Smart Parking Management System

MAD-1 Final Project Report

Rounak Mukhopadhyay
Roll No: 22f1000876
22f1000876@ds.study.iitm.ac.in

July 30, 2025

Contents

1 Introduction

- **Objective of the Project:** The primary goal of this project is to develop a smart and scalable web-based parking management system that allows users to book, cancel, and manage parking slots in real-time. Administrators can manage multiple parking lots and oversee the status and usage of individual slots. A super-admin oversees the entire system and assigns admin rights.
- **Importance of Parking Management in Urban Environments:** As urban populations continue to rise, so does the number of vehicles, leading to severe parking shortages, traffic congestion, and wasted fuel. Efficient parking systems help reduce these issues by organizing space usage, lowering wait times, and improving urban mobility.
- **Role of Web Applications in Optimizing User/Admin Workflows:** Web applications provide a platform-independent, real-time, and accessible interface for both end-users and administrators. With integrated user management, role-based access, and real-time booking updates, the web system enhances usability, reduces manual errors, and streamlines daily parking operations.

2 System Overview

2.1 Features Implemented

- Role-based user registration and authentication system
- Super admin functionality to assign or revoke admin privileges
- Admin dashboard to manage multiple parking lots and their respective slots
- Real-time slot booking and cancellation functionality for users
- Dynamic slot status view, including booking user and timestamp details
- Persistent SQLite database managed via SQLAlchemy ORM for clean model relationships

2.2 Technology Stack

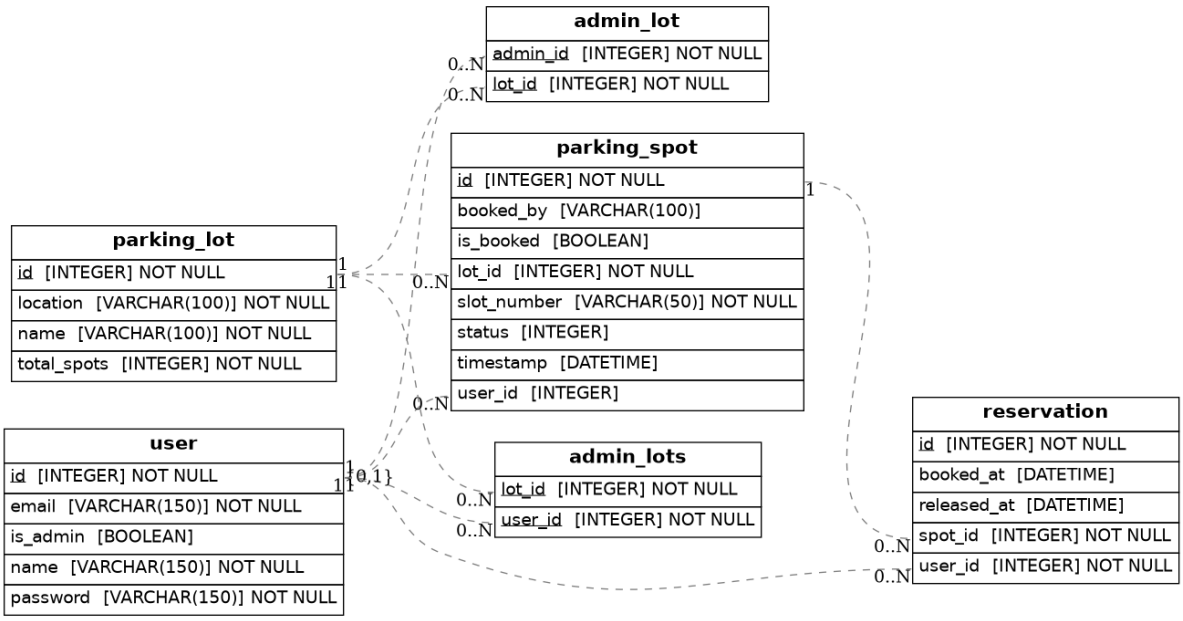
Technology	Role in Project
Flask	Backend logic, routing, request handling
SQLite	Lightweight relational database for data persistence
SQLAlchemy	Object-Relational Mapping (ORM) for managing models and queries
Flask-Login	User authentication and session management
Jinja2	Template engine for server-side HTML rendering
Bootstrap	Responsive frontend layout and styling

3 Architecture

3.1 Directory Structure

```
project/

 app.py                  # Main entry point for the Flask app
 models/                 # Contains SQLAlchemy models
    models.py

 controllers/            # Contains route definitions and view logic
    routes.py

 templates/              # HTML templates rendered using Jinja2
    base.html
    login.html
    register.html
    dashboard_user.html
    dashboard_admin.html
    view_lot.html
    ... (other templates)

 instance/               # Contains SQLite database file
    parking_app.db

 db_init.py              # Script for initializing the database
 init_db.py              # Optional DB seeding or setup script
```

3.2 ER Diagram



4 Database Models

- **User:** Stores user credentials and roles. Roles include regular users, admins, and a super admin. Admins can manage specific parking lots.
- **ParkingLot:** Contains information about each parking lot, including name, location, and total number of spots. Each lot is managed by one or more admins.
- **ParkingSpot:** Represents individual parking spots in a lot. Tracks slot number, booking status, timestamp, and the user who booked it.
- **Reservation:** Stores booking history, including timestamps of when a spot was booked and released. Each reservation links a user to a parking spot.
- **AdminLot / AdminLots:** Associative tables to manage many-to-many relationships between admins and the parking lots they oversee.

