Vehicle Parking App

PARKPAL

Project Report by Aqdas Khan

23f2001156

Modern Application Development – II

May, 2025 Term

Student details

Name – Aqdas Khan

Roll no. - 23f2001156

Email - 23f2001156@ds.study.iitm.ac.in

About me - I'm a passionate and detail-oriented software developer with a focus on building efficient, scalable, and user-centric web applications. I enjoy solving realworld problems using technology and have hands-on experience with Flask, SQL, and frontend tools like Vue.js and Bootstrap. Currently pursuing my BS in Data Science and Programming from IIT Madras, I'm always looking to learn, collaborate, and grow in dynamic environments

Project Description

ParkPal is a web-based vehicle parking management system that streamlines parking spot reservations and administration. Users can register, book, and manage parking spots, while admins oversee lots, monitor occupancy, and send notifications. With real-time dashboards, automated reminders, and efficient caching, ParkPal offers a modern, user-friendly solution for both parking lot operators and vehicle owners.

Technologies Used

Flask: Lightweight Python web framework to handle the backend API.

Vue Js: JavaScript framework for building the user interface.

Bootstrap: for CSS and styling of the application user interface.

Flask-SQLAlchemy: ORM for managing database interactions.

Flask-Restful: for creating RESTful APIs.

Flask-Security: for role-based access control and handling authentication.

Flask-Caching: Adds caching for improved performance.

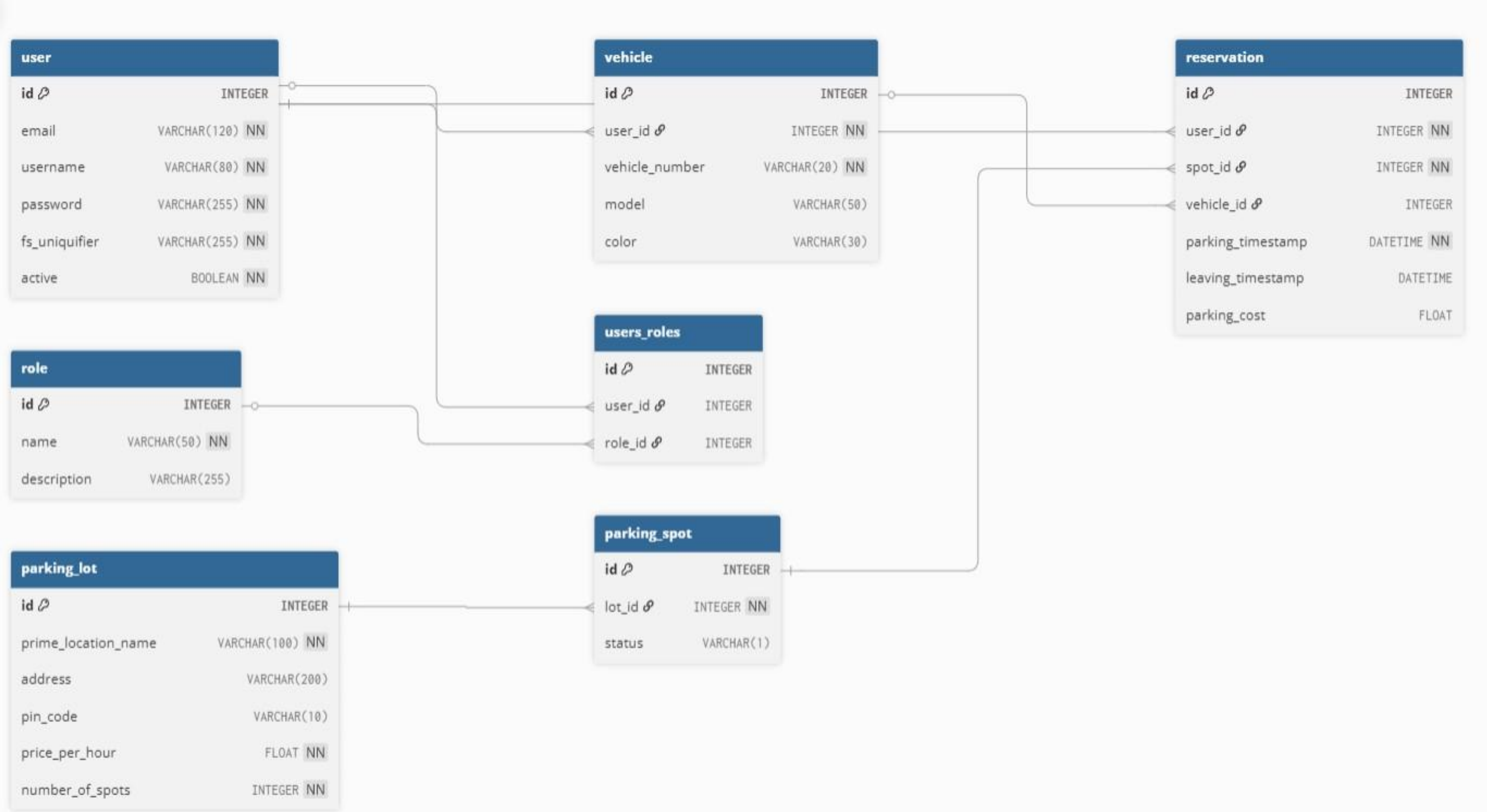
Celery: Asynchronous task queue for background jobs like reminders and monthly reports.

