# Project Report – ParkEase Vehicle Parking V2

# Author

Raghvi Gupta | [23f3001927@ds.study.iitm.ac.in](mailto:23f3001927@ds.study.iitm.ac.in)

IITM BS in Data Science and Applications Diploma Level Student

# Introduction & Description

ParkEase is a multi-user web application designed to manage 4-wheeler parking lots, parking spots, and vehicle reservations. It supports admin-level lot management and user-level parking spot bookings with real-time status tracking. Built using Flask, VueJS, SQLite, Redis, and Celery, it includes authentication, reports, and performance optimizations.

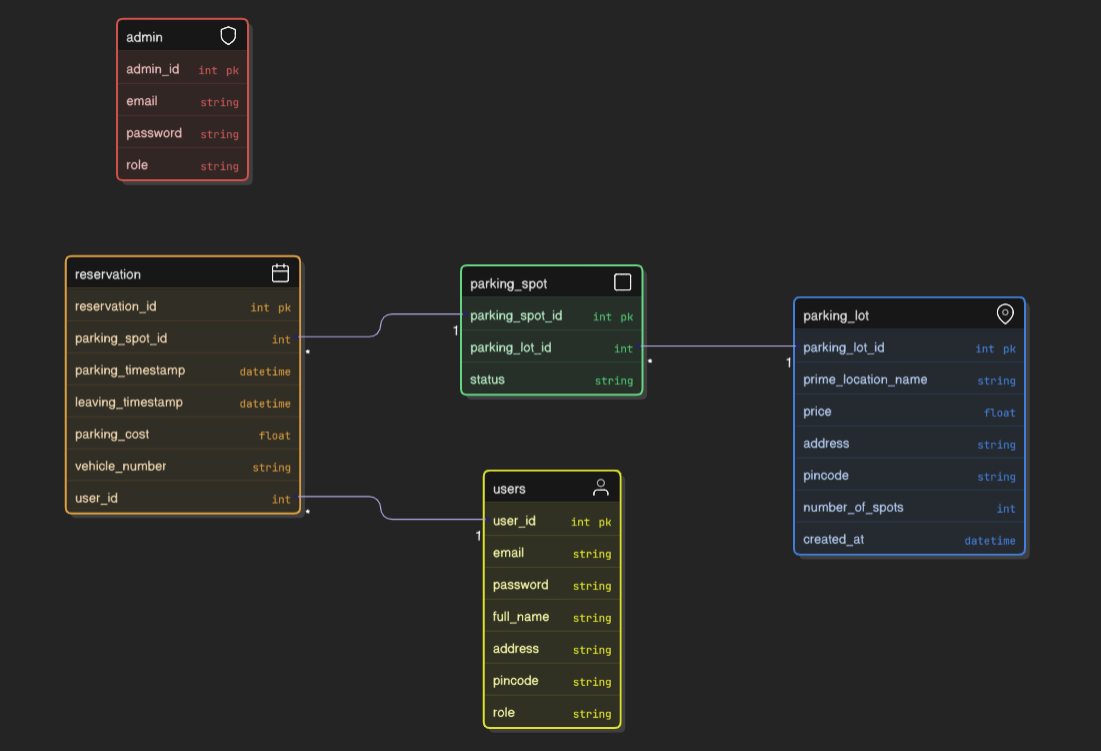
# Technologies Used

* **Backend:** Flask, Flask-SQLAlchemy, SQLite, Redis, Celery
* **Frontend:** VueJS, Bootstrap, Chart.js
* **Templates:** Jinja2

# Database Schema

The database efficiently manages all entities while ensuring data integrity through normalization and foreign keys.

* **User**: registered user details including email, password, personal info, and role, and links to reservations.
* **Admin**: Represents the system’s single superuser with privileged access to manage parking lots and users.
* **ParkingLot**: a parking lot with its location, price, number of spots, and creation timestamp.
* **ParkingSpot**: individual parking spaces within a lot and tracks their occupancy status.
* **Reservation**: user bookings with timestamps, cost, and vehicle information, linking users to spots.



# API Design

**Common routes**

@app.route('/')

@app.route('/api/login', methods=['POST'])

@app.route('/api/logout', methods=['POST'])

**Admin routes**

@app.route('/api/clear-lot-cache', methods=['GET'])

@app.route('/api/get\_all\_parking\_lots', methods=['GET'])

@app.route('/api/add\_parking\_lot', methods=['POST'])

@app.route('/api/get\_parking\_lot/<int:parking\_lot\_id>', methods=['GET'])

@app.route('/api/update\_parking\_lot/<int:parking\_lot\_id>', methods=['PUT'])

@app.route('/api/admin\_delete\_service/<int:id>', methods=['DELETE'])

@app.route('/api/admin\_users', methods=['GET'])

@app.route('/api/admin\_search', methods=['GET'])

@app.route('/api/admin\_stats/revenue', methods=['GET'])

@app.route('/api/admin\_stats/occupancy', methods=['GET'])

@app.route('/api/admin/profile', methods=['GET'])

@app.route('/api/admin/update\_profile', methods=['PUT'])

@app.route('/api/get\_spots/<int:lot\_id>/<int:spot\_id>', methods=['GET'])

@app.route('/api/spots/<int:lot\_id>/<int:spot\_id>', methods=['DELETE'])

@app.route('/api/get\_spots\_occu/<int:lot\_id>/<int:spot\_id>', methods=['GET'])

**Customer routes**

@app.route('/api/customer\_register', methods=['POST'])

@app.route('/api/parking/history', methods=['GET'])

@app.route('/api/parking/search', methods=['GET'])

@app.route('/api/reserve', methods=['POST'])

@app.route('/api/reservation/active', methods=['GET'])

@app.route('/api/reservation/release', methods=['POST'])

@app.route('/api/customer/profile', methods=['GET'])

@app.route('/api/customer/update\_profile', methods=['PUT'])

@app.route('/api/parking-usage-summary', methods=['GET'])

**Daily reminder route**

@app.route('/daily-reminder', methods=['GET'])

**Monthly reminder route**

@app.route('/monthly-report', methods=['GET'])

**CSV export routes**

@app.route('/api/trigger-export', methods=['POST'])

@app.route('/download/<filename>')

@app.route('/api/export-status/<task\_id>', methods=['GET'])

# Architecture and Features

# Modular Architecture with separate layers for API (Flask), frontend (VueJS), database (SQLite), caching (Redis), and asynchronous task processing (Celery).

# Role-Based Access Control with session-based authentication, distinguishing between admin and user functionalities.

# Key Features include auto-allocation of parking spots, real-time status updates, background notifications, and usage analytics through scheduled reports.

# Implementation Details

# Developed a RESTful backend using Flask with SQLAlchemy ORM to manage users, parking lots, spots, and reservations.

# Integrated VueJS and Bootstrap on the frontend for user/admin dashboards with dynamic parking status and booking interfaces.

# Used Redis and Celery for background jobs like daily booking reminders, monthly activity reports, and CSV exports.

# Conclusion

ParkEase is a structured, feature-rich quiz platform optimized for scalability and usability.

# Video