Author

Vaibhav Satish

21F3002068

21F3002068@ds.iitm.ac.in

I completed my BSc in Statistics from Aligarh Muslim University. I am currently a standalone BS student. I am deeply interested in Responsible AI Research and plan to do research and publish a paper in this field by next year.

Description

This project implements a multi-user web application that allows administrators to manage 4-wheeler parking lots and users to reserve parking spots. The admin can create parking lots with multiple parking spots and monitor usage, while users can register, book the first available spot, and release it after use.

Technologies used

Flask: A Backend framework for routing, request handling, and business logic.

SQLite: Lightweight relational database for persistent data storage.

Jinja2: Flask's templating engine for dynamic rendering of HTML content.

HTML5, **CSS3**, **Bootstrap**: For structuring and styling the frontend to ensure responsiveness and a clean UI.

Flask Extensions:

- **flask_login**: To handle user session management.

- **flask_sqlalchemy**: ORM for database model definition and interaction.

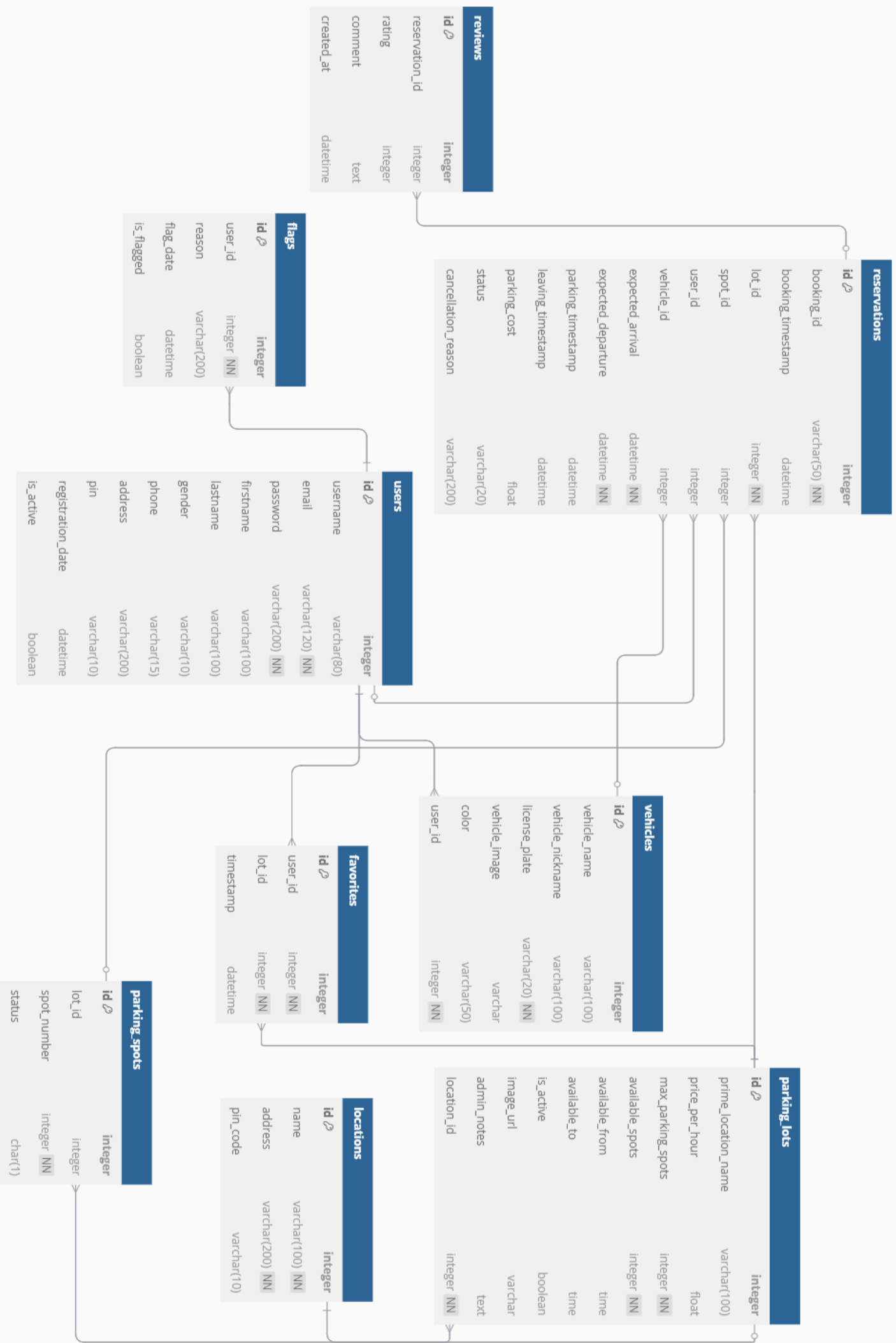- **Werkzeug Security**: Password hashing for secure user data.

Chart.js: For rendering user/admin dashboards with visual parking usage summaries.

datetime: Manage booking and parking time.

