wide statistics, including user counts, active sessions, and spot availability.
- **Comprehensive History & Analytics**: Users can review their complete parking history, including costs and durations. Administrators access detailed records of all reservations, with filtering and summary statistics.

ADDITIONAL FEATURES

- **Administrative Search**: Unified search interface allows admins to query users, lots, spots, and reservations by various attributes.
- **User Experience**: HTML templates provide a responsive and intuitive interface. Flash messages and real-time updates enhance usability and feedback.

VIDEO:

https://drive.google.com/file/d/1Z-s-181aZw-Y1ZDv3rChcLKl5PSrW-b9/view?usp=sharing