Vehicle Parking Management System

Project Report

Author:

Name: Kshitij Nigam

Roll Number: 23f3002142

Email: 23f3002142@ds.study.iitm.ac.in

I am currently pursuing a BS in Data Science at IIT Madras. I have a keen interest in backend development, databases, and learning how technology can solve real-world problems through practical applications.

Description:

This project aims to build a web-based vehicle parking management system that allows users to search for parking lots, book spots, and view booking history while enabling admins to manage data.

AI/LLM Used: 25% (For Debugging , error resolution, UI code suggestions, form validations, and optimization)

Technologies Used:

- Python (Flask) – Main backend framework

- Flask-SQLAlchemy – ORM for SQLite database

- Flask-Login – To handle user sessions and authentication

- SQLite – Lightweight database for storage

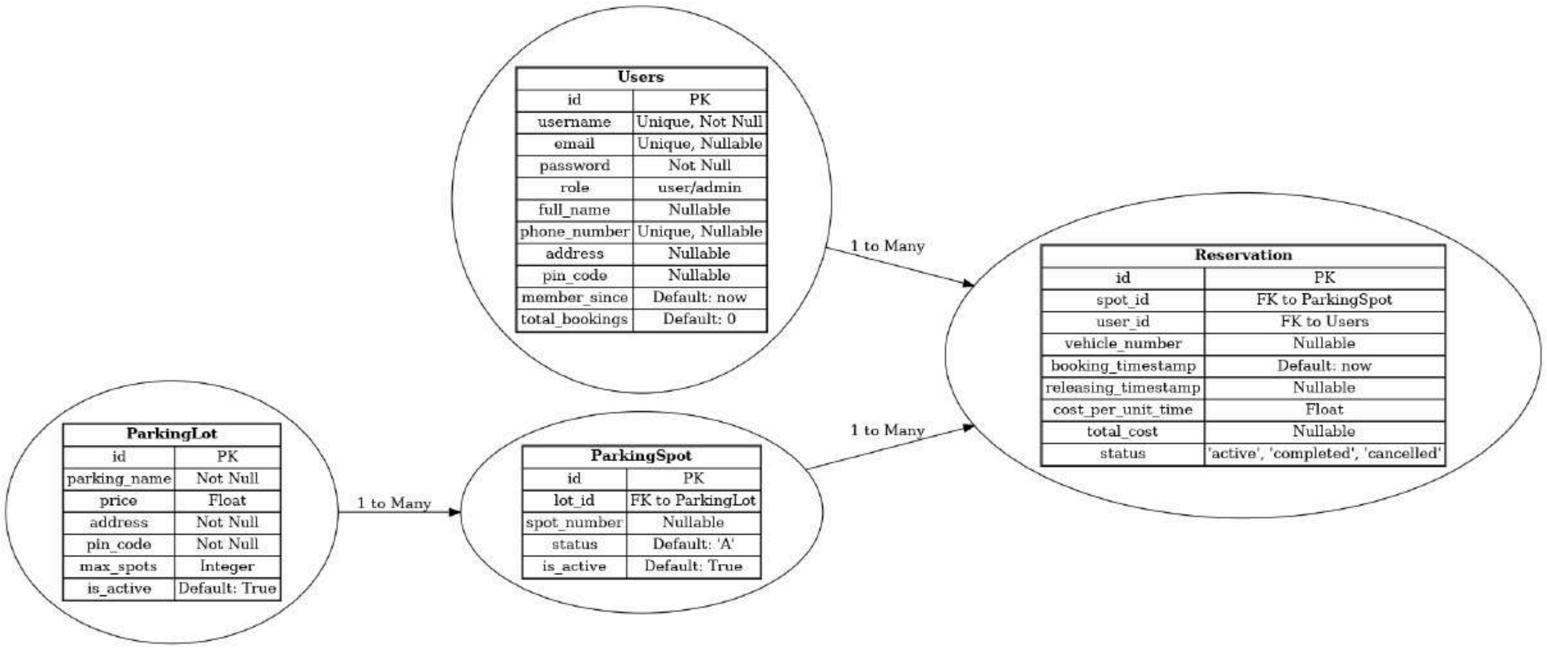
- Bootstrap 5 – Responsive frontend framework

- Jinja2 – Templating engine used with Flask

- Chart.js – To visualize booking history data

Purpose: Flask was chosen for its simplicity and lightweight nature for rapid prototyping. Flask extensions like SQLAlchemy and Flask-Login handled data and user auth effectively. Bootstrap ensured responsive, mobile-friendly design.

