Vehicle Parking Management System - Project Report
Modern Application Development I Project(MAD-I Project)

Name: Vineet Kumar
Course: Modern Application Development I
Project: Vehicle Parking Management System

Problem Statement

The challenge was to create a comprehensive vehicle parking management system that meets the increasing demand for effective parking options in cities. In order to smooth booking experiences, the system had to manage both user and administrative functionalities.

Project Approach

In order to address this issue, I developed a full-stack web application with distinct divisions of responsibilities:

1. User-Centric Design: created user-friendly interfaces for administrators and common users alike.
2. Role-Based Access Control: Created discrete user roles with the proper authorization
3. Real-Time Management: Developed dynamic cost estimation and parking space distribution
4. Indian Context Integration: Added accurate parking scenarios and prices from India.
5. Responsive Design: Guaranteed compatibility with various screen sizes and devices

The development started with the core functionality and gradually evolved, adding features like custom dashboard designs, local file handling, and a smooth booking experience. Everything was built step by step, following an iterative approach that allowed the project to grow naturally and stay focused on real user needs.

Technical Implementation

Frameworks and Libraries Used

➢ Backend technologies:
- Flask (v2.3.0): the main web framework used for application logic, request processing, and routing
- SQLAlchemy: Object-Relational Mapping for model definitions and database operations
- SQLite: A lightweight database for storing and preserving data
- Werkzeu: Security tools for authentication and hashing passwords

➢ Frontend Technologies:
- HTML5 + Jinja2: dynamic content creation and server-side template rendering
- Bootstrap 5.1.3: A responsive user interface framework for layout and styling
- FontAwesome 6.0.0: A library of icons for a better user interface
- Custom CSS: Tailored styling for distinctive visual effects and dashboard designs

Database Schema (ER Diagram Description)

The database consists of five primary entities with clearly defined relationships that support a parking management system.

Entities:

1. User
   Stores information related to users who can make parking reservations.
   - Attributes:
     - id: Primary key
     - username: Unique username
     - email: Unique email
     - full_name: Full name of the user
     - password_hash: Hashed password

        - is_admin: Boolean flag indicating admin privileges

2. Admin
   A separate entity to handle administrative authentication independently of regular users.
   - Attributes:
     - id: Primary key
     - username: Unique admin username
     - password: Admin password
3. ParkingLot
   Represents individual parking facilities with location and pricing details.
   - Attributes:
     - id: Primary key
     - prime_location_name: Name of the area or landmark
     - address: Full street address
     - pin_code: Postal code
     - price_per_hour: Rate charged per hour of parking
     - maximum_number_of_spots: Total capacity of the parking lot
4. ParkingSpot
   Represents individual parking spaces within a given parking lot.
   - Attributes:
     - id: Primary key
     - lot_id: Foreign key referencing ParkingLot.id
     - spot_number: Unique identifier for the spot within the lot
     - status: Availability status - 'A' (Available), 'O' (Occupied)
5. Reservation
   Captures booking and usage data for a parking spot by a user.
   - Attributes:
     - id: Primary key
     - user_id: Foreign key referencing User.id
     - spot_id: Foreign key referencing ParkingSpot.id
     - vehicle_number: License plate or registration number
     - parking_timestamp: DateTime when parking started
     - leaving_timestamp: DateTime when parking ended (nullable)
     - parking_cost: Total cost incurred for the session (nullable)
     - status: Reservation status – e.g., 'active', 'completed', 'cancelled'

Relationships:

- User → Reservation (One-to-Many)
  A user can have multiple reservations associated with them.
- ParkingLot → ParkingSpot (One-to-Many)
  A parking lot contains multiple uniquely numbered spots.
- ParkingSpot → Reservation (One-to-Many)
  A single parking spot may have multiple reservations over time.
- User ← → Reservation ← → ParkingSpot
  A complete reservation chain exists linking a user to a specific parking spot, managed via a reservation record.

<u>Core Features</u>

This parking management system offers a complete, user-friendly solution with powerful tools for both users and administrators. Designed with real-world usability in mind, it combines functionality, security, and a seamless experience.

User Management

- Secure Registration & Login – New users can sign up and log in with their credentials safely.
- Password Protection – All passwords are securely hashed to protect user data.
- Session Handling – Keeps users logged in during active sessions for a smooth experience.
- Role-Based Access – Separates access and features for regular users and administrators.

Parking Management

- Flexible Lot Creation – Admins can easily add new parking lots with detailed location and pricing data.
- Automatic Spot Generation – Parking spots are auto-created based on lot capacity.
- Live Availability Tracking – System keeps track of which spots are available or occupied in real time.
- Smart Allocation – Automatically assigns the first available spot to minimize user effort.

