Vehicle Parking App - V1 – Project Report

Author:

Name: Samprikta Malik

Roll Number: 21f3002038

Email: 21f3002038@ds.study.iitm.ac.in

Github: samprikta1507/vehicle_parking_app_v1

Description:

This is a simple and smart web app made to help manage car parking in a better and more organized way. It's built for two types of people – admin , who manage the parking system and users , who want to park their cars.

Admin –

- Can create , edit or delete parking lots . Spots are auto created based on the total number set for the lot.
- Can see spot status and view details of user and vehicle details if a spot is occupied.
- Can see all registered users and their details and summary charts.

Users –

- Can pick a parking lot – the system auto-assigns the first available spot.
- When parked , user marks the spot as occupied . When leaving , user marks it as released.
- The app records the time between parking in and out and calculate the cost.

Technologies:

Backend:

1. Flask: web framework for python used to build the backend of the application.
2. Flask-Login: Manages users sessions and authentication.
3. SOLAlchemy: ORM(Object Relation Mapping) to interact with the relational databases.
4. Flask-SQLAlchemy: Integrates SQLAlchemy into Flask for DB operations.
5. Flask-RESTful: Flask extention to build RESTful APIs.
6. SQLite: Embedded SQL database used via SQLAlchemy.

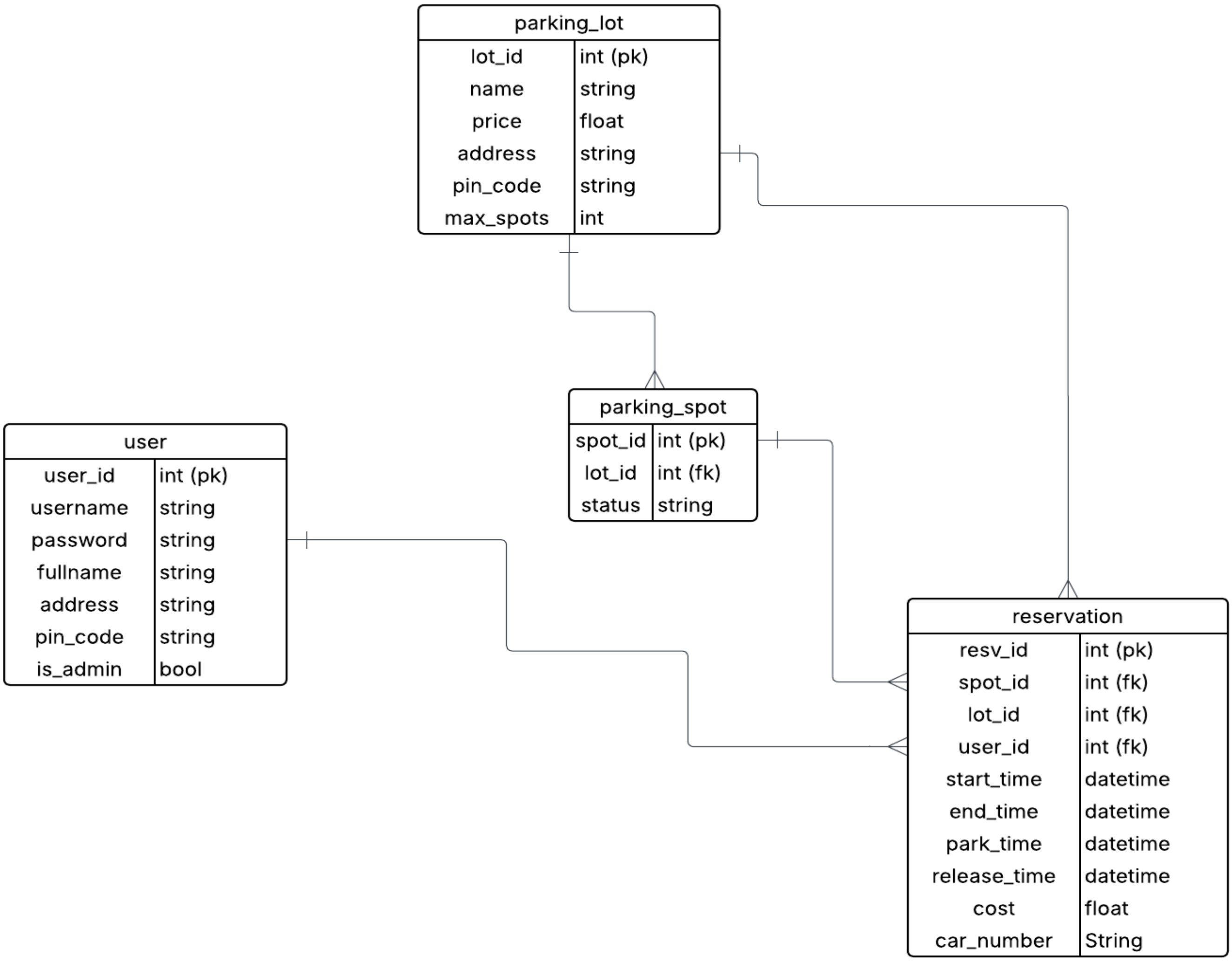
Frontend:

1. HTML5: Structure and markup of web pages.
2. CSS: Styling and layout.
3. Bootstrap: CSS framework for responsive design.
4. Jinja2: Template engine for python.
5. Chart js: for summary charts

DB Schema Design:

1. User:
    - stores user info like email, password, name, address and pin code.
    - Each user can make many reservations (ont to many relation).
2. Parking_lot:
    - Stores details of each parking lot : name,address,pin code,price and max number of spots.
    - A parking lot can have: i) many parking spots. ii) many reservations.
3. Parking_spot:
    - Represents a single spot in a parking lot.
    - Has a status(available or occupied)
    - A spot can be in many reservation over time.
4. Reservation:
    - Keeps track of a user's booking.
    - Stores: start and end times, actual parking time and leaving time, cost and car number.
    - Linked to : user,parking lot and parking spot.

ER-Diagram:



parking_lot	
lot_id	int (pk)
name	string
price	float
address	string
pin_code	string
max_spots	int

parking_spot	
spot_id	int (pk)
lot_id	int (fk)
status	string

user	
user_id	int (pk)
username	string
password	string
fullname	string
address	string
pin_code	string
is_admin	bool

reservation	
resv_id	int (pk)
spot_id	int (fk)
lot_id	int (fk)
user_id	int (fk)
start_time	datetime
end_time	datetime
park_time	datetime
release_time	datetime
cost	float
car_number	String

API Design:

1. **Parking lot operations:**
   - **Get all lot information: /api/get_lots**
   - **Add a parking lot: /api/add_lot**
   - **Edit a parking lot: /api/edit_lot/<int:id>**
   - **Delete a parking lot: /api/delete_lot/<int:id>**
   - **Search a parking lot: /api/search_lot/<int:id>**
2. **Parking spot operations:**
   - **Get all spot information: /api/get_spots**
   - **Add a parking spot: /api/add_spot**
   - **Edit a parking spot: /api/edit_spot/<int:id>**
   - **Delete a parking spot: /api/delete_spot/<int:id>**
   - **Search a parking spot: /api/search_spot/<int:id>**
3. **Reservation operations:**
   - **Get all reservation information: /api/get_reservations**
   - **Add a reservation: /api/add_reservation**
   - **Delete a reservation: /api/delete_reservation/<int:id>**
   - **Search a reservation: /api/search_reservation/<int:id>**

Architecture and Features:

1. backend/:
   - models.py: contains the database models (User,Parking_lot,Parking_spot,Reservation) using SQLAlchemy.
   - controllers.py: manages frontend-related route handling, rendering HTML pages.
   - api_controllers.py: contains api endpoints.
2. instance/:
   - holds the SQLite database file: vehicle_parking.sqlite3.
3. static/:
   - images/: contains background images used across different pages.
   - styles/: contains CSS files that style various HTML pages.
4. templates/:
   - all the HTML pages are placed here.
5. venv:
   - contains virtual environment files.
6. .gitignore:
   - Contains files to exclude from git.
7. app.py:
   - this is the main entry point of the app. It initializes the Flask app, configures the database, sets up login functionality and registers the routes and APIs.
8. Requirements.txt:
   - Lists all the python dependencies used.

Core Features:

1. Login System:
   - There are two types of logins: Admin and User.
   - Users can register and then login.
   - Admin can login (no need to register)
2. Admin Dashboard:
   - Add,edit or delete parking lots.
   - When adding a lot , the spots are added automatically.
   - Can view which spot is available and which is occupied. If occupied can view the details of user and vehicle .
   - Can view user details.
   - Can view history of parking and summary charts.
3. User Dashboard:
   - Choose a parking lot , system will assign the first empty spot.
   - Mark the spot occupied when they park their car.
   - Mark it as released when releasing the spot.
   - System will record the time they parked in and out and calculate the cost.
   - View summary charts showing their parking history.

Additional Features:

1. Edit profile: both admin and user can edit their profiles.
2. Search:
   - Admin: admin can search by user id , location and parking lot name .
   - User: user can search by location.

Video:

https://drive.google.com/file/d/1aX8iS-vH60iWdXIUZskpNb77_Nv9OBo_/view?usp=sharing