Redis: Caching layer and supports background jobs.

Chartjs: for plotting graphs and charts in summary section of admin and user

DB Schema Design

The ParkPal database schema is designed to efficiently manage parking operations. It includes tables for users, roles, parking lots, parking spots, vehicles, and reservations. Users can have different roles and register their vehicles. Each parking lot contains multiple parking spots, and reservations link users and their vehicles to specific spots for defined time periods. This structure enables seamless tracking of spot availability, user activity, and booking history.



user	
id	INTEGER
email	VARCHAR(120) NN
username	VARCHAR(80) NN
password	VARCHAR(255) NN
fs_uniquifier	VARCHAR(255) NN
active	BOOLEAN NN

vehicle	
id	INTEGER
user_id	INTEGER NN
vehicle_number	VARCHAR(20) NN
model	VARCHAR(50)
color	VARCHAR(30)

reservation	
id	INTEGER
user_id	INTEGER NN
spot_id	INTEGER NN
vehicle_id	INTEGER
parking_timestamp	DATETIME NN
leaving_timestamp	DATETIME
parking_cost	FLOAT

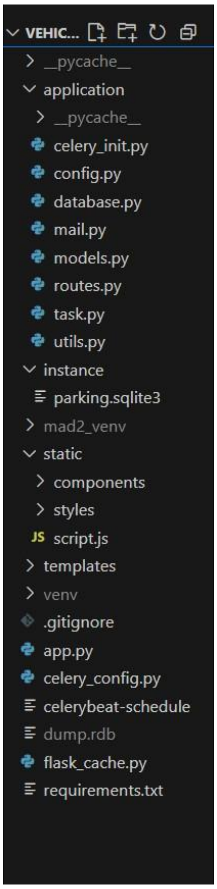
role	
id	INTEGER
name	VARCHAR(50) NN
description	VARCHAR(255)

users_roles	
id	INTEGER
user_id	INTEGER
role_id	INTEGER

parking_lot	
id	INTEGER
prime_location_name	VARCHAR(100) NN
address	VARCHAR(200)
pin_code	VARCHAR(10)
price_per_hour	FLOAT NN
number_of_spots	INTEGER NN

parking_spot	
id	INTEGER
lot_id	INTEGER NN
status	VARCHAR(1)

Directory Structure



```
VEHIC...
  > __pycache__
  v application
    > __pycache__
    celery_init.py
    config.py
    database.py
    mail.py
    models.py
    routes.py
    task.py
    utils.py
  v instance
    parking.sqlite3
  > mad2_venv
  v static
    > components
    > styles
    script.js
  > templates
  > venv
  .gitignore
  app.py
  celery_config.py
  celerybeat-schedule
  dump.rdb
  flask_cache.py
  requirements.txt
```

API Endpoints

Method	Endpoint	Description	Access
GET	/	Render home page	Public
POST	/api/register	Register a new user	Public
POST	/api/login	Login and get token	Public
GET	/api/profile	Get logged-in user profile	User/Admin
GET	/api/export	Trigger full CSV report	Admin
GET	/api/csv_result/<id>	Download CSV report	Admin
GET	/api/user/export	Trigger user-specific CSV report	User
GET	/api/user/csv_result/<id>	Download user's CSV report	User
GET	/api/admin/summary	Get admin dashboard summary	Admin
GET	/api/admin/parking-lots	List all parking lots	Admin
POST	/api/admin/parking-lots	Create a new parking lot	Admin
PUT	/api/admin/parking-lots/<int:lot_id>	Update a parking lot	Admin
DELETE	/api/admin/parking-lots/<int:lot_id>	Delete a parking lot	Admin
GET	/api/admin/parking-lots/<int:lot_id>/spots	Get all spots in a lot	Admin
GET	/api/admin/users	Get all users with reservation counts	Admin
POST	/api/admin/users/<int:user_id>/block	Block a user	Admin
POST	/api/admin/users/<int:user_id>/unblock	Unblock a user	Admin
GET	/api/admin/vehicles	Get all registered vehicles	Admin

GET	/api/user/history	User parking history	User
GET	/api/user/lots	List lots for user	User
POST	/api/user/book	Book a parking spot	User
POST	/api/user/release	Release a parking spot	User
GET	/api/user/vehicles	List user's vehicles	User
POST	/api/user/vehicles	Add a vehicle	User
DELETE	/api/user/vehicles/<int:vehicle_id>	Delete a vehicle	User
GET	/api/test-cache	Test cache endpoint	Public

Project Video

https://drive.google.com/file/d/1WwmnwIn3ZgUCW-x9wJFVFyq8QUHle5c3/view?usp=sharing