Summit Motors Dealership

CS 4347 - Professor Jamal Omer

Team Members:
Angel Jimenez, Jair Gutierrez, Vuong Nguyen, Akhil Srirangam

Date: 11/30/25

TABLE OF CONTENTS

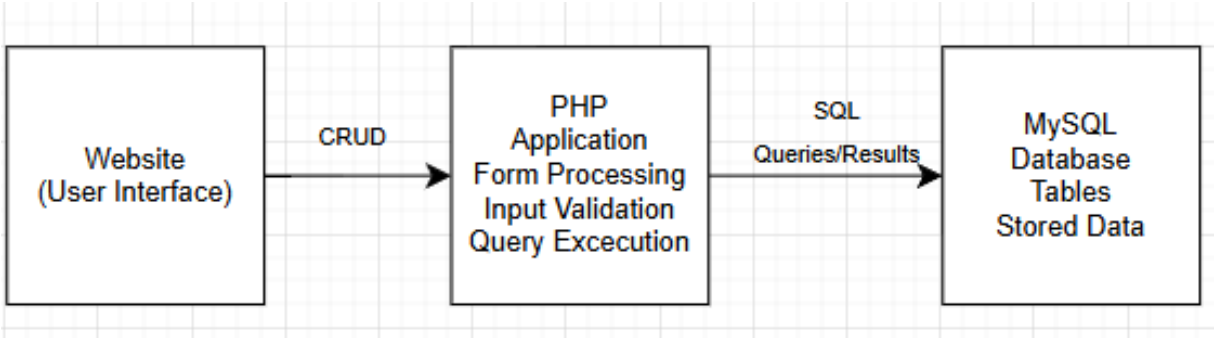
Introduction……..3
System Requirements……3
Conceptual Design of the Database……5
Logical Database
Schema……...11
Functional Dependencies and Database Normalization……....16
The Database System……19
Suggestions on Database Tuning……..23
User Application
Interface……24
Conclusion and Future Work……..27
References……27
Appendix……....27

Introduction

This project consists of developing a complete Car Dealership Database Management System capable of storing, modifying, and retrieving information related to customers, vehicles, employees, ownership, and service history.

System Requirements

System architecture diagram:



List the interface requirements of the system:

User Interface:

User Login
User Registraction
User home dashboard
Add Vehicle
Delete Vehicle
View Vehicle
Edit Vehicle Information

List the functional requirements of the database system:

User Authentication

- Register new users
- Prevent duplicate SSNs and emails
- Validate data format (SSN, phone number, VIN, etc.)

Customer Functions

- View/update personal information
- Add vehicles they own
- View all owned vehicles
- Add service requests
- View vehicle service histories

List non-functional requirements of the database system

Reliability

- CASCADE deletes where appropriate
- SET NULL where data should persist

Security

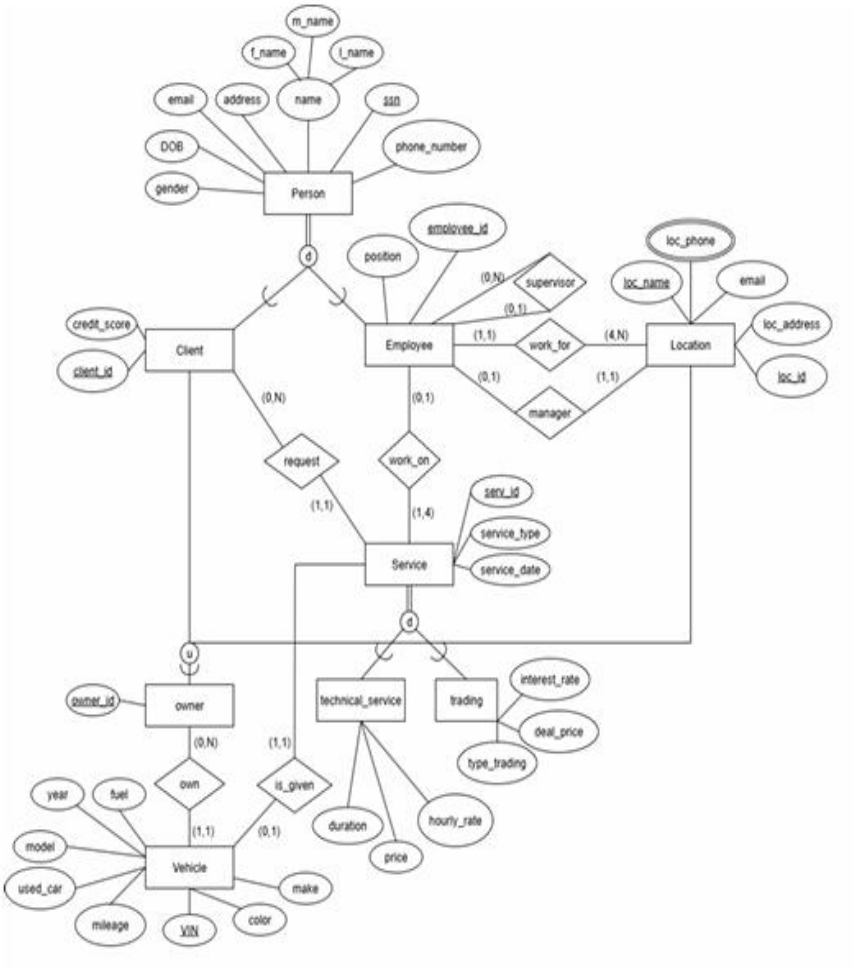
- Employ prepared SQL statements
- Use php sessions for log in

