Project Report

Vehicle Parking App

Author:

Name: Devojyoti Misra
Roll No.: DS24F1002239
E-mail: 24f1002239@ds.study.iitm.ac.in

I am currently majoring at IIT Madras, in Data Science and Application. I am very enthusiastic about Machine Learning, Artificial Intelligence and Data Science.

Description:

App Name: Parkify
Description: A web application that is hereto solve modern problems related to parking, this application supports multiple types of users and is quite interactive. LLMs were lightly consulted to aid in debugging and fine-tuning.

Technologies Used:

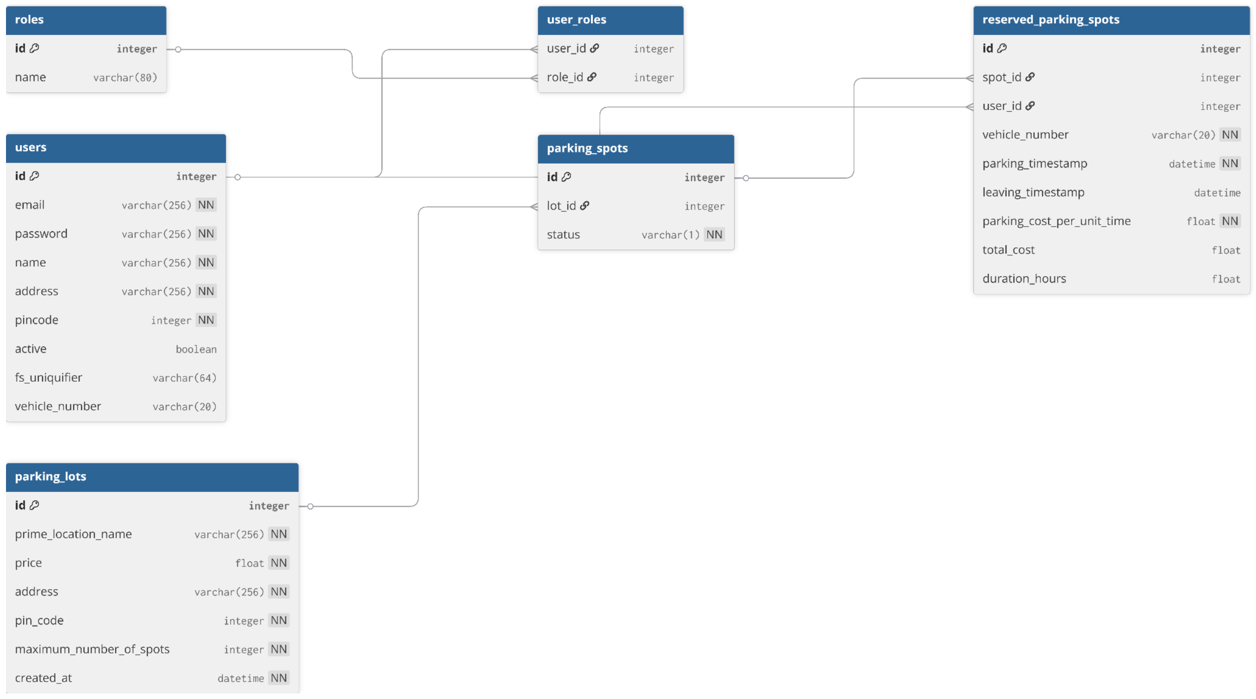
1. **Flask**: A lightweight Python web framework used for building the core backend of the application.
2. **SQLAlchemy**: An Object-Relational Mapping (ORM) library that facilitates seamless interaction between Python objects and the database.
3. **SQLite3**: A lightweight, file-based relational database used for storing application data.
4. **Flask-Login**: A user session management extension for handling authentication and login functionality.
5. **Werkzeug**: A comprehensive WSGI utility library used for password hashing and security-related operations.
6. **Chart.js**: A JavaScript library for creating responsive, interactive charts and data visualizations.
7. **Jinja2**: A templating engine for dynamically generating HTML content from backend data.
8. **Vanilla CSS + Bootstrap 5**: Used for styling and responsive design, combining custom styles with Bootstrap's component-based framework.
9. **JavaScript**: Enhances front-end interactivity and handles dynamic UI behavior.

API Design:

We have implemented APIs for a smart parking management system, divided across different user roles (Admin and User). The APIs enable various functionalities such as managing parking lots, reserving spots, searching for availability, and viewing booking history.
Elements for which APIs were created:

1. **Authentication APIs**:
    a. /login
    b. /logout
    c. /signup
2. **Admin APIs** (Few Examples):
    a. Add a parking lot: POST /dash/admin/add_lot
    b. Edit a parking lot: POST /dash/admin/edit_lot/<int:lot_id>
    c. Delete a parking lot: POST /dash/admin/delete_lot/<int:lot_id>
3. **User APIs** (Few Examples):
    a. Book a slot: POST /dash/user/book
    b. Release a slot: POST /dash/user/release

DB Model:



The schema of the database consists of six main relations: users, roles, user_roles, parking_lots, parking_spots, and reserved_parking_spots. These relations are interconnected through foreign key constraints, reflecting a well-structured relational model designed for a parking reservation system.

The roles and user_roles tables enable role-based access control. Users can have one or more roles through the many-to-many mapping defined in user_roles.