DB Schema Design:



Users	
id	PK
username	Unique, Not Null
email	Unique, Nullable
password	Not Null
role	user/admin
full_name	Nullable
phone_number	Unique, Nullable
address	Nullable
pin_code	Nullable
member_since	Default: now
total_bookings	Default: 0

Reservation	
id	PK
spot_id	FK to ParkingSpot
user_id	FK to Users
vehicle_number	Nullable
booking_timestamp	Default: now
releasing_timestamp	Nullable
cost_per_unit_time	Float
total_cost	Nullable
status	'active', 'completed', 'cancelled'

ParkingLot	
id	PK
parking_name	Not Null
price	Float
address	Not Null
pin_code	Not Null
max_spots	Integer
is_active	Default: True

ParkingSpot	
id	PK
lot_id	FK to ParkingLot
spot_number	Nullable
status	Default: 'A'
is_active	Default: True

Tables:

Users

- **id**: PK, Integer
- **username**: Unique, String, Not Null
- **email**: Unique, String, Nullable
- **password**: String, Not Null
- **role**: String ('user' or 'admin'), Not Null
- **full_name**: String, Nullable
- **phone_number**: Unique, String, Nullable
- **address**: String, Nullable
- **pin_code**: String, Nullable
- **member_since**: Datetime, Default: `datetime.utcnow`
- **total_bookings**: Integer, Default: 0

Relationships: One-to-many with **ReservationParkingLot**

- **id**: PK, Integer
- **parking_name**: String, Not Null
- **price**: Float, Not Null
- **address**: String, Not Null
- **pin_code**: String, Not Null
- **max_spots**: Integer, Not Null
- **is_active**: Boolean, Default: True

Relationships: One-to-many with **ParkingSpotParkingSpot**

- **id**: PK, Integer
- **lot_id**: FK to ParkingLot.id, Not Null
- **spot_number**: String, Nullable (Human-friendly identifier)
- **status**: String, Default: 'A' ('A' for Available)
- **is_active**: Boolean, Default: True (Admin control)

Relationships: One-to-many with **ReservationReservation**

- **id**: PK, Integer
- **spot_id**: FK to ParkingSpot.id, Not Null
- **user_id**: FK to Users.id, Not Null
- **vehicle_number**: String, Nullable
- **booking_timestamp**: Datetime, Default: `datetime.utcnow`
- **releasing_timestamp**: Datetime, Nullable
- **cost_per_unit_time**: Float, Not Null
- **total_cost**: Float, Nullable
- **status**: String, Default: 'active' (Values: 'active', 'completed', 'cancelled')

Design Rationale: Designed to normalize data, reduce redundancy, and ensure clear user-parking relationships. Foreign keys maintain referential integrity.

API Design:

APIs were created for:
- User Authentication (Login/Register)
- Search Parking Lots (by name, pincode, address)
- Book Spot and Release Spot
- Admin Search (by user, lot number, lot name)

Implementation: All APIs were created using Flask routes, some returning HTML via render_template() and others returning JSON using @app.route(..., methods=['POST']).
(YAML file is submitted separately.)

Architecture and Features:

Architecture:
- app.py: Entry point and route definitions
- templates/: HTML templates (Jinja2)
- static/: Bootstrap, JS, and Chart.js files
- models.py: All SQLAlchemy models

Default Features:
- User login/registration

- Admin login
- Book a parking spot
- View booking history
- Generate visual report (Chart.js)

Additional Features:

- Search by lot name/address/pincode
- Admin dashboard with search filters
- Auto-update available spots on booking/release
- Clean and responsive UI using Bootstrap

Video:

Link: Easepark.mp4