Reservation System

- Simple Booking Interface – Users can quickly book spots with just a few clicks.
- Vehicle Number Entry – Required during booking to identify the vehicle.
- Auto Timestamping – Start time is recorded the moment a reservation is made.
- Cost Calculation – Charges are automatically calculated based on the parking duration.
- Spot Release Option – Users can release their spot once done, freeing it for others.

Admin Tools

- Powerful Dashboard – Visual dashboard with key metrics and real-time insights.
- Full CRUD for Lots – Create, update, or delete parking lots and their details.
- User Oversight – View and monitor registered users and their activity.
- Reservation Logs – Track all reservations and their status/history.
- Custom Admin Interface – Sidebar-driven, intuitive design tailored for admin workflows.

User Experience & Local Integration

- Timeline-Based Dashboard – Users can view current bookings and history in an organized, chronological layout.
- Real-Time Cost Display – Parking charges update dynamically based on time used.
- Mobile-Responsive Design – Optimized for smooth use on smartphones and tablets.
- Localized Features – Supports Indian locations, pricing standards, and vehicle number formats.

API Endpoints

This Flask application includes a set of internal API endpoints that support both user and administrative functions.

Authentication Endpoints

- POST /login
  Authenticate a regular user and initiate a session.
- POST /admin/login
  Authenticate an admin user.
- GET /logout
  Terminate the current user session.

User Management Endpoints

- POST /register
  Register a new user with required credentials and personal information.
- GET /user/dashboard
  Retrieve data for the user’s dashboard, such as active reservations and profile info.
- GET /user/history
  Fetch the parking history for the logged-in user.

Reservation Management Endpoints

- POST /book_parking/<lot_id>
  Create a new reservation in the specified parking lot.
- GET /release_parking/<reservation_id>
  Release the reserved parking spot and calculate the parking cost.

Administrative Endpoints

- GET /admin/dashboard
  Display admin dashboard with usage statistics and system summaries.
- POST /admin/add_lot
  Add a new parking lot with details like location, capacity, and pricing.

- PUT /admin/edit_lot/<lot_id>
  Update existing parking lot information.
- DELETE /admin/delete_lot/<lot_id>
  Permanently remove a parking lot and its associated spots.

Testing & Sample Data
To facilitate development and testing, I have added realistic sample data with common urban parking scenarios in India

Sample Parking Locations
- BKC Corporate Hub, Mumbai – ₹100/hour
- Electronic City Tech Park, Bangalore – ₹40/hour
- Cyber City, Gurgaon – ₹90/hour
- Brigade Road Shopping District, Bangalore – ₹55/hour, etc.

Test User Accounts
- Rajesh Kumar
  rajesh.kumar@gmail.com
- Priya Sharma
  priya.sharma@yahoo.in
- Amit Patel
  amit.patel@rediffmail.com
- Sneha Reddy
  sneha.reddy@hotmail.com
- Vikram Singh
  vikram.singh@gmail.com
- Admin Account
  Username: admin
  Password: admin123

AI/LLM Usage Disclosure
As part of the MAD-1 project guidelines, I'm acknowledging the ways in which AI tools (like ChatGPT) were used during the development of this application. Rather than relying on AI for full-code generation, I used it more like an assistant helping me brainstorm, debug, and refine ideas along the way.

Here are the areas where AI supported me:
1. Structuring the Code (~10%)
2. Backend (~15%) - Guided me in designing SQLAlchemy relationships and building a clean, normalized schema.
3. Frontend (~8%) - Offered suggestions for layout improvements using Bootstrap and custom CSS, especially for the timeline view and admin sidebar.
4. Debugging Support (~8%) - Assisted in tracking down tricky issues like import errors, Jinja2 bugs, and database connection problems.
5. Documentation (~5%) - Helped me write clear comments and polish sections of this project report.

Specific Contributions from AI:-
1. Template Layouts – Assisted me in crafting a sidebar-based admin panel and a timeline-style user dashboard.
2. Database Relationships – Helped me troubleshoot foreign key relationships and solve datetime issues related to UTC and timezones.
3. Static Files & Assets – Helped me in shifting from external image URLs to local static assets for better control and offline usage.
4. Error Fixing – Helped resolve common frontend/backend errors.

