MAD 1 PROJECT REPORT

TruLot Parking App - Find, Book and Park — Effortlessly

ABOUT AUTHOR

Name – Shivam
Roll Number – 24f1002692
Student Email – 24f1002692@ds.study.iitm.ac.in
Course – Mad 1 project

As part of the **Diploma in Programming**, I've taken up the **App Development Project**, focusing on building a real-world solution that demonstrates both backend logic and frontend usability through a complete web application.

Approach to App Development Project :

To build this system effectively, the development was carried out in structured phases:

1. **User Authentication System:**
    - The project began with the creation of user login and signup forms.
    - Implemented secure authentication to differentiate between regular users and administrators based on the role.
2. **Admin Dashboard Development:**
    - Developed first to focus on the core functionalities like creating, editing, and monitoring parking lots and slots.
    - Integrated features for viewing user activity, managing bookings, and generating analytics dashboards.
3. **User Dashboard Development:**
    - Designed next to allow registered users to browse available lots, book parking slots, and manage their reservations.
    - Included real-time availability display and OTP-based verification for added security

---

PROJECT OVERVIEW

This project is a web-based **Four-Wheeler Car Parking System**, developed as part of the **Modern Application Development 1** course

The system enables an **Admin** to create and manage multiple parking lots, each with a any number of spots. Admins can track spot availability (free, occupied, or under maintenance(extra feature)), monitor each user active & completed reservations, and view detailed reservation information. Admin can also delete a parking lot only if the every spot of the lot is unoccupied, can also view the summary charts of the monthly and daily earning, can also track the daily progress of the TruLot.
The system also includes user authentication while signup, also enabling users to book parking spots with a one-time mobile number verification. Once a user is done using the spot, they can release it and proceed to pay the bill.
Additionally, I've implemented a **Pending Bills** section, where any unpaid or "pay later" bills are listed. Users must clear all pending dues before making a new reservation,
The system provides detailed insights by displaying **monthly**, **daily**, and **lot-wise** expenses through interactive charts, helping users track and manage their parking spendings effectively.

TECHNOLOGY & MODULE USED

TECHNOLOGY & MODULE USED	PURPOSE / USAGE
PYTHON	Core programming language also used to develop backend logic.
FLASK	Micro web framework for routing, server-side rendering, and API handling. Basic flask utilities we are using are render_template, jsonify, blueprint, session, url_for . . . etc

TECHNOLOGY & MODULE USED	PURPOSE / USAGE
FLASK-SQL ALCHEMY	ORM (Object Relational Mapper) used to manage database models with SQLite. ORM is a way to interact with the database without using raw SQL Queries
SQLITE3	Lightweight relational database for storing application data.
HTML	Used for structuring the frontend pages.
CSS	Used for styling and designing the frontend layout.
JAVASCRIPT	Adds interactivity to frontend elements such as buttons and forms through DOM manipulation also used to hit external & internal backend Api's.
CHART.JS	JavaScript library used to render dynamic charts for data visualization.
JINJA 2	A templating engine used in Flask to dynamically render HTML pages with Python variables and logic.
REDIS (REDIS - CLIENT)	In the project, we use redis-client specifically for **rate limiting** and **protecting services like OTP verification, Track the number of OTP requests** made by a user within a given time (like 1 hour). **Block further OTP requests** if a limit is exceeded (e.g., 3–5 OTPs per hour). Also, **Auto-expire the key** after a time period using Redis TTL (Time To Live)

TECHNOLOGY & MODULE USED	PURPOSE / USAGE
SIB-API-V3-SDK	Sendinblue (Brevo) SDK for sending OTPs or emails via API. sib-api-v3-sdk Python module connects the backend with Brevo's platform to send OTPs and other automated emails securely and efficiently.
PYJWT	Handles JSON Web Token (JWT) generation and verification for secure sessions.
PYDANTIC	Used for validating and parsing structured data models on server side.
REQUESTS	Sends HTTP requests to external APIs or services from Python server.
APSCHEDULER	Scheduler library for running tasks at regular intervals (e.g., auto cleanups) in background.
BASIC MODULES	random module, re module(regex), collection.defaultdict, calender(in charts), phonenumbers , unquote (Python's urllib.parse)

EXTERNAL API USED IN THE PROJECT

OpenCage Geocoding API

Validate user-entered addresses by checking if the location exists.
Convert textual addresses (like “Kidwai Nagar, Kanpur”) into precise **latitude and longitude** coordinates (but not so precised with local locations).

