SmartPark- Smart Parking Management System

Name – Chauhan Saumyakumar Dilipkumar

Project – Smart Parking Management System

Roll No. – 24f1000666

Student-mail – 24f1000666@ds.study.iitm.ac.in

Description

The **Smart Parking Management System** is a full-stack web application designed to streamline parking lot management for both users and administrators. This system provides a user-friendly platform for real-time spot booking, monitoring, and data-driven oversight, ensuring efficient space usage and improving the parking experience across locations.

Key Features

- **Parking Spot Reservation and Cancellation**: Users can view available spots and reserve or cancel bookings in real-time.
- **Role-Based Dashboards**: Admins and users have customized dashboards tailored to their roles. Admins manage users and parking infrastructure, while users can manage their reservations and view usage summaries.
- **Analytics and Summary Charts**: Admins can view visual summaries showing revenue generated from each parking lot, current availability, and occupancy. Users can also view summaries of their parking usage patterns.
- **Search Functionality**: Admins can search for user bookings either by user ID or by location, enhancing ease of monitoring.
- **Parking Spot Management**: Admins can add new parking spots or delete existing ones, ensuring real-time infrastructure control.
- **Secure Profile Management**: Both admins and users can update personal details and passwords with validation for secure access.

- **Login and Access Control**: The application uses secure password hashing and role-based access through Flask-Login.
- **Visual Dashboard Integration**: Summary charts use donut and bar graphs for better understanding of booking statistics and financial revenue.

Technologies used:

• HTML/CSS/Bootstrap - (Front-end)

• JavaScript - (dynamic control)

• SQLite - (Database Management)

• Flask - (Backend)

• SQLAlchemy - (SQL Library)

• Migrations - (Database Schema Version Control)

• Werkzeug utils – (File security)

• FlaskSQLAlchemy – (extension)

• Flask-Login - (Login Management)

• Werkzeug Security - (Hashing passwords)

• Logging - (Log all HTTP requests)

Database: (Ctrl + Click HERE to go to the Schema diagram)
In our database, we have six tables that store all necessary information about the parking lot application:

1. **User** – id, email, password, full_name, address, pincode, role
2. **ParkingLot** – id, owner_id, location_name, address, pincode, price, total_slots, available_slots
3. **Booking** – id, user_id, lot_id, vehicle_no, timestamp, status, spot_id
4. **ParkingSpot** – id, lot_id, spot_number, is_available
5. ***(Relationships)*** –
   • User ↔ ParkingLot (owner)
   • User ↔ Booking
   • ParkingLot ↔ ParkingSpot
   • ParkingLot ↔ Booking
   • ParkingSpot ↔ Booking

Folder structure:

Parking_app_24f1000666-

- ***app.py***: application entry point
- ***extensions.py***: external extensions (e.g., Flask extensions)
- __pycache__/: auto-generated Python cache files

- **controllers**: request handlers and business logic
  - ***admin_controller.py***: admin-related routes
  - ***auth_controller.py***: login, signup, authentication
  - ***dashboard_controller.py***: dashboard views for users/admins
  - ***decorators.py***: route protection decorators
  - ***graph_controller.py***: analytics or chart-related logic
  - ***user_controller.py***: user profile and operations
  - ***__pycache__/***: compiled controller files

- **models**: data models and ORM
  - ***models.py***: defines database models
  - ***__pycache__/***: compiled model files

- **instance**: app-specific files
  - ***parking.db***: SQLite database file

- **templates**: HTML templates for rendering frontend
  - admin_search.html
  - admin_summary.html
  - admin_users.html
  - book.html
  - dashboard_admin.html
  - dashboard_user.html
  - edit_parking_lot.html
  - edit_profile.html
  - login.html
  - new_parking_lot.html
  - release.html
  - signup.html
  - summary.html
  - view_parking.html

- **static**: static resources (images, CSS, JS)
  - background_black.png

    - car.jpg
    - profile pics.png

- **migrations**: database migration scripts (Flask-Migrate/Alembic)
    - README
    - alembic.ini
    - env.py
    - script.py.mako
    - versions/: migration version files
    - __pycache__/: compiled migration files

Video Demonstration link: (Ctrl + Click HERE)