Application Development - 2 Project Report

Vehicle Parking App

Student Details:

Name: Devojyoti Misra
Roll No.: DS24F1002239
E-mail: 24f1002239@ds.study.iitm.ac.in

About Me: I am currently majoring in Data Science and Applications at IIT Madras. I am very enthusiastic about Machine Learning, Deep Learning, Artificial Intelligence and Data Science.

Project Details:

Project Title: Parkify

Problem Statement: To design and develop a web-based parking management solution that solves the inefficiency of urban four-wheeler parking by allowing users to find and book spots in real-time, while enabling administrators to effectively manage parking inventory, monitor occupancy, and track revenue.

Approach: The application was architected using Flask for a scalable backend and Vue.js for a reactive user interface. It implements a full-featured booking system with role-based access, utilises background workers (Celery) for automated email notifications and gamified monthly reports, and leverages interactive dashboards to provide actionable insights into parking utilisation and user activity.

AI/LLM Declaration:

I used **Gemini-3.0-Flash** primarily for quick syntax references and formatting documentation strings. The extent of AI/LLM usage is minimal (**approx. 5%**), limited strictly to basic boilerplate generation. All implementation logic, debugging, and integration were done manually.

Technologies Used:

Technology/Library	Purpose
Flask	Python backend framework
Flask-Restful	Simplifies REST API development
Flask-JWT-Extended	Handles JWT-based authentication
Flask-Caching	optimised response times via caching
VueJS	Frontend UI framework
Vite	Next-generation frontend tooling and bundler
SQLAlchemy	ORM for database abstraction
SQLite3	Primary database storage
CSS + Bootstrap5	UI design and responsive styling
Werkzeug	Security utility for password hashing
ChartJS	Data visualisation and charting
Celery	Handles background tasks asynchronously
Redis	High-performance message broker and cache store

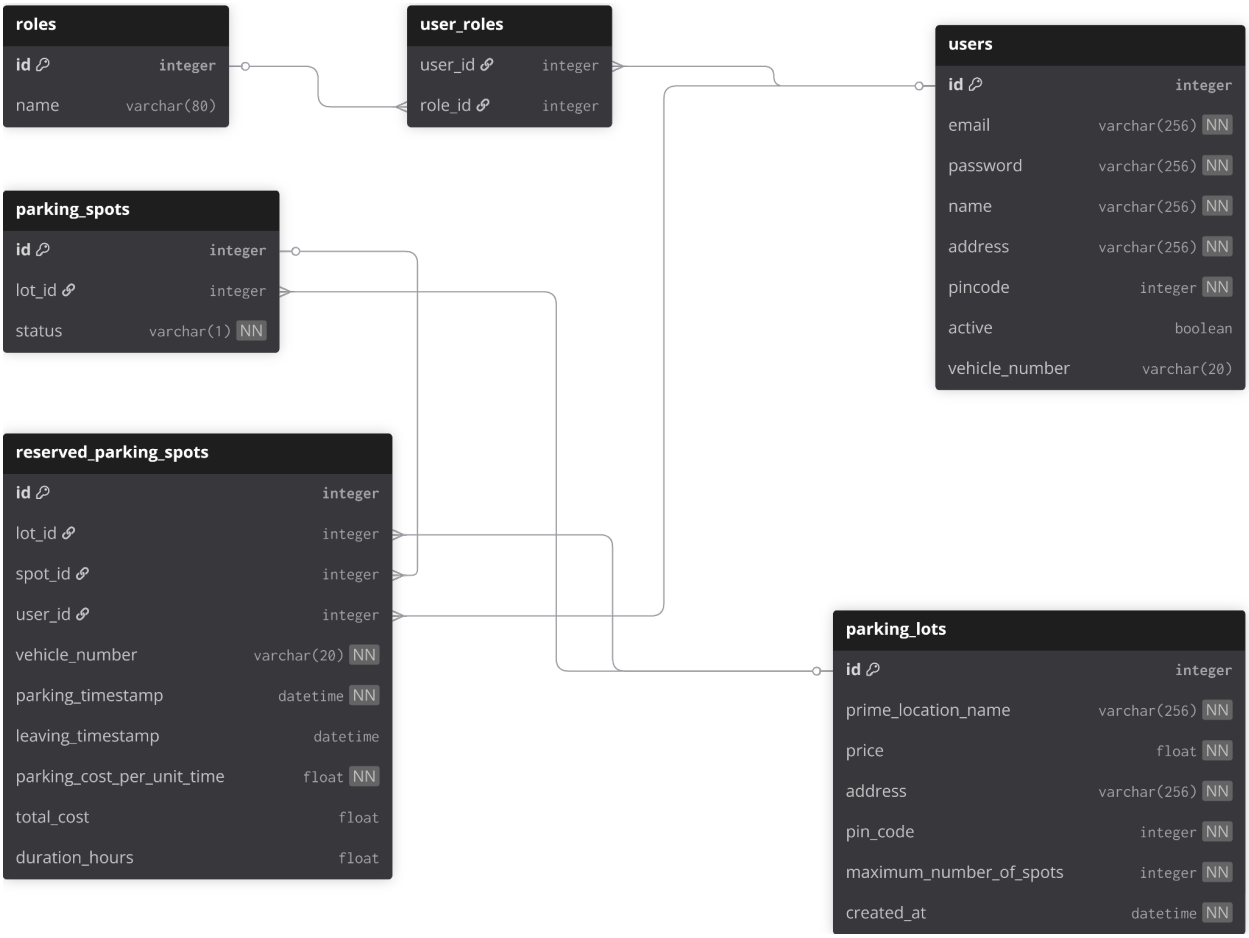
Database Schema / ER Diagram:

Tables:

1. **User** - stores user profile and authentication details (id, email, password, name, address, pincode, active, vehicle_number)
2. **Role** - defines user permissions and roles (id, name)
3. **ParkingLot** - manages parking lot locations and capacity (id, prime_location_name, price, address, pin_code, maximum_number_of_spots, created_at)
4. **ParkingSpot** - tracks individual spots within a lot (id, lot_id, status)
5. **ReserveParkingSpot** - logs all parking transactions and history (id, lot_id, spot_id, user_id, vehicle_number, parking_timestamp, leaving_timestamp, total_cost)

Relationships:

1. Many-to-Many → User ↔ Role (via user_roles association table)
2. One-to-Many → ParkingLot → ParkingSpot
3. One-to-Many → User → ReserveParkingSpot
4. One-to-Many → ParkingSpot → ReserveParkingSpot
5. One-to-Many → ParkingLot → ReserveParkingSpot



API Resource Endpoints:

Endpoint	Method	Description
/api/login	POST	Authenticates user credentials and returns JWT access token
/api/signup	POST	Registers a new user with email and password
/api/admin/dash	GET	Retrieves the admin dashboard data, including all parking lots and their status
/api/admin/add_lot	POST	Creates a new parking lot with capacity and pricing details
/api/admin/edit_lot/<int:lot_id>	PUT	Updates details of an existing parking lot
/api/admin/delete_lot/<int:lot_id>	DELETE	Deletes a parking lot if it has no active reservations
/api/admin/view_spots/<int:lot_id>	GET	Retrieve the status of all spots in a specific lot
/api/admin/spot_details/<int:spot_id>	GET	Retrieves detailed information about an occupied spot
/api/admin/users	GET	Retrieves a list of all registered users
/api/admin/search	POST	Performs a search for lots
/api/admin/history	GET	Retrieves global booking history and revenue analytics
/api/admin/profile	GET	Retrieves the current admin's profile information
/api/admin/edit_profile	PUT	Updates the admin's profile details and password
/api/user/delete_account	DELETE	Deletes the user's account if no active reservations exist
/api/user/dash	GET	Retrieves the user dashboard with active reservations and available lots
/api/user/select_slot/<int:lot_id>	GET	Retrieves the layout and availability of spots for a lot
/api/user/book_slot/<int:lot_id>	POST	Book a specific parking spot for the user
/api/user/release_slot/<int:reservation_id>	POST	Releases a parking spot and calculates the final cost
/api/user/search	POST	Searches for parking lots matching a location query
/api/user/available_lots	GET	Retrieves a list of all parking lots with available spots
/api/user/history	GET	Retrieves the user's past booking history
/api/user/profile	GET	Retrieves the current user's profile information
/api/user/edit_profile	PUT	Updates the user's profile details and password
/api/user/export	GET	Exports the user's booking history as a CSV file