Brevo (Sendinblue) API

Used for sending **OTPs and verification emails, password reset** links to the users via the sib-api-v3-sdk module. It ensures secure, reliable delivery of transactional emails during signup or verification processes. (used in signup route for email verification, and in login route to reset the password of the user, if user forgets his/ her password).

UnSplash API

Used to dynamically fetch relevant, **high-quality images** for newly created parking lots, enhancing visual appeal. It is also utilized to assign a **gender-specific** profile image to users, providing a more personalized user interface experience.

Twilio API

Used to send SMS-based OTPs or alerts to users' mobile numbers. I Integrated it in my project to send OTPs via WhatsApp.
Due to free-tier limitations, users must first join the service manually by sending "**join bush-gradually**" through a provided link. Once connected, users can request their OTP by messaging "**what is my otp**." Verified mobile numbers are stored in the database to skip re-verification for future reservations unless a new number is used.

Ngrok Tunnel – ngrok http port (free service)

Ngrok is a tool that allows you to expose a local server (like a web app running on `localhost:5000`) to the internet through a secure tunnel.

This is helpful for:

- Testing webhooks from third-party services (Twilio in the project), where Twilio response my server that he has sent the OTP, so that my frontend will when to switch the DOM (change the page state).

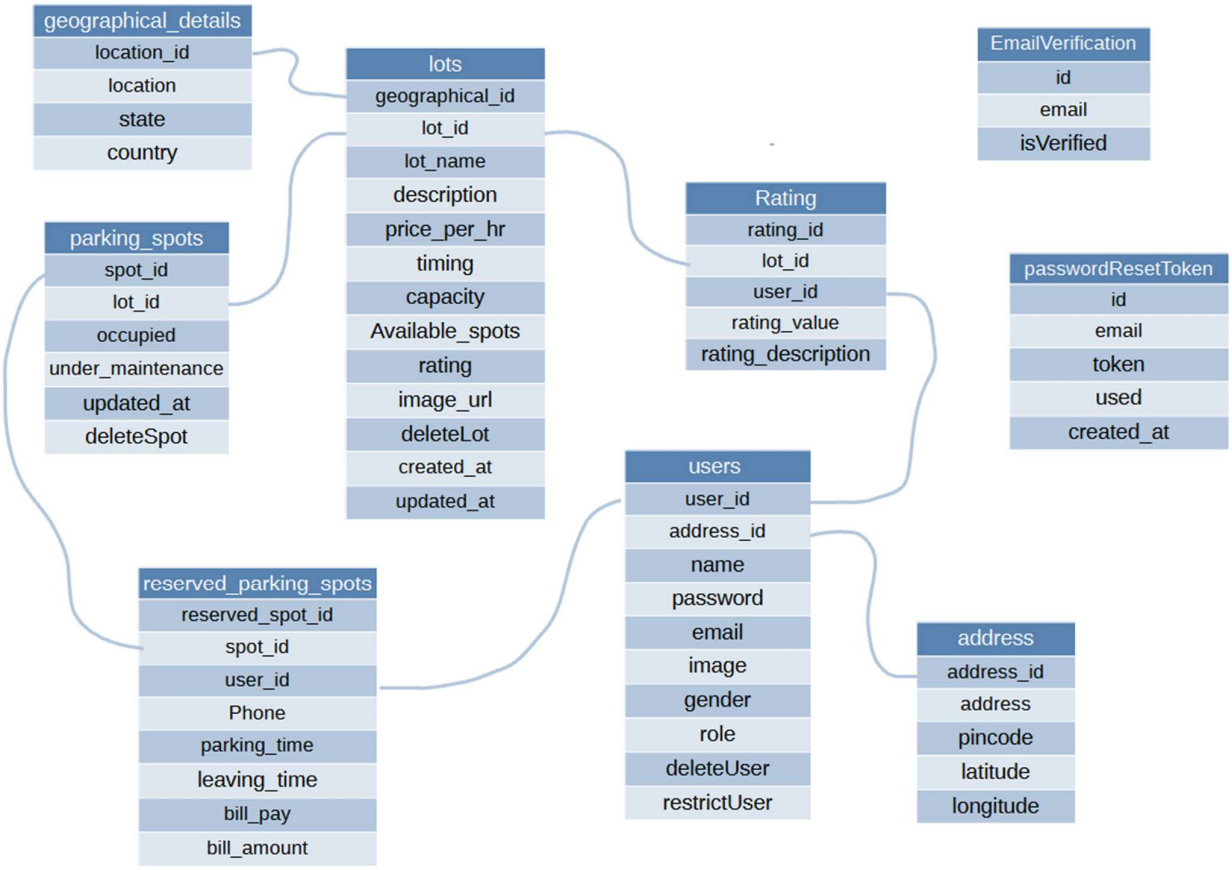
Ngrok is ideal for development and testing environments. However, one major drawback is that:

> **Each time you run `ngrok http 5000`, a new public URL is generated.**

This becomes inconvenient for services like Twilio Sandbox, where you need to repeatedly update the webhook URL.

