# Vehicle Parking App - Modern Application Development I - May 2025

## Author

**Name**: RONY MANDAL
**Roll number**: 23f2002480
**Student email**: 23f2002480@ds.study.iitm.ac.in
**About**: Dual degree student pursuing IITM-BS Online Degree along with B. Tech in AI-ML

## Description

This project involved creating a multi-user web application to manage vehicle parking. The system supports two roles: an Admin who manages parking lots and views all activity, and a User who can register, book, and release parking spots. The application handles real-time spot allocation, reservation history, and cost calculation.

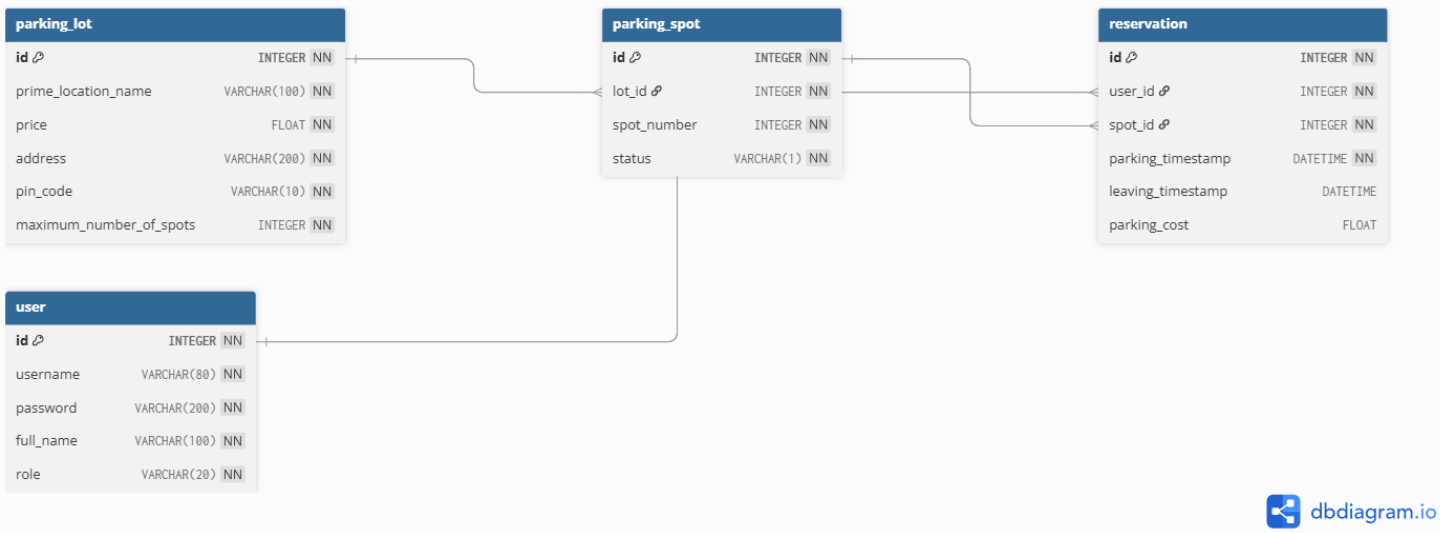
**AI Usage Declaration**: I declare that I used approximately 25% of AI/LLM assistance for this project.
**Extra Details**: The AI was primarily used for debugging complex errors related to data integrity and application logic.

## Technologies used

- **Flask:** A lightweight Python web framework used for the backend logic and routing.
- **Flask-SQLAlchemy:** An extension for Flask that simplifies database interactions using SQLAlchemy ORM.
- **Flask-Login:** Managed user sessions for handling authentication and protecting routes.
- **Jinja2:** The templating engine used to render the frontend HTML pages with dynamic data.
- **SQLite:** A serverless, file-based database used for all data persistence.

This tech stack was selected to support rapid development, clean architecture, and secure user management. Each tool serves a distinct purpose—from handling routing and templates to managing data and authentication—resulting in a robust and maintainable web application.

## DB Schema Design




This design was chosen to normalize data, prevent redundancy, and use database-level constraints to ensure long-term data integrity.

## API Design

API creation was an optional enhancement. As the primary focus was on completing the core application, dedicated RESTful APIs were not implemented. However, the application is structured with a clear separation of concerns, which would allow for straightforward implementation of APIs using Flask-RESTful.

## Architecture and Features

The project is organized into a standard Flask application structure.

- **app.py**: The main application file containing all routes and core logic.
- **/models**: Contains database.py, which defines the SQLAlchemy database models.
- **/templates**: Contains all Jinja2/HTML templates for the frontend.
- **/static**: Contain CSS and JavaScript files for styling.

Features Implemented

- **Role-Based Access:** Separate dashboards and permissions for Admins and Users.
- **Admin Lot Management:** Full CRUD (Create, Read, Update, Delete) functionality for parking lots, including robust logic to prevent deletion of lots with active or historical reservations.
- **User Parking Cycle:** Users can view lots, be auto-assigned a spot, occupy it, and release it.
- **History & Costing:** All parking sessions are recorded, with cost automatically calculated based on duration and stored. Both Admins and Users can view relevant parking history.

## Video

**Link**: https://drive.google.com/file/d/1hl15eXN902-KAwxpU3aRRLfXrxypcsE9/view