### Author

### **Name:** Vinoothna Nanumasa **Roll Number:** 23f1001294 **Email:** [23f1001294@ds.study.iitm.ac.in](mailto:23f1001294@ds.study.iitm.ac.in) I am an enthusiastic computer science student with a keen interest in building practical web applications using Python and Flask. I enjoy designing systems that solve real-world problems and help users in an efficient way.

### Description

This project implements a web-based **Parking Lot Management System** that allows users to search for parking lots, book slots, view reservations, and release occupied slots with dynamic cost calculation.  
 **AI/LLM Used:** 0% — the entire logic is implemented using core Flask, Jinja2 templating, SQLAlchemy ORM and standard Python without any AI code generation in production.

### Technologies used

### **Python** — Core language for backend logic. **Flask** — Micro web framework. **Flask-SQLAlchemy** — ORM to interact with the database. **Jinja2** — Templating engine for dynamic HTML rendering. **SQLite** — Lightweight relational database. **Bootstrap** — For simple and responsive front-end styling. **Werkzeug** — For secure routing and URL handling.

**Purpose:** Flask provides an easy and modular way to build REST endpoints and HTML rendering. SQLAlchemy simplifies DB operations. Jinja2 handles templates and makes it easy to reuse layouts and dynamic data rendering.

### DB Schema Design

**Tables:**

1. **User\_Info**id *(Primary Key, Integer)*email *(Unique, String)*

password *(String)*role *(Integer, default=1)* — 0=Admin, 1=User.  
full\_name *(String)*address *(String)*pin\_code *(Integer)*vehicle\_no *(String)*

1. **ParkingLot**id *(Primary Key, Integer)*prime\_location\_name *(String)*location *(String)*pin\_code *(Integer)*tkt\_price *(Float)* — Cost per hour.  
   capacity *(Integer)*
2. **Slot**id *(Primary Key, Integer)*status *(Enum — Available/Occupied)*parkinglot\_id *(Foreign Key → ParkingLot.id)*
3. **Reservation**id *(Primary Key, Integer)*user\_id *(Foreign Key → User\_Info.id)*slot\_id *(Foreign Key → Slot.id)*parking\_timestamp *(DateTime)*leaving\_timestamp *(DateTime, Nullable)*parking\_cost *(Float, Nullable)*

**Design Reason:** Normalized to separate user data, parking lot details, slots, and reservations with clear relations. Slots and reservations are connected through slot\_id. Constraints maintain integrity.

### API Design

### This Parking Lot Management System exposes several **Flask route endpoints**, organized by user role and functionality.

### **Key Endpoints:**

### **/** — Homepage.

### **/login** — Login for both Admin and Users (checks credentials, sets session).

### **/register** — New User Registration.

### **/logout** — Clears session and redirects to login.

### **Admin Panel Endpoints:**

### **/admin/<name>** — Admin dashboard showing all parking lots.

### **/add\_lot** — Add new parking lot with dynamic slot creation.

### **/edit\_lot/<lot\_id>** — Edit parking lot details and adjust capacity (slots are added or removed accordingly).

### **/delete\_lot/<lot\_id>** — Delete an entire parking lot and a lot cannot be deleted if tere are any occupied slots in it.

### **/view\_delete\_slot/<slot\_id>** — View or delete an individual parking slot (only if available).

### **/occupied\_slot\_details/<slot\_id>** — View details for an occupied slot’s active reservation.

### **/users** — View all registered users.

### **/admin\_search** — Search for users by ID or lots by location.

### **/summary** — Admin summary with revenue pie chart and occupancy bar graph (generated dynamically with Matplotlib).

### **/edit\_profile** — Admin can edit their profile information (excluding email).

### **User Endpoints:**

### **/user/<id>/<name>** — User dashboard: view lots, reservations, and search.

### **/search** — Search parking lots by location or pin code.

### **/book\_slot/<lot\_id>/<user\_id>** — Book an available slot in a selected lot.

### **/release/<reservation\_id>** — Release a booked slot, automatically calculate parking cost based on lot-specific tkt\_price.

### **/summary/<user\_id>** — User-specific summary showing usage statistics with simple visualizations.

### **/edit\_userprofile** — Users can edit their profile details (excluding email).

### **Implemention:**

### All endpoints are implemented using **Flask routes**.

### Session handling is used to persist user/admin identity.

### Data is fetched and updated using **SQLAlchemy ORM models**.

### Templating is done with **Jinja2** for dynamic HTML pages.

### Cost calculations use the **tkt\_price** stored in the ParkingLot model, ensuring the price per hour is dynamic, not hardcoded.

### Charts for Admin and User summaries are generated on-the-fly using **Matplotlib**, encoded to Base64 PNG, and embedded in the dashboard HTML.

### Architecture and Features

The project is organized as:

* **controllers.py** → All Flask routes, DB logic, and view functions.
* **templates/** → HTML templates: user/admin dashboards, forms.
* **static/** → CSS and optional static files.
* **Models** → Defined using SQLAlchemy ORM for clean DB operations.

**Features Implemented:**

* User authentication & registration.
* Search for lots by location/pin.
* Book first available slot.
* Dynamic slot status updates (Available/Occupied).
* Cost calculation using the tkt\_price stored for each lot.
* Reservation summary for users.
* Admin can add/edit/delete lots and slots.
* Revenue & occupancy summary for admin with simple charts.

### Video

<https://drive.google.com/file/d/1LuRmK7YSQVBkI-h8igXEZBtyupERBmfd/view?usp=sharing>