# PROJECT REPORT

**Author**
Name : Vignesh S
Roll Number : 23f3004120
Student Email : 23f3004120@ds.study.iitm.ac.in
I am currently at diploma level completing my first project.

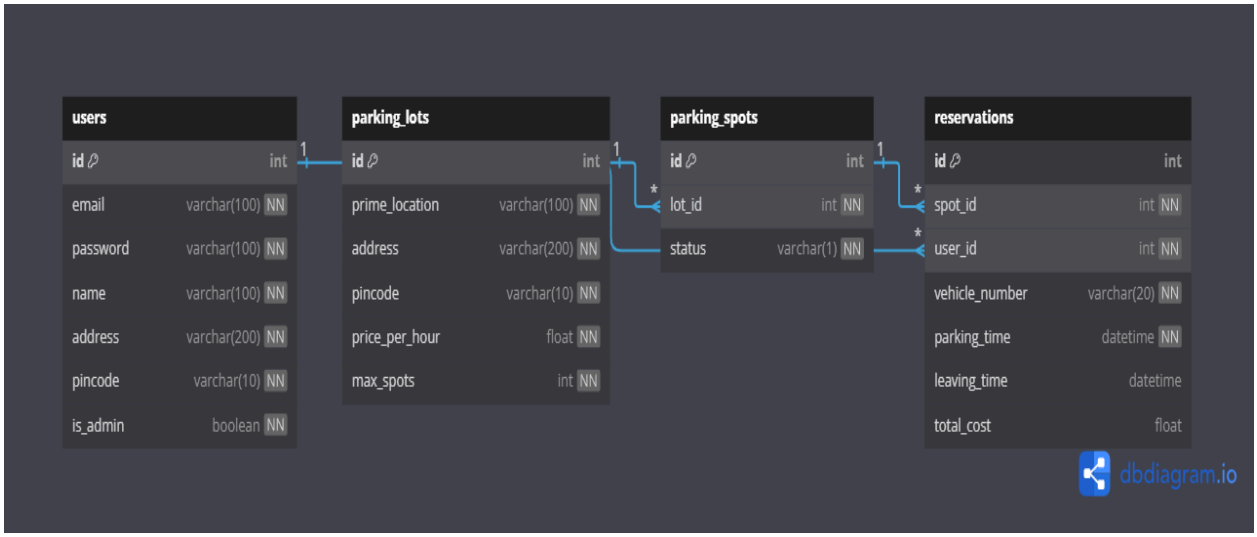
**Description**
This project is a web-based vehicle parking management system called **My-Spot**. It allows users to reserve and release parking spots while admins can manage parking lots and view summaries. The system tracks reservations, calculates costs, and displays real-time data using interactive charts.

**Technologies used**

1. **Flask**: Lightweight web framework used to build the web application.
2. **Flask-SQLAlchemy**: ORM (Object Relational Mapper) for managing database operations easily with Python classes.
3. **SQLite**: Lightweight relational database used to store user, parking lot, spot, and reservation data.
4. **HTML, CSS, Bootstrap**: Frontend technologies used to create responsive and user-friendly UI.
5. **Jinja2**: Templating engine used to render dynamic content in HTML pages.

**DB Schema Design**

All the classes have an id field as the primary key.




**Normalization**: Data is broken into separate tables to avoid redundancy (e.g., users and lots are not duplicated in reservations).

**Relationships**:

- One-to-many: A lot can have many spots.
- One-to-many: A spot can have multiple reservations (over time).

**API Design**

The application follows a modular API design using Flask routes and Blueprints. APIs were implemented for both **admin** and **user** functionalities.

- Routes are logically grouped using Blueprints: user_bp and admin_bp.
- Forms use POST method and render results using Jinja2 templates.
- Data interactions handled through SQLAlchemy ORM.
- Plotly is used for rendering responsive data visualizations in the admin and user summary pages.

**Architecture and Features**

The project follows the **MVC (Model-View-Controller)** architecture using Flask. The app is modularized with Blueprints for better separation of admin and user functionalities. The **controllers** (routes and logic) are organized inside separate files (admin.py, user.py) and registered in the main app.py file using register_controllers(app). The **models** are defined in a single models.py file using SQLAlchemy to manage all the database tables and relationships. All HTML pages are stored under the templates/ folder, grouped into admin/, user/, and shared templates. Static assets like CSS, images, and the logo are located in the static/ folder.

Features Implemented

- **User Authentication**: Handled using Flask-Login with login, logout, session, and role-based redirection (admin or user).
- **Admin Dashboard**: Admins can add, edit, and delete parking lots. Each lot contains a configurable number of parking spots.
- **Dynamic Spot Management**: Admin can view which spots are available or occupied. Spots are color-coded (green for available, red for occupied).
- **User Booking**: A user can reserve a parking spot. The system automatically assigns the first available one.
- **Release Parking**: Users can release a spot, and the system calculates duration and cost automatically.
- **Summary Dashboards**: Admin and user dashboards feature **Plotly bar charts** for visual insights (e.g., revenue by lot, user parking history).
- **Search Functionality**: Admin can search users by ID and lots by location.
- **Clean UI**: Implemented using **Bootstrap 5**. Forms are responsive and styled consistently.

**Declaration of AI Usage**

This project incorporates Artificial Intelligence components amounting to approximately **40%** of the overall implementation.

**Video**

MAD-1 Project demo video