# Parking Management System

## AUTHOR

**Name :** Rishika Sahu
**Roll no. :** 23f2004785
**Email :** 23f2004785@ds.study.iitm.ac.in
**About me :** I am deeply interested in Data Science and Machine Learning and have strong programming skills in Python. I enjoy working with various web development tools

## DESCRIPTION OF PROJECT

The Parking Management System simplifies the process of finding and reserving parking spots for four-wheelers. It allows users to manage their parking bookings, while also enabling administrators to track the entire system, including lot status and visual summaries of parking activity.

## TECHNOLOGIES USED

- **Flask:** Used for handling routes and logic.
- **SQLAlchemy:** Used to interact with the database.
- **SQLite**: Used for storing application data.
- **Matplotlib**: Used to generate charts for data visualization.
- **Pandas:** Used for data manipulation with the help of DataFrames.
- **HTML/CSS:** Used to design the user interface.
- **Jinja2:** Used for rendering dynamic HTML pages.
- **Datetime:** Used for handling date and time operations.

## DB SCHEMA DESIGN

DB schema contain tables: User, ParkingLot, ParkingSpot and Reservation. Admin and user are distinguished by role (0 = user, 1 = admin).

- **User :-**

  id → Integer, Primary Key
  email → String, Unique, Not Null
  password → String, Not Null
  full_name → String
  address → String
  pin_code → String
  role → SmallInteger (0 = user, 1 = admin)

- **ParkingLot :-**

  Id → Integer, Primary Key
  prime_location_name → String
  address → String
  pin_code → String
  city → String
  price → Float
  max_number_of_spots → Integer

- **ParkingSpot :-**

  id → Integer, Primary Key
  lot_id → Integer, Foreign Key
  spot_name → String
  status → String ('A' or 'O')

- **Reservation :-**

  id → Integer, Primary Key
  user_id → Integer, Foreign Key
  spot_id → Integer, Foreign Key
  vehicle_number → String
  parking_time → DateTime
  leaving_time → DateTime
  cost_per_unit_time → Float

### Relationships :-

**User → Reservation**: One-to-many
(A single user can have multiple reservations.)
**ParkingLot → ParkingSpot**: One-to-many
(A single parking lot can contain multiple parking spots.)
**ParkingSpot → Reservation**: One-to-many
(A parking spot can appear in many reservation records, but only one can be active at a time)

## ARCHITECTURE

- **app.py:** The main controller of the application. It handles routing, business logic, and all user/admin functionalities.
- **models.py:** Defines the database models using SQLAlchemy.
- **instance/:** Contains the SQLite database (database.db).
- **templates/:** Holds all the HTML templates for both User and Admin.
- **static/:** Contains CSS stylesheets and generated chart images.

## FEATURES

**User Management:**

- Registration and authentication for users.
- Choose a parking lot by searching for their city.
- Book the first available spot and release it after use.
- View summarized booking history with charts.
- Profile management including address and pin code update.

**Admin Management**:

- Admin account is automatically created when the system is first launched.
- Admin can manage parking lots (add, edit, delete).
- Admin can view, create, and delete parking spots.
- Admin can view application statistics and revenue charts.
- Admin can view complete user details.

**Additional Features:**

- Spot names are auto-generated based on location initials.
- Spot deletion is blocked if occupied; lot deletion is allowed only when all spots are available.

**AI/LLM USAGE**

AI is used to assist with backend logic (~25%), frontend templates (~15%), authentication and CRUD workflows (~10%). All core implementation was done manually and AI is used mainly for guidance.

Video Link

https://drive.google.com/file/d/13UmXTzG0ktTTafNoT0A0Bq-mCIefJ8gM/view?usp=sharing

# THANK YOU