## Author

Sharbba Sengupta
23f2002684
23f2002684@ds.study.iitm.ac.in

I am currently a Diploma level student enrolled in BS in Data Science and Applications at IITM. I am also a CSE college student at SRMIST Tiruchirappalli. Unlike regular CSE students I aspire to be a Data Scientist.

## Description

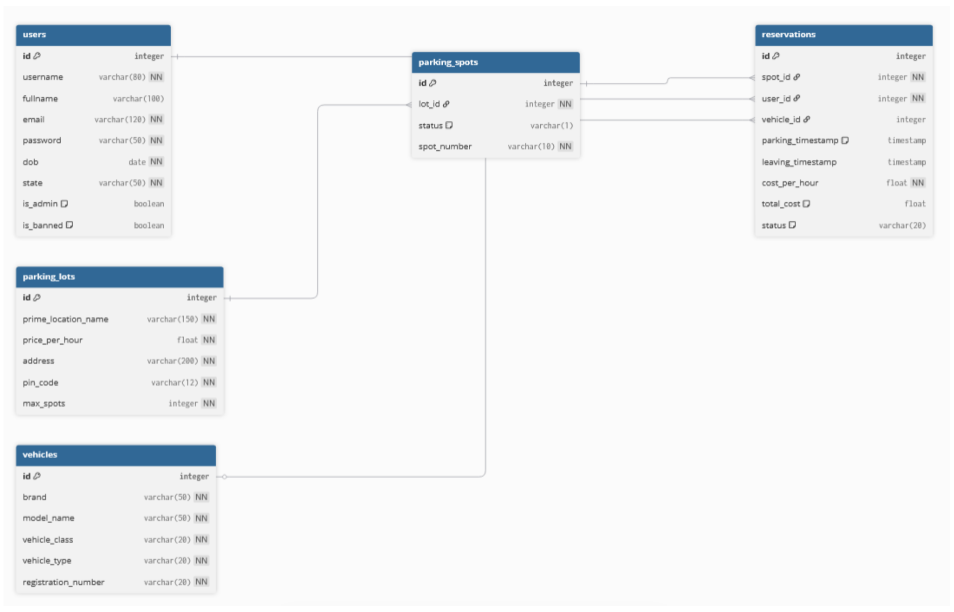
The main objective was to create a Vehicle Parking App for 4 wheelers using HTML/CSS, Flask as backend and Sqlite as the lightweight database. The main features include admin and user access to the parking lots through the booking process keeping deadlocks in mind.

AI/LLM Used percentage:
Flask(App + API)=13.6%, SQLite + SQLAlchemy=10%, HTML + jinja2=15%, CSS=5%, Javascript=5%
These values are according to best of my knowledge of AI/plagiarism checks on free platforms.

## Technologies used

1. Flask: Defining routes for the app, request handling, rendering templates and core backend logic.
2. SQLAlchemy : Defining the models, fetching data from the database and commiting the new changes to the database.
3. Jinja: For templating and dynamically fetching the values from Flask backend.
4. SQLite: A lightweight and easy to handle database.
5. Datetime: Used for formatting timestamps and getting current date and time values.
6. Wraps(from Functools): Writing custom decorators in python.
7. Check/generate_password_hash(from werkzeug.security): Hashing the password for security

## DB Schema Design



1. User class:
    a. id=user id, primary key
    b. username=string, unique, not null
    c. fullname=string, unique, not null
    d. email=string, unique, not null
    e. password=string, not null
    f. dob=date of birth, not null
    g. state=string, not null
    h. is_admin=boolean default=False
    i. is_banned=boolean, default=False

2. ParkingLot class:
    a. id=parking lot id, primary key
    b. prime_location_name=string, unique, not null
    c. price_per_hour=float, not null
    d. address=string, unique, not null
    e. pin_code= string, unique, not null
    f. max_spots= integer, not null

3. ParkingSpot class:
    a. id=parking spot id, primary key
    b. lot_id=string, not null
    c. status=string, default='A'
    d. spot_number= string, not null

4. Reservation class:
    a. id=reservation id, primary key
    b. spot_id=db.Column(db.Integer, db.ForeignKey('parking_spot.id'), nullable=False)
    c. user_id=integer, not null
    d. vehicle_id=integer, not null
    e. parking_timestamp=date_time, not null
    f. leaving_timestamp=date_time, can be null
    g. cost_per_hour=float, not null
    h. total_cost=float, default=0.0
    i. status=string, default='Pending'

5. Vehicle class:
    a. id=vehicle id, primary key
    b. brand=string, not null
    c. model_name=string, not null
    d. vehicle_class=string, not null
    e. vehicle_type=string, not null
    f. registration_number=string, not null

All fields have an id set as Primary Key. Such a design efficiently connects all the tables. The parking spot is the key central element because based on its availability and properties the rest of the tables are dependent.

## API Design

I created a RESTful API for a Vehicle Parking Web App with Flask and SQLite, including registering users, user login, making bookings, and an admin side for managing parking lots, users, and receipts. The application's routing is handled using Flask decorators; forms share URL-encoded data; templates are rendered using Jinja2. The whole API documentation was built using the OpenAPI (YAML) specification.

## Architecture and Features

Architecture:
Templates folder contains all the html files with inline css. All webpages have been designed with jinja2 are stored here. Static folder contains the images used in the frontend. The instance folder has the database defined in sqlite. Database models that are used are defined in the models.py file. The main application logic, including route controllers that handle incoming HTTP requests and business processes, resides in the app.py file.

Features:

1. In the admin routes section the admin is defined with a unique id and password.
2. Initially the admin logs in and creates atleast one parking lot with details such as prime location name, price/hr, address, maximum spots allocated etc.
3. The first user needs to Register with fullname, email, dob, state and a username-password combination of their choice.
4. Admin:
   a. Can create a lot and assign a number of spots to it and a rate.
   b. Can edit an existing lot or delete a lot that has no spots occupied.
   c. Can ban/unban users. Most common reason could be fraud/pending payments.
   d. Can view a bar chart of how many spots are occupied for each lot.
   e. Also generate receipts for each booking.
5. User:
   a. A user can only login with username and password if the account exitsts or is not banned.
   b. Can see their previous bookings on their dashboard.
   c. Bookings are also expressed by lot in form of pie-chart.
   d. User can book a spot and also view details about that lot first.
   e. Vehicle information is absolutely compulsory during the booking process along with the timings and the date.
   f. A line chart shows how much the user has been spending on bookings month wise.
   g. User can update their profile details (except password).
   h. A settings section is added for dark mode, email alerts, help queries, logout and delete account features.

## Video

https://drive.google.com/file/d/1k1J0w-xU6TWhtDgnQdQiz0r5T9fI7Klj/view?usp=sharing