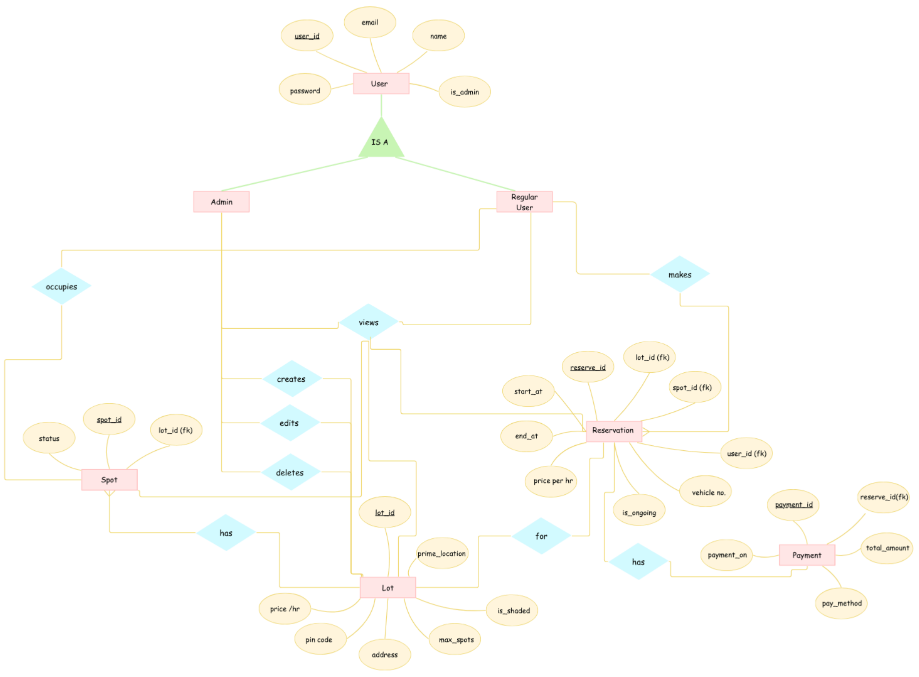
MAD-1 T22025 Project Report
Project Statement - Vehicle Parking Application

Author:
Name: Priyal Pandey
Roll No: 23f2005558
Email: 23f2005558@ds.study.iitm.ac.in
I am a dual-degree student, studying Information Technology at PICT (Pune) and Data Science and Applications at IITM. I enjoy problem-solving and working on full-stack web applications. I also have a keen interest in creative frontend design, as well as data science and analytics, which I'm exploring through my academic coursework and projects.

Description:
The project, named '**Lot And Found'** is a multi-user **Vehicle Parking Application** built with **Python (Flask)**, that manages parking lots, spots and parked vehicles. Users can sign up, view available spots, and book a lot based on location and availability, with the cost calculated based on hourly rates. An **Admin** role, created at app initialization, can perform CRUD operations on parking lots. Both admins and users can also view summary charts for insights into parking activity.
AI LLM Use- 5% for Testing/Debugging, Charts and understanding API Integration

Technologies Used:
Backend: Python (Flask)
SQLite (Flask-SQLALchemy)
Frontend: HTML, Jinja2, Bootstrap and Vanilla CSS
Summary Charts: Chart.js

DB Schema Design:



1. **User**
    - user_id (INTEGER, PRIMARY KEY, AUTO INCREMENT))
    - name (String)
    - email (String, UNIQUE, NOT NULL)
    - password (String, NOT NULL)
    - is_admin (Boolean, NOT NULL, default = False)

2. **Lot**
    - lot_id (INTEGER, PRIMARY KEY, AUTO INCREMENT)
    - prime_loc (String, NOT NULL)
    - address (String, NOT NULL)
    - pincode(String, NOT NULL)
    - price_per_hr (Double, NOT NULL)
    - max_spots (INTEGER, NOT NULL)
    - is_shaded (Boolean, NOT NULL)

3. **Spot**
    - spot_id (INTEGER, PRIMARY KEY, AUTO INCREMENT)
    - lot_id (INTEGER, FK References Lot(lot_id), NOT NULL)
    - status (CHAR(1), default='a', NOT NULL)

4. **Reserve**
    - reserve_id (INTEGER, PRIMARY KEY, AUTO INCREMENT)
    - lot_id (INTEGER, FK References Lot(lot_id), NOT NULL)
    - spot_id (INTEGER, FK References Spot(spot_id), NOT NULL)
    - user_id (INTEGER, FK References User(user_id), NOT NULL)
    - vehicle_num(String, NOT NULL)
    - start_time(DateTime, NOT NULL, default = datetime.now)
    - end_time(DateTime)
    - price_per_hr(Double, NOT NULL)
    - is_ongoing(Boolean, NOT NULL, default=True)

5. **Payment**
    - payment_id (INTEGER, PRIMARY KEY, AUTO INCREMENT)
    - reserve_id (INTEGER, FK References Reserve(reserve_id), NOT NULL)
    - total_amt (Double)
    - payment_method(String)
    - transaction_date (DateTime)

Reason for above DB Design

- **User**: Stores login credentials and roles (admin or user) with unique email-based authentication.
- **Lot**: Represents parking locations with pricing, address, and capacity details.
- **Spot**: Contains individual parking spot entries linked to specific lots, with real-time availability status.
- **Reserve**: Tracks user bookings with timing, pricing, and vehicle details for each reservation, and whether the reservation is ongoing or previously booked (is_ongoing)
- **Payment**: Logs transaction details for each booking, simulating billing and payment history.

Architecture and Features:

The project is structured with **app.py** handling setup, **config.py** for configuration variables, and **.env.sample** for environment variables. All controller logic and routes are defined in routes.py, and database models in **models.py** using SQLAlchemy. Templates (Jinja2) are organized in **templates/** with subfolders **admin/** and **user/** for separating admin and user-specific templates. Static assets like custom style.css and images are in **static/**. Dependencies are managed via venv and **requirements.txt**, with project details and setup instructions in **README.md**. A .gitignore excludes unnecessary files from version control.

Features Implemented:
The project was developed by completing **all core requirements** followed by select **optional enhancements**, as outlined in the milestone plan.

1. Authentication and Role-Based Access for admin and users - login and sign up forms
2. Unauthorized access prevention for all pages
3. Profile Editing for both admins and users
4. Admin Dashboard
    - Create new parking lots
    - Edit existing lots
    - View lot details
    - Delete lots(only if no spots booked)
    - View Individual Spot details - status, user_id, vehicle number if occupied
    - Search parking lots based on location, maximum price and shaded/open
    - View all registered users, search by name
    - Access parking and transaction history
    - Summary Chart for Spot available vs occupied using Chart.js
5. User Dashboard
    - Browse available parking lots - parameters like price, shaded/open, location etc.
    - Book a lot, spot is automatically allocated
    - Search parking lots based on location, maximum price and shaded/open
    - View live booking status
    - Release spots, cost calculation based on duration, payment portal
    - View parking history and transaction history
6. Responsive UI using Bootstrap and plain CSS
7. Frontend Validation using HTML5
8. Backend Validation using Flask and in routes

Video: MAD1_23f2005558.mp4