The users table captures essential user data, including email, hashed password, name, address, and pincode. Additional optional fields such as vehicle_number allow users to register their vehicles, while fields like fs_uniquifier are used for secure identification. The email and vehicle_number fields are enforced to be unique. Passwords are stored in hashed form, so the schema allows up to 256 characters to accommodate hash outputs.

The parking_lots table represents high-level parking locations with information like price, address, and maximum capacity. Each lot is assigned a creation timestamp (created_at) to record when it was added to the system.

The parking_spots table tracks individual spots within each lot. A spot's status field indicates availability (A for available by default).

The reserved_parking_spots table keeps a record of parking spot reservations. It links users, spots, and vehicles, and captures reservation metadata such as parking_timestamp, leaving_timestamp, duration_hours, and total_cost.

All datetime fields, such as created_at and parking_timestamp, are timezone-aware and default to IST (UTC+5:30) where applicable. Constraints like not null, unique, and foreign key references ensure data integrity, while application-level logic enforces additional validations.

Architecture:

```
root/.
│   .env
│   .env.sample
│   .gitignore
│   .python-version
│   api.yaml
│   app.py
│   Project Report.pdf
│   pyproject.toml
│   README.md
│   uv.lock
│
├───application
│   ├───extensions
│   │       db_extn.py
│   │       security_extn.py
│   │       session_extn.py
│   │
│   ├───helpers
│   │       config.py
│   │       models.py
│   │
│   ├───middlewares
│   │       init_cache_control.py
│   │       init_db.py
│   │       init_error.py
│   │       init_security.py
│   │       init_session.py
│   │
│   └───routes
│       ├───admin
│       │       admin_add_lot_route.py
│       │       admin_dash_route.py
│       │       admin_delete_lot_route.py
│       │       admin_edit_lot_route.py
│       │       admin_history_route.py
│       │       admin_profile_edit_route.py
│       │       admin_profile_route.py
│       │       admin_search_route.py
│       │       admin_view_spots_route.py
│       │       admin_view_spot_details_route.py
│       │       admin_view_users_route.py
│       │
│       ├───general
│       │       home_route.py
│       │       login_route.py
│       │       logout_route.py
│       │       signup_route.py
│       │
│       └───user
│               user_book_slot_route.py
│               user_dash_route.py
│               user_history_route.py
│               user_profile_edit_route.py
│               user_profile_route.py
│               user_release_slot_route.py
│               user_search_route.py
│               user_select_slot_route.py
│
├───instance
│       parkify_db.sqlite3
│
├───static
│   ├───assets
│   │       logo.png
│   │
│   ├───css
│   │       select_parking_space.css
│   │
│   └───js
│           admin_chart.js
│           auto_hide_alert.js
│           select_spot.js
│           user_chart.js
│
└───templates
    ├───admin
    │       admin_add_parking_lot.html
    │       admin_dashboard.html
    │       admin_edit_parking_lot.html
    │       admin_edit_profile.html
    │       admin_history.html
    │       admin_list_users.html
    │       admin_parking_spot_overview.html
    │       admin_profile.html
    │       admin_search_results.html
    │       admin_single_spot_details.html
    │
    ├───components
    │       alerts.html
    │       base.html
    │       navbar.html
    │
    ├───general
    │       error.html
    │       index.html
    │       login.html
    │       signup.html
    │
    └───user
            user_dashboard.html
            user_edit_profile.html
            user_history.html
            user_profile.html
            user_release_spot.html
            user_search_parking.html
            user_select_parking_space.html
```

Above is the directory structure of the flask application.

Features at a Glance:

1. **Roles:**
    a. **Admin**: Has root access (no registration required). Manages parking lots, views users, monitors parking spot statuses, and accesses summary charts.
    b. **User**: Registers/logs in, reserves parking spots, views personal parking history, and interacts with the system via the user dashboard.
2. **Core Functionalities:**
    a. **Authentication:**
        i. Admin and User login system.
        ii. Separate login/register forms for users; only login for admin.
        iii. User credentials and roles stored using appropriate models.
    b. **Admin Dashboard:**
        i. Auto-created admin when DB is initialized.
        ii. Create/Edit/Delete parking lots (delete only if all spots are vacant).
        iii. Bulk generation of parking spots based on lot capacity.
        iv. View real-time status of all parking spots and parked vehicle details.
        v. View all registered users.
        vi. Visualize lot/spot usage through summary charts.
    c. **User Dashboard:**
        i. Selects a parking lot; assigned first available spot (no spot-level selection).
        ii. Marks spot as "Occupied" upon parking; updates to "Released" when leaving.
        iii. Duration and timestamps are recorded for billing and history.
        iv. Sees charts summarizing personal parking activity.
3. **Optional / Recommended Features:**
    a. **Spot Search**: Admin can search parking spots by ID/status.
    b. **APIs**: RESTful endpoints for users, lots, and spots using JSON
    c. **Charts**: Visual analytics using Chart.js.
    d. **Validation**: Backend (Flask) form validations.
    e. **Styling**: Responsive UI with Bootstrap and custom CSS.
    f. **Secure Login**: Use flask_login, flask_security for proper auth management.
    g. **Extendability**: Open for additional features like payment modules, email alerts, etc.

Video:

mad1-project-explainer-vieo.mkv