App Dev Project Report

1. Student Details

Name: SAKET ANAND
Roll Number: 24dp3000021
Email: 24dp3000021@ds.study.iitm.ac.in

---

2. Project Details

Project Title: Vehicle Parking Management System (ParkPrime)

Problem Statement:

To design and build a multi-user web application to manage vehicle parking lots, individual parking spots, and vehicle bookings. The system must support two distinct roles (Admin and User) and handle real-time spot availability, booking management, and automated background tasks like daily reminders and monthly reports.

Approach:

The application was built using a decoupled Client-Server architecture. The backend is a Flask REST API that manages data via SQLAlchemy (SQLite), handles authentication with JWT, and processes background jobs using Celery and Redis. The frontend is a responsive Single Page Application (SPA) built with Vue.js 3 and Bootstrap, consuming the backend API for all operations.

---

3. AI/LLM Declaration

I used **Gemini (Google)** to assist with approximately **10–15%** of the project. The extent of usage was strictly limited to **generating initial boilerplate code** (such as standard Flask configurations) and **verifying syntax** for specific libraries like Celery and Redis. All core business logic, database schema design, API implementation, and frontend integration were done manually.

---

4. Technologies and Frameworks Used

Technology / Library	**Purpose**
Flask	Core backend web framework for serving APIs
Vue.js 3	Frontend JavaScript framework for the User Interface
SQLite	Lightweight relational database for storing application data
SQLAlchemy	Object Relational Mapper (ORM) for database interactions
Redis	Message broker for Celery and in-memory caching
Celery	Distributed task queue for background jobs (Emails, Reports)
Flask-JWT-Extended	Secure user authentication and role-based access control
Bootstrap 5 (Bootstrap-Vue-3)	Frontend styling and responsive layout grid

---

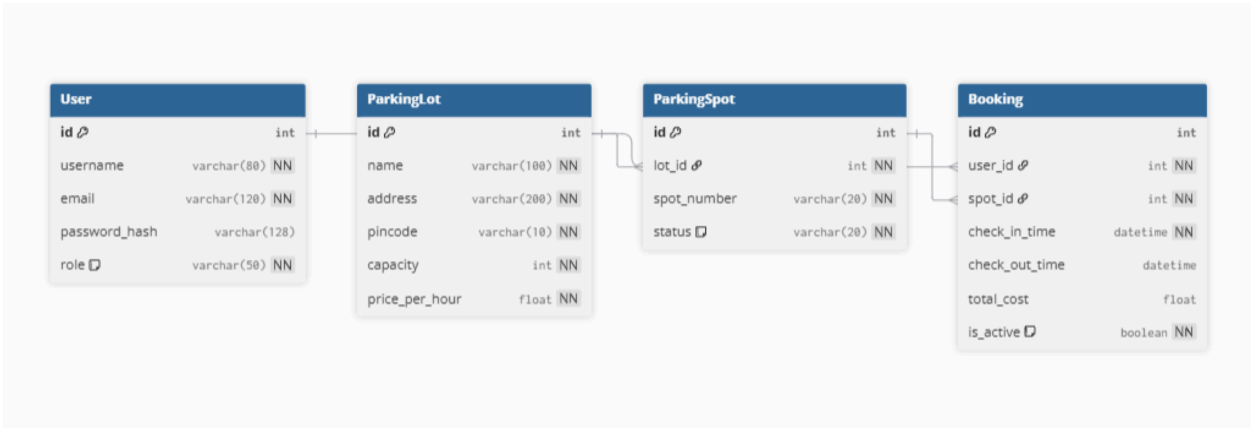
5. Database Schema / ER Diagram

Tables:

1. **User: Stores login credentials and roles (`id`, `username`, `email`, `password_hash`, `role`).**
2. **ParkingLot: Stores details of physical parking areas (`id`, `name`, `address`, `pincode`, `capacity`, `price_per_hour`).**
3. **ParkingSpot: Tracks individual spots within a lot (`id`, `lot_id`, `spot_number`, `status`).**
4. **Booking: Logs user reservations (`id`, `user_id`, `spot_id`, `check_in_time`, `check_out_time`, `total_cost`, `is_active`).**

Relationships:

- **One-to-Many:** `ParkingLot` → `ParkingSpot` (One lot has many spots).
- **One-to-Many:** `User` → `Booking` (One user can have many bookings).
- **One-to-Many:** `ParkingSpot` → `Booking` (One spot can have many bookings over time).



---

6. API Resource Endpoints

Endpoint	Method	Description
/api/register	POST	Register a new user account

/api/login	POST	Authenticate user/admin and return JWT token
/api/lots	GET	Fetch list of available parking lots (Cached)
/api/book	POST	Book a specific parking spot
/api/bookings	GET	Fetch booking history for the logged-in user
/api/release/<id>	PUT	Release a spot and calculate final cost
/api/user/summary	GET	Fetch statistics for User Dashboard
/api/admin/summary	GET	Fetch statistics for Admin Dashboard
/api/admin/lots	GET/POST	Manage parking lots (CRUD operations)
/api/admin/lots/<id>	PUT/DELETE	Update or delete a specific lot
/api/admin/users	GET	View all registered users

/api/export-csv	**POST**	**Trigger async job to email booking history CSV**

YAML API Definition File:
Included separately in the submission ZIP as `api.yaml`.

---

7. Architecture and Features (optional)

Architecture Overview:

- **`backend/app.py`**: Main Flask application entry point and API route definitions.
- **`backend/models.py`**: Database schema classes using SQLAlchemy.
- **`backend/tasks.py`**: Celery tasks for background processing (Emails, CSVs).
- **`frontend/src/views`**: Vue.js page components (Dashboards, Booking Forms).
- **`frontend/src/store.js`**: Global state management for Authentication.

Implemented Features:

- **Role-Based Access:** Distinct dashboards and permissions for Admins and Users.
- **Parking Management:** Admin can create lots; the system auto-generates spots.
- **Booking System:** Users can search, book, and release spots in real-time.
- **Analytics:** Interactive charts showing occupancy and revenue data.
- **Background Jobs:** Automated daily reminders and monthly activity reports sent via email.
- **Performance:** Redis caching implemented for frequently accessed "GET" endpoints.

Additional Features:

- Export user logs as CSV
- AI-generated weekly summary insights (optional extension)

---

8. Video Presentation

Drive Link:
30_11_2025, 11_47_59 pm - Screen - Video Project.webm
(Accessible to all with “View” permission.)
