Author

Siddhant Ramesh Thakar

Roll No. - 23f3001451

Email – 23f3001451@ds.study.iitm.ac.in

This is Siddhant and I'm a junior year student at Pune Institute Of Computer Technology and am passionate about cybersecurity. I aspire to be the best cybersecurity architect.

Description :

This is a Flask-based vehicle parking web app with two types of users that is a customer and an admin. It is a simple web application in which the user can book a slot from a lot and the admin can create lots as per required.

Technologies used :

Extensions :

Flask

Flask-SQLAlchemy

Jinja2

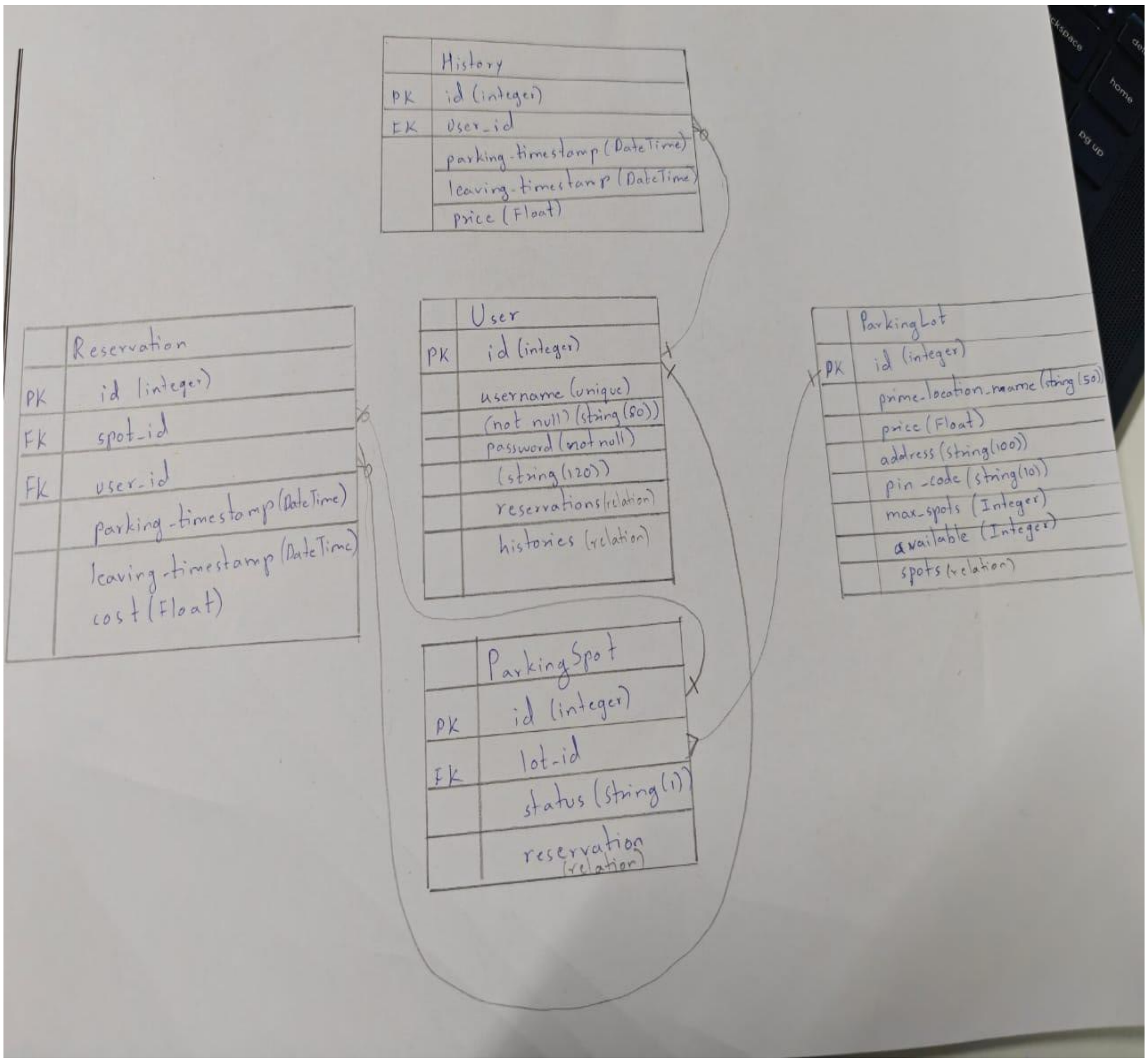
Purpose :

Flask : For designing UI and dynamic routing.

Flask-SQLAlchemy: Used for setting up databases locally using SQLite.

Jinja2 – For templating and easy iterations.

DB Schema Design



Design logic:

- A user can have multiple reservations and a reservation can have a single user so there is a one to zero or many relationship between user and reservation.

- A user can have multiple histories and a history can have a single user so there is a one to zero or many relationship between user and history.

- A Parkinglot can have multiple parkingspots but a parkingspot can have a single parking lot so they have a one to many relationship between them.

- A parking spot can be reserved once at a time so there is a one to one relationship between parking spot and reservation.

API Design

- GET / - Loads Home page.
- GET/POST/register - Registers a new user and redirects to login after registering.
- GET/POST/login - Used for user login via credentials either for user or admin.
- GET/logout – Logging the user out and redirecting to login page.
- GET/user/dashboard – Loads user dashboard with current bookings and history.
- GET/user/book/<int:lot_id> - Used for booking a slot for the user.
- GET/user/release/<int:res_id> - Used to release the occupied slot.
- GET/admin/dashboard – Displays admin dashboard with option to create/edit/delete lots and list of users.
- GET/POST/admin/create_lot – Used for creating a new lot with max_spots.
- POST/admin/delete_lot/<int:lot_id> - Used for deleting a parking lot using its it.
- GET/POST/admin/edit_lot/<int:lot_id> - Used for updating the max_spots in a lot.

Architecture and Features

The root folder has static folder and templates folder. All the html code files are present in the templates folder. The css stylesheet is located in the static folder. The backend logic along with the schemas and controllers is entirely located in the app.py python file.

Features :

The Admin superuser is already present in the User table.

The username is admin and password is admin123. The admin can add a parking lot by filling in the necessary details.

A normal user can book a spot from the lot that was created by admin.

The admin can edit the maximum slots if the new maximum slots are greater than or equal to occupied slots.

The admin can delete a lot if the lot is entirely empty.

The admin has access to the lot information that is the maximum spots in respective lots and the available spots as well.

The admin can also see the users that have been registered in the app.

The user can see the available lots and his/her User history.

Video link :

https://drive.google.com/file/d/10zWfZ_AmRsx5apJdGXqyyPfqQmYTX2Cx/view?usp=sharing

AI/LLM use –

I had to use it for debugging my code when I was stuck like when I was creating the edit_lot route for editing the maximum number of spots for my admin dashboard the key max_spots was not getting initialized in the session so I had to check for that particular part why it wasn't working.

I did not copy the llm code but designed my code by referring to the mistake.

For defining one-to-one relationship I had to refer to SQLAlchemy official documentation as well as some llm for the uselist attribute.

Apart from that I took a few templates from the official flask documentation for the initial setup.

For receiving the flash messages I took the template from the official flask documentation.

The logic part was done by me entirely and most of the coding part as well.