DATABASE DESIGN – SCHEMA / MODELS

The system uses **SQLite3** as its backend database to store and manage essential information.
The schema is designed to support scalability and efficient querying.



`geographical_details`

- Stores location, state, and country of the parking Lot.
- **Relationships:**
    - Connected to `lots table` via '`geographical_id`' — and each lot is tied to a specific geographical location.

🅿️ `lots`

- Stores details of each parking lot including: name, description, price per hour, timing, capacity, available spots, ratings, and images.
- **Key Features:**
    - `deleteLot` flag for soft deletion.
- **Relationships:**
    - Linked to specific `geographical_details` via `geographical_id`.
    - Connected to `parking_spots`, `Rating`, and indirectly to `reserved_parking_spots`.
    -

🚙 3. `parking_spots`

- Represents individual spots within each lot.
- Stores details `occupied, available and under_maintenance.`
- **Relationships:**
    - Each spot belongs to a lot (`lot_id`).
    - Related to `reserved_parking_spots` via `spot_id`.

🗓️ 4. `reserved_parking_spots`

- Stores reservation details including: user ID, spot ID, parking time, leaving time, and bill info.
- Also stores the phone number used during one-time verification.
- **Relationships:**
    - Linked to `users` via `user_id`.
    - Linked to `parking_spots` via `spot_id.`

👤 `users`

- Core user profile table with fields like name, email, password, image, gender, and role (e.g., user/admin).
- Flags for `deleteUser` and `restrictUser`.
- **Relationships:**
    - Linked to `address` via `address_id`.
    - Linked to `reserved_parking_spots` and `Rating`.

🏠 `address`

- Stores the full address details including: pincode, latitude, and longitude.
- **Relationships:**
    - Connected to `users` via `address_id`.

⭐ `Rating`

- Users can rate each parking lot.
- Fields include: rating value and description.
- **Relationships:**
    - Linked to both `users` and `lots`.

📧 `EmailVerification`

- Stores verification status of user emails.
- **Fields:** email and verification flag (`isVerified`).
- Likely used during signup or login.

🔒 `passwordResetToken`

- Handles password reset flow with fields: email, token, usage status, and timestamp.
- Helps in managing secure and time-bound resets.

🚀 API RESOURCE SUMMARY

🔐 **Signup-Login Forms,** Authentication & User Management

File	Purpose
`login_route.py`	Handles user login requests and session creation.
`signup_route.py`	Manages user registration with validations.
`generators.py`	Utility functions to generate JWT token and also to decode it.
`validate_phone.py`	Validates and sanitizes user phone number input.
`validate_address.py`	Validates location/address data, possibly via OpenCage API.

👩‍💼 Admin Endpoints

File	Purpose
`createLot.py`	Allows admin to create new parking lots and define slots.
`registered_users.py`	Lists all users registered on the platform.
`spotDetails.py`	Displays detailed info about individual parking slots.
`admin_summaryChart.py`	Provides statistical data to generate admin dashboard charts.
`generateImages.py`	Generates relevant images for parking lots using Unsplash or similar services.

👥 User Features

File	Purpose
`myProfile.py`	Displays and manages user profile data including profile image.
`reservedSpots.py`	Shows currently and previously booked parking slots.
`view_book_spot.py`	Lets users make new bookings.
`ratings.py`	Allows users to rate their parking experience.
`user_summaryChart.py`	Sends statistical data for visualizations on the user dashboard.

📜 **Middlewares** and Landing Page

File	Purpose
`landing_route.py`	Manages homepage or landing route logic.
`check_authorisation.py`	Middleware to verify user authentication tokens or session status.
`validate_form.py`	Validates form submissions for consistency an

PROJECT VIDEO LINK – PRESS THE LINK BELOW

HTTPS://DRIVE.GOOGLE.COM/FILE/D/1-GH43LKBHTLIPBTO4BGZN1QQI1FT23A5/VIEW?USP=DRIVE_LINK