DB Schema Design

Entities and Relationship Diagram:

[on next page]

reviews

column	type	
id	integer	
reservation_id	integer	
rating	integer	
comment	text	
created_at	datetime	

reservations

column	type	
id	integer	
booking_id	varchar(50)	NN
booking_timestamp	datetime	
lot_id	integer	NN
spot_id	integer	
user_id	integer	
vehicle_id	integer	
expected_arrival	datetime	NN
expected_departure	datetime	
parking_timestamp	datetime	
leaving_timestamp	datetime	
parking_cost	float	
status	varchar(20)	
cancellation_reason	varchar(200)	

flags

column	type	
id	integer	
user_id	integer	NN
reason	varchar(200)	
flag_date	datetime	
is_flagged	boolean	

users

column	type	
id	integer	
username	varchar(80)	
email	varchar(120)	NN
password	varchar(200)	NN
firstname	varchar(100)	
lastname	varchar(100)	
gender	varchar(10)	
phone	varchar(15)	
address	varchar(200)	
pin	varchar(10)	
registration_date	datetime	
is_active	boolean	

vehicles

column	type	
id	integer	
vehicle_name	varchar(100)	
vehicle_nickname	varchar(100)	
license_plate	varchar(20)	NN
vehicle_image	varchar	
color	varchar(50)	
user_id	integer	NN

favorites

column	type	
id	integer	
user_id	integer	NN
lot_id	integer	NN
timestamp	datetime	

parking_spots

column	type	
id	integer	
lot_id	integer	
spot_number	integer	NN
status	char(1)	

parking_lots

column	type	
id	integer	
prime_location_name	varchar(100)	
price_per_hour	float	
max_parking_spots	integer	NN
available_spots	integer	NN
available_from	time	
available_to	time	
is_active	boolean	
image_url	varchar	
admin_notes	text	
location_id	integer	NN

locations

column	type	
id	integer	
name	varchar(100)	NN
address	varchar(200)	NN
pin_code	varchar(10)	

dbdiagram

Entities and Relationships:

- **User** (id, username, email, password, firstname, lastname, gender, phone, address, pin, registration_date, is_active)

    - One-to-Many with **Vehicle**
    - One-to-Many with **Favorite**
    - One-to-Many with **Reservation**
    - One-to-Many with **Flag**
    - **Purpose:** User to register with personal details and manage multiple Vehicles, mark multiple Favorite parking lots, make multiple Reservations, and may be Flagged multiple times for misconduct.

- **Vehicle** (id, vehicle_name, license_plate, color, user_id)

    - One-to-Many with **Reservation**

    - **Purpose**: Each **Vehicle** is to have multiple Reservations.

- **Location** (id, name, address, pin_code)

    - One-to-Many with **ParkingLot**
    - **Purpose**: Location to host multiple **ParkingLots**.

- **ParkingLot** (id, prime_location_name, price_per_hour, max_parking_spots, available_spots, available_from, available_to, is_active, location_id, latitude, longitude, image_url, admin_notes)

    - One-to-Many with **ParkingSpot**
    - One-to-Many with **Reservation**
    - One-to-Many with **Favorite**
    - **Purpose:** Each **ParkingLot** to have multiple **ParkingSpots** and many Reservations.

- **ParkingSpot** (id, lot_id, spot_number, status)

    - One-to-Many with **Reservation**
    - **Purpose**: A parking spot to have multiple reservations.

- **Reservation** (id, booking_id, booking_timestamp, lot_id, spot_id, user_id, vehicle_id, expected_arrival, expected_departure, parking_timestamp, leaving_timestamp, parking_cost, status, cancellation_reason)

    - One-to-One with **Review**
    - **Purpose:** Each Reservation is to receive one Review.

Architecture and Features

```
park_ease_21f3002068
├── app.py
├── config.py
├── utils.py
├── instance
│   └── parkease.db
├── model.py
├── Project Report.pdf
├── README.md
├── requirements.txt
├── routes
│   ├── admin_routes.py
│   ├── __init__.py
│   └── user_routes.py
├── static
│   ├── icon
│   ├── image
│   ├── scripts
│   ├── style
│   └── uploads
└── templates
    ├── admin
    ├── index.html
    ├── partials
    └── user
```

- **app.py** - Main application entry point.

- **/templates** - All HTML templates (Admin dashboard, User dashboard, Login/Register, Lot management, Spot management).

- **/static** - Static files (CSS, Bootstrap assets, JS).

- **/model.py** - All SQLAlchemy ORM models (User, ParkingLot, ParkingSpot, Reservation).

- **/routes.py** - All application routes (User, Admin, CRUD operations, booking/reserving).

- **/config.py** - Database and environment configurations.

Features Implemented

- **Authentication**
  - **Admin login** with hardcoded credentials.
  - User **registration** and **login**.
  - **Passwords hashed** securely using Werkzeug.

- **Admin Features**
  - **View** all parking lots.
  - **Create** new parking lots with admin assigned number of spots.
  - **Edit/Delete** parking lots.
  - **Delete** parking spots.
  - **View** all parking spots.
  - **View** registered users and their parking reservations.
  - **View** summary charts

- **User Features**
  - **View** available parking lots.
  - **Reserve** the first available spot in the selected lot.
  - **View** and **manage** current parking status.
  - **Release** parking spot and **generate** parking cost based on duration.
  - **View** parking history and **summary charts**.

- **Additional features**
  - **Display flash messages on top-right corner of the screen.**
  - **Admin**
    - **Flag/Unflag** users.
    - **Flagged** user will be unable to login to their dashboard.
    - **Delete user**.
    - Create **multiple parking lots** for a location.
    - **Print out** parked-out parking reservation details of a users
  - **User**
    - Profile completion bar.
    - **Flagged user will be unable to login to their dashboard.**
    - **Cancel** parking reservations with statuses '**Pending**' and '**Confirmed**'.
    - The reservation status '**Rejected**' in cases of no-show.
    - **Re-book** a parking lot reservation with status '**Cancelled**/**Rejected**' or '**Parked Out**'.
    - **Favorite/Save** a parking lot.
    - Add **multiple vehicles** to their profile
    - **Print out the** parked-out parking reservation details.
    - **Delete** their account

Video

DriveLink:

https://drive.google.com/file/d/1PNSmYSB8zykQxnyztq75t_j98t2MALwi/view?usp=sharing

YTLink (in case the drive link takes a longer loading time):

https://youtu.be/Ly-Wm7Tt9Ck