Author

- Name: Harsh Ashok Pemare
- Roll_Number: 23f2001616
- Email_id: 23f2001616@ds.study.iitm.ac.in
- Introduction:

Myself HARSH ASHOK PEMARE, I am currently at diploma level. My hobbies are playing chess,reading books and playing cricket. I am pursuing another offline degree from College and currently a 3rd year student.I am very interested in Machine Learning and Artificial Intelligence.

Description

This project allows users across the globe to book the parking spot in the parking lot and the admin can use the inbuilt feature of the app to edit,create,delete parking lots.

- AI usage

I have used AI/LLM about 8% for styling and 3% for templates and 2% for routes(mainly one to two API)

Technologies used

- Python:main programming language for backend logic
- Flask:Micro web-framework for web development
- Flask-SQLAlchemy:For object relational mapping
- jinja2:For templating
- Bootstrap:For Creating tables giving headers and navbar
- Font Awesome:For Icon library and better UI
- Werkzeug:Security utilities for password hashing and checking
- HTML/CSS:For frontend and Styling
- SQLite:For Database
- Flask-Session:For session management
- dotenv: environment variable management

DB Schema Design

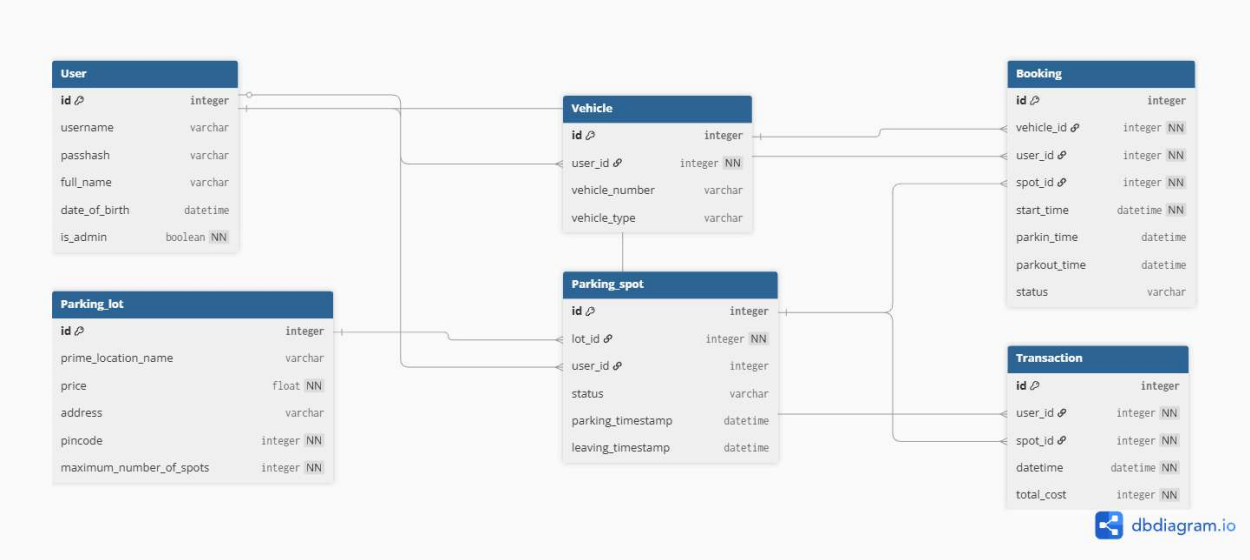
1. User Table:
    - id: Primary key (Integer)
    - username: Unique username (String, 50)
    - passhash: Hashed password (String, 256)
    - full_name, date_of_birth: Optional personal details
    - is_admin: Boolean to indicate admin role
    - Relationships: Bookings, Transactions, Vehicles, Parking Spots (1-to-many)

2. Parking_Lot Table:
    - prime_location_name: Unique location name (String, 50)
    - price: Price per unit time (Float)
    - address: Unique address (String, 200)
    - pincode: Area code (Integer)
    - maximum_number_of_spots: Maximum spots allowed
    - Relationship: Parking_spots (1-to-many)

3. Parking_spot Table:
    - id: Primary key (Integer)
    - lot_id: Foreign key to Parking_lot
    - user_id: Foreign key to User (nullable)
    - status: Current status (e.g., 'A' = Available, 'O' = Occupied)
    - parking_timestamp, leaving_timestamp: Timings for park-in/out

4. Transaction Table:
    - id: Primary key (Integer)
    - user_id: Foreign key to User
    - spot_id: Foreign key to Parking_spot
    - datetime: Timestamp of transaction
    - total_cost: Final cost calculated

5. Vehicle Table:
    - id: Primary key (Integer)
    - user_id: Foreign key to User
    - vehicle_number: Unique vehicle registration number
    - vehicle_type: 2W/4W etc.
6. Booking Table
    - vehicle_id, user_id, spot_id: Foreign keys
    - start_time: Booking made time
    - parkin_time, parkout_time: Optional actual timings
    - status: Current status of booking (e.g., Active, Complete)



API Design

User Authentication and Management:

- **Login (`/login`):**
  - **GET:** Renders the login page (`login.html`).
  - **POST:** Authenticates a user. It retrieves `username` and `password` from the form. s.
- **Registration (`/register`):**
  - **GET:** Renders the registration page (`register.html`).
  - **POST:** Handles new user registration.
- **User Profile (`/profile`):**
  - **GET:** Displays the current user's profile details (`profile.html`). Access is restricted to logged-in users via the `@login_required` decorator.
  - **POST:** Allows a logged-in user to update their profile information, including username, password, full name, and date of birth.
- **Logout (`/logout`):**
  - **GET:** Logs out the current user by removing their `user_id` from the `session`. Access is restricted to logged-in users.