Architecture Overview (Project Directory Structure):

```
root/.
|   .gitignore
|   MAD-2 Project Report.pdf
|   README.md
+---Backend
|   |   .env
|   |   .env.example
|   |   api.yaml
|   |   app.py
|   |   pyproject.toml
|   |   uv.lock
|   +---application
|   |   +---extensions
|   |   |       cache_extn.py
|   |   |       db_extn.py
|   |   |       jwt_extn.py
|   |   +---helpers
|   |   |       cache_utils.py
|   |   |       config.py
|   |   |       decorators.py
|   |   |       models.py
|   |   |       tasks.py
|   |   |       validators.py
|   |   +---middlewares
|   |   |       init_app_context.py
|   |   |       init_bg_jobs.py
|   |   |       init_cache.py
|   |   |       init_celery.py
|   |   |       init_config.py
|   |   |       init_db.py
|   |   |       init_jwt.py
|   |   |       init_token_refresh.py
|   |   \---resources
|   |       +---admin
|   |       |       admin_dashboard_resource.py
|   |       |       admin_history_resource.py
|   |       |       admin_parking_lot_add_resource.py
|   |       |       admin_parking_lot_delete_resource.py
|   |       |       admin_parking_lot_update_resource.py
|   |       |       admin_parking_spots_details_resource.py
|   |       |       admin_parking_spot_details_resource.py
|   |       |       admin_profile_fetch_resource.py
|   |       |       admin_profile_update_resource.py
|   |       |       admin_search_resource.py
|   |       |       admin_users_list_resource.py
|   |       +---general
|   |       |       login_resource.py
|   |       |       signup_resource.py
|   |       \---user
|   |               user_account_delete_resource.py
|   |               user_available_lots_resource.py
|   |               user_dashboard_resource.py
|   |               user_export_csv_resource.py
|   |               user_history_resource.py
|   |               user_profile_fetch_resource.py
|   |               user_profile_update_resource.py
|   |               user_search_resource.py
|   |               user_slot_book_resource.py
|   |               user_slot_release_resource.py
|   |               user_slot_select_resource.py
|   \---templates
|           email_booking_confirmation.html
|           email_csv_export.html
|           email_daily_reminder.html
|           email_monthly_report.html
|           email_slot_release.html
\---Frontend
    |   .env
    |   .env.sample
    |   index.html
    |   jsconfig.json
    |   package-lock.json
    |   package.json
    |   vite.config.js
    +---public
    |       favicon.png
    |       pwa-192x192.png
    |       pwa-512x512.png
    \---src
        |   App.vue
        |   main.js
        +---components
        |       Modal.vue
        |       Navbar.vue
        |       Toast.vue
        |       useToast.js
        +---router
        |       index.js
        \---views
            +---admin
            |       dashboard.vue
            |       history.vue
            |       profile.vue
            |       search.vue
            |       users.vue
            +---general
            |       error.vue
            |       home.vue
            |       login.vue
            |       signup.vue
            \---user
                    dashboard.vue
                    history.vue
                    profile.vue
                    Search.vue
```

The directory structure of the application is shown above.

Implemented Features:

1. **User & Admin Authentication:**
    a. Secure registration and login with JWT-based authentication.
    b. Role-based access control (Admin vs. User) with distinct dashboards.
    c. Secure password hashing and session management.
2. **Admin Dashboard & Management:**
    a. **Parking Lot Management:** Create, update, and delete parking lots with details like location, price, and capacity.
    b. **Real-time Monitoring:** View live status of all parking spots (Available/Occupied) in any lot.
    c. **User Management:** View the list of all registered users and their active status.
    d. **Analytics & History:** Visual charts for revenue and occupancy trends; comprehensive booking history logs.
    e. **Advanced Search:** Search for lots, users, or vehicles using specific queries.
3. **User Features & Booking System:**
    a. **Interactive Dashboard:** View active reservations and quickly access available parking lots.
    b. **Smart Search:** Find parking lots by location name or address.
    c. **Visual Slot Selection:** Interactive map to select specific parking spots within a lot.
    d. **Real-time Booking:** Book spots instantly with vehicle number validation.
    e. **Automated Cost Calculation:** Dynamic cost estimation based on parking duration upon release.
    f. **Booking History:** View past parking records with detailed cost and duration breakdowns.
    g. **CSV Export:** Download personal booking history as a CSV file.
4. **Automated Notifications & Reporting:**
    a. **Daily Pulse Email:** Automated daily emails featuring "Lot of the Day" and parking tips.
    b. **Monthly Report:** Gamified monthly summary emails with "Parker Badges," spending stats, and usage insights.
    c. **Background Tasks:** Celery-powered background jobs for reliable email delivery and scheduled tasks.
5. **System & UI:**
    a. **Responsive Design:** Mobile-friendly UI built with Vue.js and Bootstrap.
    b. **Global Search:** Integrated search functionality for quick access to resources.
    c. **Profile Management:** Update personal details, address, and password for both admins and users.

Video Presentation:

Drive Link: mad2-explainer-video.mkv