Author
VAISHNABI GHOSE
23f3001025
23f3001025@ds.study.iitm.ac.in

I'm currently a student pursuing my BSc in Programming and Data Science. This is one of my projects from the MAD-1 course, and I had fun learning while building it.

Description
This project is a web-based smart parking system. It allows users to book and release parking spots, and gives special access to an admin (superuser) who can manage parking lots and view charts. I used Flask, SQLite, and Jinja2 to build everything.
I used AI tools just a bit—mainly for some HTML pages and writing Matplotlib chart code. The rest of the backend logic and app functionality were done by me.
AI/LLM Usage: ~15% — Mostly for writing HTML and plotting charts.

Technologies Used

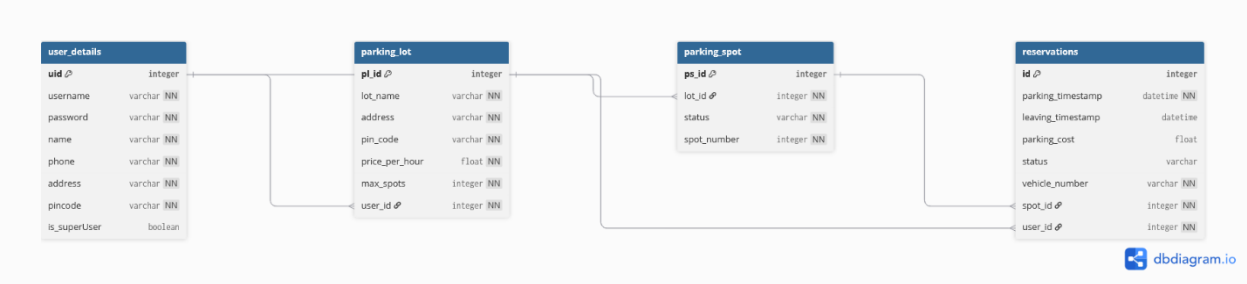
- Flask: To handle routing and backend logic
- SQLite with SQLAlchemy: For managing and storing data
- Jinja2 with HTML: For dynamic templates and frontend pages
- Matplotlib: Used to make charts showing parking stats
- Flash Messages: To give alerts and success messages to users
- AI Tools (ChatGPT): Used just a bit for writing HTML and plotting chart code

DB Schema Design
The system uses four main tables:

1. **User Table (user_details)**
    - uid: Integer, Primary Key, Auto-incremented
    - username: String, Unique, Not Null
    - password: String, Not Null
    - name: String, Not Null
    - phone: String(10), Unique, Not Null
    - address: String(255), Unique, Not Null
    - pincode: String(6), Not Null
    - is_superUser: Boolean, Default=False
2. **ParkingLot Table (parking_lot)**
    - pl_id: Integer, Primary Key, Auto-incremented

- lot_name: String(100), Unique, Not Null
- address: String(200), Unique, Not Null
- pin_code: String(10), Not Null
- price_per_hour: Float, Not Null
- max_spots: Integer, Not Null
- user_id: FK → user_details.uid

3. **ParkingSpot Table (parking_spot)**
   - ps_id: Integer, Primary Key, Auto-incremented
   - lot_id: FK → parking_lot.pl_id, Not Null
   - status: String(1), Default 'A' (Available), can be 'O' (Occupied)
   - spot_number: Integer, Not Null
4. **Reservation Table (reservations)**
   - id: Integer, Primary Key
   - parking_timestamp: DateTime, Not Null
   - leaving_timestamp: DateTime, Nullable
   - parking_cost: Float, Nullable
   - status: String (Active/Released), Default 'Active'
   - vehicle_number: String(20), Not Null
   - spot_id: FK → parking_spot.ps_id, Not Null
   - user_id: FK → user_details.uid, Not Null



Design Rationale:
The tables are designed to be clear and linked so the app knows who booked what, when, and for how much. Admins can manage parking areas easily. Spots keep track of their own availability.

API Design

Everything is in app.py—all the routes, logic, and conditions.
Flask decorators are used to define pages i.e., a **decorator** (like @app.route(...)) is used to **connect a URL (like /) to a specific function**.
No APIs from outside were used.

Architecture and Features

This project is organized like a simple Flask monolith:

- All routes and logic are in one file: app.py
- HTML files live in the templates/ folder
- Chart images are stored in the static/ folder
- The database is powered by SQLite with SQLAlchemy and Flask-Migrate

Key Features:

- Login system with user roles (normal user and superuser)
- Admin (superuser) can add, edit, and delete parking lots
- When adding a lot, spots get auto-created
- Every spot has a number and its status (Available or Occupied)
- Users can search by pin code to find lots
- Users can book a spot by entering their vehicle number
- Booking cost is based on time duration
- Users can release the spot after use
- Users can update their profile with checks in place
- Admin dashboard shows lot status (how many spots free/used)
- Admin can view all registered users
- Users can see their full booking history
- Charts (via Matplotlib) for summary

Extra Features:

- Max 4 active bookings allowed per user
- Admin can filter lots by pincode

Video Link:

https://drive.google.com/file/d/1YhsZO6idK8IiIPmSw66ea4HSArL4Qe_M/view?usp=sharing