

Author: Ansul Mishra  
Roll No.: 24f1001211  
Email: 24f1001211@ds.study.iitm.ac.in  
Education: 3rd year B.Tech CSE (AI & ML), SRMIST KTR

A web-based solution for managing parking operations. It allows lot owners to manage spaces and users to find, book, and pay for spots. Built with Flask (modular MVC structure) and SQLite, the project emphasizes secure authentication and analytics-ready design.

Backend: Flask, Flask-SQLAlchemy, SQLAlchemy, Flask-Login, Werkzeug  
Database: SQLite  
Frontend: Bootstrap 5, Font Awesome, Jinja2  
Chosen for: scalability, security, simplicity in rapid development

LLMs supported ~45–50% of the project:

- Frontend (HTML/JS): 28%
- UI styling: 8%
- Extras (API integration, edge cases): 9%
- Backend/DB design: minor support

ER diagram for a parking management system:

- Entities:**
  - User** (Attributes: username, qualification, qualification, Rollnum)
  - ParkingLot** (Attributes: Full\_name, (UNIQUE), Crdtime\_gf, Admimo\_gf, Created\_at)
  - User (Unnt)** (Attributes: Prime Address, Address, Crpated at)
  - User (Lemuræ)** (Attributes: Prime Address, Address, Crpated at)
  - Eitel gore (500)** (Attributes: Prime Address, Address, Crpated at)
  - Lot\_id (FK)** (Attributes: Prime Address, Address, Crpated at)
  - Total cost** (Attributes: Prime Address, Address, Crpated at)
  - Resolation** (Attributes: Prime Address, Address, Crpated at)
  - Resolotion** (Attributes: Prime Address, Address, Crpated at)
- Relationships:**
  - User** ↔ **User (Unnt)** (Relationship: Full\_name)
  - User (Unnt)** ↔ **User (Lemuræ)** (Relationship: User)
  - User (Lemuræ)** ↔ **Eitel gore (500)** (Relationship: Eitel gore (500))
  - Eitel gore (500)** ↔ **Lot\_id (FK)** (Relationship: Lot\_id (FK))
  - Lot\_id (FK)** ↔ **Total cost** (Relationship: Total cost)
  - Total cost** ↔ **Resolation** (Relationship: Resolation)
  - Resolation** ↔ **Resolotion** (Relationship: Resolotion)
- Notes:**
  - Resolotion** is a **PK, FK V** (Maximum number of spots (Recenarpld or Vind)).
  - User make for a/n multiple (and ito-ows the Many) over lastrut**.

- users: id (PK), username (UQ), email (UQ), password, role, balance, vehicle info

- parking\_lots: id (PK), location, price, layout, features
- parking\_spots: id (PK), lot\_id (FK), coordinates, status (A/O/M)
- reservations: id (PK), user\_id (FK), spot\_id (FK), time, cost, vehicle, status
- payments: id (PK), user\_id (FK), reservation\_id (FK), amount, date, method
- transactions: id (PK), user\_id (FK), amount, type, description, reference\_id (UQ)
- system\_stats: date (UQ), revenue, reservations, occupancy

Relationships:

- Users → Reservations, Payments, Transactions
- ParkingLots → ParkingSpots
- ParkingSpots → Reservations (1 active)
- Reservation → Payment

Design Goals: Modular, normalized, secure, supports real-time queries and analytics.

## API Design (Flask Routes)

Auth: /login, /register, /logout

User: /user/dashboard, /profile, /wallet, POST /book/<lot\_id>, POST /release/<reservation\_id>, /api/wallet/balance

Admin: /admin/dashboard, /analytics, CRUD routes for lots, /api/lot/<lot\_id>/layout

## Architecture & Key Features

Architecture (MVC):

- app.py: Flask setup & controller loading
- models/models.py: ORM schema
- controllers/: auth, user, admin
- templates/: HTML (Jinja2)
- static/: CSS
- utils.py: helper functions

User Features:

- Register/login/logout
- Profile edit & password change
- Wallet view, add/withdraw funds
- Book/release parking spots
- View history

Admin Features:

- Dashboard: user/lots/revenue overview
- CRUD for parking lots & layout
- View analytics: revenue, occupancy

## Video Demo

View here:

[https://drive.google.com/file/d/1\\_7MMqy59Qza8ewNzLwFcw3YtWdixuKvI/view?usp=sharing](https://drive.google.com/file/d/1_7MMqy59Qza8ewNzLwFcw3YtWdixuKvI/view?usp=sharing)