5 User Interface Screens

- **User Login and Register:** Users can securely sign up with their name, email, and password. The login screen allows authentication and redirects based on user role (admin or regular user).
- **Admin Dashboard (Lot and Slot Management):** Admins can view and manage parking lots assigned to them. They can monitor slots, check booking status, and manage availability in real time.
- **User Booking Interface:** Regular users can select a parking lot, view available slots, and make a booking. Slot availability is updated live.
- **My Bookings (History View):** Users can view their past and current reservations with timestamps of booking and release. Useful for tracking parking activity.
- **Super Admin Role Manager:** A special interface for the super admin (hard-coded user) to assign or revoke admin privileges to other users. Ensures centralized admin control.

6 AI Assistance Usage (MAD-1)

- **Code Generation:** AI was used to generate boilerplate and production-ready code for Flask routes, SQLAlchemy models, and the overall file structure of the web application.
- **Jinja2 and HTML Templating Support:** Help was taken to structure Jinja2 templates effectively for dynamic data rendering, loop logic, and conditional formatting in the HTML pages.
- **Database Model Design Guidance:** Assistance was provided in designing relational database models with proper relationships (one-to-many, many-to-many), foreign key constraints, and normalization.

- **Debugging and Refactoring:** AI helped debug route logic, resolve template errors, and improve code efficiency via refactoring suggestions for both frontend and backend components.
- **UI/UX Feedback:** Suggestions were taken to improve user interface layout, user flow, and overall usability, ensuring intuitive navigation and functionality for admins and users.

7 Contribution Breakdown (MAD-1)

This section presents the breakdown of the student's individual contribution to the Vehicle Parking Web App project. The work is divided across backend development, frontend implementation, database design, and integration of AI assistance.

Component / Task	Weight (%)	Description
Flask Backend Development	25%	Implemented routing, user authentication, and integration with database models using SQLAlchemy
Admin Functionality	15%	Developed role-based admin controls including multi-lot management and super admin privileges
User Booking System	20%	Created logic for slot booking, cancellation, and real-time availability status updates
Frontend Development (HTML/CSS + Jinja2)	20%	Designed and structured UI templates using Bootstrap and dynamic rendering with Jinja2
Database Design (SQLite + ORM)	10%	Defined models for User, ParkingLot, ParkingSpot, and Reservation with proper relationships
AI Assistance Usage	10%	Used ChatGPT for guidance on architecture, debugging, templating logic, and optimization suggestions

8 Challenges Faced

- **Handling Many-to-One Relationships:** Designing the schema so that each admin could manage multiple parking lots without violating referential integrity or causing query complexity required careful relationship mapping using SQLAlchemy.
- **Ensuring Slot Uniqueness and Availability:** Preventing race conditions during booking, especially when multiple users attempted to book the same slot simultaneously, involved implementing validation checks and database-level constraints.
- **Building Reusable Templates for Different User Roles:** Maintaining a clean UI/UX while differentiating access and interface for regular users, admins, and the

super admin involved the use of dynamic Jinja2 templating and role-based access control.

- **Protecting Super Admin Account from Deletion or Role Change:** Special logic had to be implemented to safeguard the primary super admin (e.g., based on email ID), ensuring this account could not be accidentally edited or removed from the admin panel.

9 Conclusion and Future Scope

- Successfully implemented all core MAD-1 features including user registration, parking lot and slot management, role-based admin access, and booking workflows.
- The project follows a modular and scalable architecture, making it well-prepared for transition into the MAD-2 phase with minimal structural changes.
- **Future Enhancements:**
  - Integration of QR-based check-in/check-out system for seamless entry and exit tracking.
  - Incorporation of secure payment gateways to enable paid bookings and real-time transaction logs.
  - Development of a dedicated mobile application using Flutter or React Native for better accessibility and user experience on mobile devices.

10 References

- Flask Documentation: `https://flask.palletsprojects.com/`
- SQLAlchemy ORM Docs: `https://docs.sqlalchemy.org/`
- Bootstrap CSS Framework: `https://getbootstrap.com/`
- StackOverflow and GitHub Discussions – For resolving implementation issues and exploring best practices
- ChatGPT – Assisted in code generation, logic refinement, debugging, UI design ideas, and LaTeX formatting