### Author

K Anandi Raghavi

23F3002074

[23f3002074@ds.study.iitm.ac.in](mailto:23f3002074@ds.study.iitm.ac.in)

I’m a 3rd Year CSE Student in GNITS, Hyderabad. I’m an enthusiast of technology and love exploring new fields to gain knowledge.

### Description

Vehicle Parking App project is a web application built on python framework that allows users to book a spot in a selected parking lot to place a 4-wheeler car, charging some amount on an hourly basis.

### Technologies used

The main tech stack used for this project is:

* Flask
* SQLite
* Jinja template

Some flask extensions used are:

* Flask-SQLAlchemy
* Flask-Bootstrap

### DB Schema Design

The database contains the following tables and columns:

* User: id (primary key), email, password, is\_admin (to check if its user or admin)
* ParkingLot: id (primary key), name, address, pincode, price\_per\_hour, max\_spots (all are not null)
* ParkingSpot: id (primary key), lot\_id (foreign key), spot\_number, status
* Reservations: id (primary key), spot\_id (foreign key), lot\_id (foreign key), vehicle\_number, parking\_time, leaving\_time, total\_cost

Relationships:

* reservations: One-to-many with Reservations, links users and their reservations
* spots: one-to-many with ParkingSpot, link between spots and lots
* lot: many-to-one with ParkingLot, allows reverse lookup

Constraints:

* unique constraint on username and email ensures no duplicate accounts.
* is\_admin allows role-based access.
* Relationship with Reservation tracks all bookings made by the user.

### API Design

### In this project, API endpoints were implemented using Flask’s routing system to manage users, parking lots, and spot reservations. Routes were created for core CRUD operations such as user registration/login, creating and deleting parking lots, reserving and releasing parking spots, and viewing dashboard summaries. These routes handle HTTP methods (GET, POST) and interact with a SQLite database using SQLAlchemy ORM. Responses are typically rendered as HTML pages, but can be extended to return JSON for a RESTful API.

### Architecture and Features

The main running code of the project is in app.py, which imports flask and sqlite. Templates folder consists of a user and admin folder containing html/jinja web pages for user and admin respectively. Base.html is the main template which is extended by other files. App.py contains “routes” to enable the webpages and functions. For example, the route app.route(‘admin/create\_lot’) leads to a function that is enabled with this url. Instance folder has the database file storing details of users.

This application implements feature to register, login, book and release spot by user and create, modify and delete lots along with viewing user data by admin.

### Video

https://drive.google.com/file/d/1wgHA7FPXUzOEFGp\_T8yJ32SqtROEyHun/view?usp=drive\_link