Vehicle Parking Project (SpotFinder)

Author

Name: Sarfaraj Alam
Roll no: 24F2007613
Email id: 24f2007613@ds.study.iitm.ac.in
Course: MAD-1 Project

Institute: IIT Madras

Description

This is a vehicle parking system project. This project is about making a website where Admin(honour) can manage his parking lots containing parking spots. Admin can fetch the details of booking of parking spots. This system also provides the users to book the parking spot from a distance according to the availability of spots. I named it SpotFinder because it makes finding spots easier.
I have used 8% of LLM in CSS, 10% in the backend, 10% in html and jinja2. I have not directly used it. I took suggestions from it and modified my code accordingly.

Technologies used

- **Flask**: A Python-based web framework that handles the core backend logic, including routing, request handling, and responses.
- **Flask-SQLAlchemy**: A Flask extension that serves as an Object-Relational Mapper (ORM), allowing Python code to interact with the database using objects instead of raw SQL.
- **SQLite**: A lightweight, file-based database engine used for storing all application data, such as user information, parking lots, and reservations.
- **Jinja2**: A templating engine integrated with Flask to dynamically generate HTML pages by inserting backend data into the HTML structure.
- **Session Management**: A built-in Flask feature that maintains a logged-in state for users by storing their information in a session.
- **Flash Messages**: Flask's messaging system, which displays one-time, temporary messages to users for events like successful registration or access denial.
- **Bootstrap 5**: A front-end framework that provides a responsive and consistent user interface through pre-designed CSS components for the navigation bar, cards, forms, and tables. The project uses a base template (`bootstrap.html`) to ensure a uniform design across all pages.

DB Schema Design

All the tables are created in the udata.sqlite3

- **`User`**: Stores user information with a unique email, name, password, and optional address. It includes boolean flags (`isUser`, `isAdmin`) and a relationship to reservations.
- **`Parking_lot`**: Contains details about each lot, including its location, price, and the maximum number of spots. It has a one-to-many relationship with `Parking_spot`, with a cascading delete.
- **`Parking_spot`**: Represents an individual spot, linking to its parent lot via a foreign key and tracking its status (`A` for available, `O` for occupied).
- **`Reserve_parking_spot`**: A junction table that manages bookings, linking a user and a spot. It records parking timestamps, leaving timestamps, and the final cost.

Architecture and Features

The project is organized with a clear separation of concerns: `app.py` contains the backend controllers (Flask routes and business logic), `models.py` defines the database schema and models (database layer), and a templates directory holds all the HTML files for the user interface (presentation layer).

The project includes core features like user registration, login, and session management. It has an admin dashboard for creating, viewing, editing, and deleting parking lots, which automatically generates spots. For users, the system allows them to reserve and release parking spots, with the cost automatically calculated.

Video

Link to your online demonstration video