Author

Naman Kumar
Roll no- 23F3001701
Email- 23f3001701@ds.study.iitm.ac.in
I am a diploma level student at IIT Madras BS degree and also a sophomore at IIT Mandi (B.Tech.) . This is my first full-stack web app project using Flask and SQLite.

Description

This project is a web-based **Vehicle Parking Management System** that manages 4-wheeler parking. Admin can create/edit parking lots, track status and view usage summaries. Users can book and release spots. The cost is calculated using reserve and release time.

Technologies used

- **Flask**: Backend application framework for routing and logic.
- **Flask-SQLAlchemy**: For creating and managing SQLite database tables programmatically.
- **HTML, CSS, Bootstrap**: For frontend design and responsive layout.
- **Jinja2**: For HTML templating with Flask.
- **SQLite**: For Database.
- **Matplotlib**: For generating visual summary charts in admin and user dashboards.
- **Time:** For recording timestamps of parking and leaving events.

DB Schema Design

1. **User**
    - **id** - primary key
    - **name, email, password** - Not Null
    - **type** - default ‘General’. (Admin is pre-created with type ‘Admin’)
    - **address, dob**
2. **Parking_Lot**
    - **id** - primary key
    - **location, price, pincode, max_spots** - Not Null
    - **address**
3. **Parking_Spot**
    - **id** - primary key
    - **status** - default ‘A’ (Available), changes to ‘O’ (Occupied) when reserved.
4. **Reserve_Spot**
    - **id -** primary key
    - **parking_timestamp, vehicle_no** - Not Null
    - **leaving_timestamp, cost -** Initially zero, updates after user releases the spot

5. **Reserved_History**
    - **id** - primary key
    - **primary_timestamp, vehicle_no, spot_id, user_id -** Not Null
    - **leaving_timestamp, cost, address**
    - **status -** initially 'B' (Booked), changes to 'R' (Released) when released

Separated live **(`Reserve_Spot`)** and completed reservations **(`Reserved_History`)** for keeping track of user history. **Parking_Lot** and **Parking_Spot** are linked 1-to-many. Similarly, (**User** and **Reserve_Spot**), (**Parking_Spot** and **Reserve_Spot**) are also linked 1-to-many,

API Design

- A few GET-based APIs are created for searching users, viewing lot and status, etc.

Architecture and Features

- **Controllers:** All route logic is in **controllers.py**
- **Models:** Defined using SQLAlchemy in **models.py**
- **Templates:** HTML files with Jinja are in **templates/** folder
- **Static:** CSS and chart images are stored in **static/**

Core Features Implemented: Admin login (no registration), admin dashboard, create/edit/delete parking lots, auto create parking spots, view spot status and user profiles, user registration/login, book and release spots with time and bill, summary charts .

Optional Features Implemented: Search ability for admin to search user by name and parking lot with its prime location,spot-level delete by admin, search ability for user to search parking lot by its pincode or location, styling using bootstrap, profile editing.

Video

VPA Demo.mp4