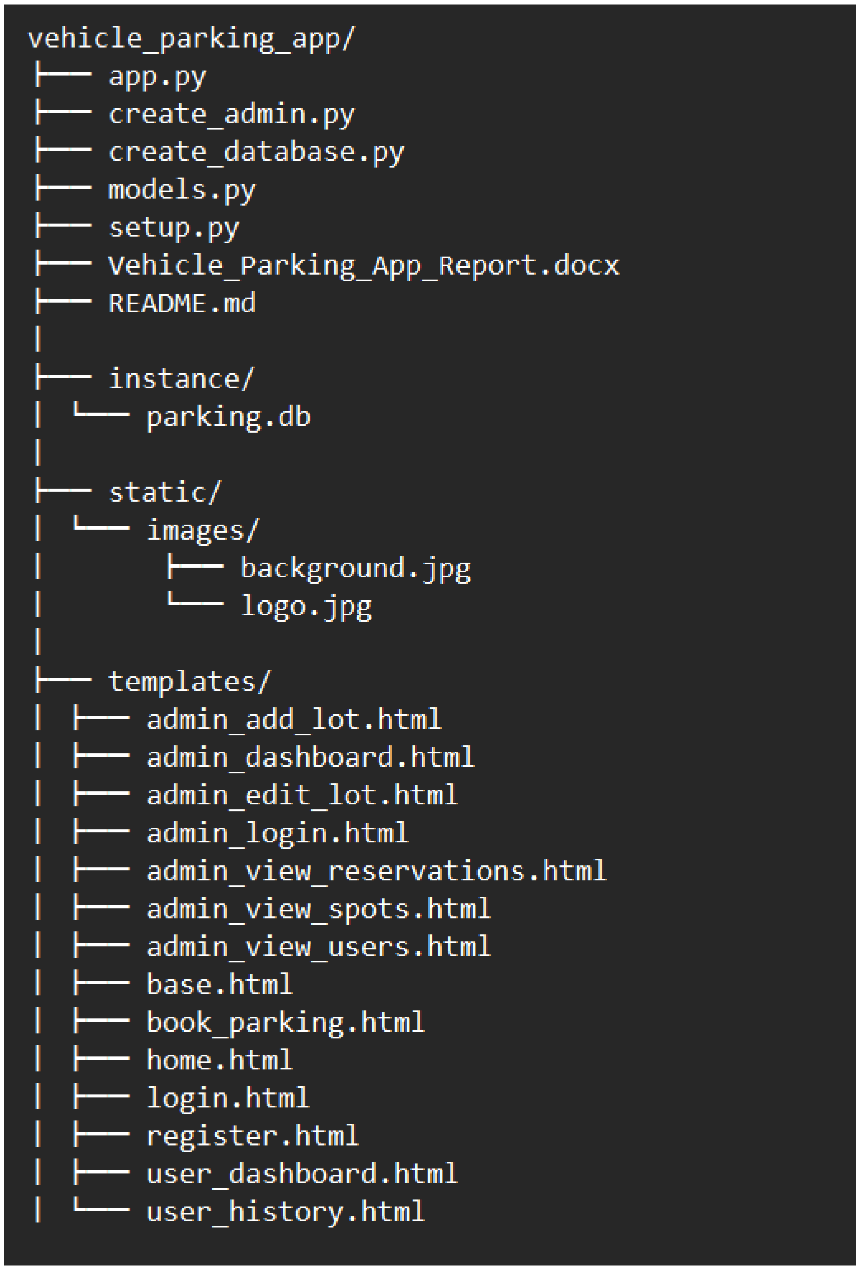
My Part:-

All core features, like reservation logic, pricing calculations, dashboard logic, and the overall flow were planned and built by me. I made key decisions around how things should work, how users interact with the system, and how to balance simplicity with functionality. Custom styling and structure also reflect my personal design choices.
Throughout the project, I treated AI as a learning companion. Every time it offered a solution or suggestion, I made sure I understood it, adapted it to fit the project, and wrote the final implementation myself. My goal wasn't to copy and paste rather it was to learn by doing, with a little help along the way.

Conclusion
The Vehicle Parking Management System was built to tackle the everyday challenge of managing urban parking more efficiently. It combines solid backend logic with a clean, user-friendly interface, all prepared to real-world Indian scenarios. From booking spots to admin control, everything works smoothly and is ready for practical use. Overall, this project was a great learning experience for me in full-stack development and real-world problem-solving.

Project Video Drive Link:- https://drive.google.com/file/d/1jdM7AoQd0Y795CsyMoDCPrYIinzNAJvC/view?usp=sharing

File Structure:-



```
vehicle_parking_app/
├── app.py
├── create_admin.py
├── create_database.py
├── models.py
├── setup.py
├── Vehicle_Parking_App_Report.docx
├── README.md
|
├── instance/
|  └── parking.db
|
├── static/
|  └── images/
|        ├── background.jpg
|        └── logo.jpg
|
├── templates/
|  ├── admin_add_lot.html
|  ├── admin_dashboard.html
|  ├── admin_edit_lot.html
|  ├── admin_login.html
|  ├── admin_view_reservations.html
|  ├── admin_view_spots.html
|  ├── admin_view_users.html
|  ├── base.html
|  ├── book_parking.html
|  ├── home.html
|  ├── login.html
|  ├── register.html
|  ├── user_dashboard.html
|  └── user_history.html
```