Author

Name: Vaibhav Goyal
Roll Number: DS23F1000082
Email: 23f1000082@ds.study.iitm.ac.in
I am currently pursuing the online BSc degree from IIT Madras. I am passionate about full-stack development and enjoy solving real-world problems through code.

Description

This project aims to build a smart and responsive Parking Reservation System [ParkSmart] using Flask. It enables users to search, reserve, and manage parking spots while also offering control to parking lot owners for managing availability and pricing.

Technologies Used

1 **Backend**:
- **Flask**: A lightweight Python web framework used to build the core backend logic and APIs for routing and rendering.
- **Jinja2**: Templating engine used with Flask to dynamically generate HTML pages using backend data
- **Flask Sessions**: User session management and state handling.

2 **Frontend**:

- **HTML5 & CSS3**: Used to structure and style the web pages of the website.
- **JavaScript**: Powers dynamic content updates, especially for real-time chart rendering
- **Chart.js**: A JavaScript library used to create interactive and responsive charts for visualizing analytics data.
- **Bootstrap**: Ensures a mobile-first and responsive design that adjusts to various screen sizes.

3 **Database**:

- **SQLite**: Relational databases used to persist user, booking, and parking lot data.

DB Schema Design

Tables:

1. **users**

- **Purpose: To store login credentials and user role information.**
- **Attributes:**

    - **id:** Primary key, auto-incremented unique identifier**, username:** Unique username of the user, **password:** Hashed password stored securely using Werkzeug, **is_admin:** Integer flag to indicate if the user is an admin (1) or not (0).

2. **parking_lots**

- **Purpose: To store metadata and pricing for each registered parking lot.**
- **Attributes:**

    - **id:** Primary key, auto-incremented lot ID, **prime_location_name**: Display name of the location , **price:** Parking price per unit duration, **address**: Full address of the lot,

pin_code: Pincode of the lot's location**, max_spots:** Maximum number of available parking spots.

3. **parking_spots**
    - **Purpose: To track each individual spot within a parking lot and its current availability.**
    - **Attributes:**
        - **id**: Primary key, auto-incremented spot ID**, lot_id:** Foreign key referencing parking_lots(id), **status:** Current status of the spot. Default is 'A' for available.
4. **reservations**
    - **Purpose: To track parking reservations, user IDs, and timestamps.**
    - **Attributes:**
        - **id:** Primary key, auto-incremented reservation ID, **spot_id:** Foreign key referencing parking_spots(id), **user_id:** Foreign key referencing users(id), **parking_timestamp:** Time when the parking was started, **leaving_timestamp:** Time when the vehicle left the parking spot.

API Design

APIs were implemented for user login/signup, listing parking lots, making reservations, and retrieving past bookings. Yaml file is present in zip folder.

1. **Bootstrap / CSS**
   Utilized for styling and layout purposes in the application
2. **Charts / Data Visualization**
   Used for displaying analytical data, such as user expenses, revenue statistics, through visual representations using Chart.js. This aids in better decision-making and user engagement.

Architecture and Features

Architecture:

- The project follows a Flask application factory pattern with modular organization. Controllers (routes) are defined in app/__init__.py, templates are stored in app/templates/ using Jinja2, static assets (CSS/images) are in app/static/, and the database schema is managed through schema_creation.py. The main entry point is run.py which initializes the Flask application.

- The project implements core parking management features including user authentication, parking lot search/booking, and cost calculation. Additional features include a comprehensive admin analytics dashboard with real-time charts, dynamic spot management, parking lot freeze/unfreeze functionality, and user behavior analytics. The system uses Flask routes for controllers, SQLite for data persistence, Bootstrap for responsive UI, and Chart.js for data visualization.

NOTE -> Admin Credentials -> Username – admin & password -> admin123

Video Link https://drive.google.com/file/d/19zhQuqBXlve50qddV_rJe9Q3pCCIgJJP/view?usp=sharing