Parking Lot Management (Admin Only):

- **Add Parking Lot (`/parking_lot/add`):**
  - **GET:** Renders the page to add a new parking lot (`parking_lot/add.html`).
  - **POST:** Processes the addition of a new parking lot.
- **List Parking Lots (`/parking_lot`):**
  - **GET:** Displays a list of all registered parking lots (`parking_lot/lot.html`). Restricted to admins.

- **Show Parking Lot Details (`/parking_lot/<int:lot_id>/show`):**
    - **GET:** Displays details of a specific parking lot, including its associated parking spots (`parking_lot/show.html`). Restricted to admins.
- **Edit Parking Lot (`/parking_lot/<int:id>/edit`):**
    - **GET:** Renders the edit page for a specific parking lot (`parking_lot/edit.html`).
    - **POST:** Handles updates to an existing parking lot's details, including location, capacity, price, address, and pincode.
- **Delete Parking Lot (`/parking_lot/<int:id>/delete`):**
    - **GET:** Renders a confirmation page for deleting a parking lot (`parking_lot/delete.html`).
    - **POST:** Deletes a specified parking lot from the database.

Parking Spot & Booking Management (User & Admin):

- **Admin Spot Details (`/<int:lot_id>/details/<int:spot_id>`):**
    - **GET:** Displays detailed information about a specific booked parking spot, including vehicle and user details. Restricted to admins.
- **Book Parking Spot (`/user/parking_lot/<int:lot_id>/book_spot`):**
    - **POST:** Initiates the booking process by finding an available spot in a given lot and redirects to a page (`user/spot_info.html`) to collect vehicle details.
- **Confirm Spot Info (`/<int:lot_id>/book_spot/<int:spot_id>`):**
    - **POST:** Completes the parking spot booking.
- **User Bookings (`/bookings`):**
    - **GET:** Displays all active bookings for the logged-in user (`user/bookings.html`).
- **Park-in (`/<int:lot_id>/<int:spot_id>/Parkin`):**
    - **POST:** Records the actual park-in time for a booked spot.
- **Park-out (`/<int:lot_id>/<int:spot_id>/Parkout`):**
    - **POST:** Records the actual park-out time for an occupied spot.

Transaction and History (User & Admin):

- **Transaction Calculation (`/<int:spot_id>/transaction`):**
    - **GET:** Calculates the parking cost based on `start_time`, `parkout_time`, and `price_per_hour` of the parking lot.
- **Transaction Complete (`/transaction_complete`):**
    - **POST:** A simple endpoint to confirm payment and redirect to the home page.
- **Transaction History (`/transaction_history`):**
    - **GET:** Displays the transaction history for the logged-in user (`user/transaction_history.html`).
- **Booking History (`/booking_history`):**
    - **GET:** Displays past booking records (status 'H') for the logged-in user (`user/booking_history.html`).

- **Admin Parking Records (`/parking_records`):**
    - **GET:** Displays all booking records for administrators (`parking_records.html`)

Architecture and Features

The project follows a modular Flask architecture. All route controllers and business logic are implemented in routes.py, while database models are defined in models.py. HTML templates for rendering views are organized in the templates folder, with subfolders for user, admin, parking lot, and parking spot pages. Static assets and Bootstrap are used for styling and layout.

Features Implemented:

- User Authentication: Registration, login, logout, and profile update with password change and validation.
- Admin Dashboard: Admin can view statistics, manage parking lots (add, edit, delete), and view user and parking records.
- Parking Lot Management: Admin can create new lots, set capacity and price, and manage individual spots.
- Parking Spot Booking: Users can view available lots, book spots, and associate vehicles with bookings.
- Vehicle Management: Users can register vehicles and view vehicle details for each booking.
- Parking Actions: Users can mark parking in/out, with timestamps recorded for each action.
- Transaction Management: Automatic calculation of parking fees based on duration; users can view transaction history.
- Booking History: Users can view their past bookings and transaction records.
- Flash Messages: All user actions provide feedback via flash messages for errors and successful operations.
- Session Management: User sessions are tracked for authentication and authorization.

All features are implemented using Flask routes, SQLAlchemy ORM queries, and Jinja2 templates for dynamic content rendering.

Video

https://drive.google.com/file/d/1_HLolqnc5tW60b9e5OL_nzEZpxVsSpI_/view?usp=sharing

Link to whole folder:
https://drive.google.com/drive/folders/1xX4exmeiWYdCHn0FgHHPhNmWdIb36Ml6?usp=sharing