Scalability

- Support multi-location dealerships
- Allow thousands of service records

Conceptual Design of the Database

Entity-Relationship (ER) model:



Data dictionary

Person

Attribute	Data Type	Purpose	Default Value
ssn	char(9)	Identifier for a person	not null
f_name	varchar	First name	not null
m_intial	varchar	Middle initial	null
l_name	varchar	Last name	not null
address	varchar	Home address	null

DOB	DATE	Date of birth	null
gender	char(1)	gender	null
email	varchar	Email for a person	null
phone_no	varchar	Phone number of a person	null

Client

Attribute	Data Type	Purpose	Default Value
client_id	INT	Identifier for a client	not null
ssn	char(9)	Links client to personal record	not null
Credit_score	INT	Credit rating for client	null
owner_id	INT	Link to id of owner	null

Employee

Attribute	Data Type	Purpose	Default Value
Employee_id	INT	Identifier for employees	not null

ssn	char(9)	Links employee to personal records	not null
position	varchar	Job title/role	null
surpevisor_id	char(9)	SSN of employees supervisor	null
location_no	varchar	Links employee to the location they work at	not null
service_id	INT	Link employee to service they are currently working on	null

Location

Attribute	Data Type	Purpose	Default Value
loc_name	varchar	Name of location	not null
loc_no	INT	Identifier for the location	not null
loc_email	varchar	Contact email of location	null
loc_address	varchar	Physical address of location	null

manager_id	varchar	Manager of the location links to employee	not null
owner_id	INT	Location is the owner of a vehicle (until sold)	-

Location_Phone

Attribute	Data Type	Purpose	Default Value
loc_no	INT	Identifier of the location	not null
loc_phone	varchar	Location contact number	not null

Vehicle

Attribute	Data Type	Purpose	Default Value
VIN	char(17)	Identifier for the vehicle	not null
fuel	varchar	Fuel type of the vehicle	null
make	varchar	Manufacturer of the vehicle	null
model	varchar	Model of the vehicle	null
year	YEAR	Year of the car	null

color	varchar	Color of the car	null
milage	INT	Mileage of the vehicle	0
used	BOOLEAN	Is the vehicle used or new	false
owner_id	INT	Link to the person who owns the vehicle	null

Services

Attribute	Data Type	Purpose	Default Value
serv_id	INT	Identifier for the a service given	not null
serv_type	varchar	Type of service given	null
serv_date	DATE	Date service is performed	null
client_id	INT	Links the service to a client	not null
VIN	char(17)	Links the vehicle getting the service	null

Technical Service

Attribute	Data Type	Purpose	Default Value
serv_id	INT	Identifier for the service given links it with services	not null
duration	INT	Duration of the service in minutes	0
hourly_rate	DECIMAL (10,2)	Hourly labor rate for employee	0.00
price	DECIMAl (10,2)	Cost of the service	0.00

Trading

Attribute	Data Type	Purpose	Default Value
serv_id	INT	Identifier for the service given links it with services	not null
type_trading	varchar	Type of trade (buy/sell)	null
interest_rate	DECIMAL (4,2)	Interest rate applied to trade details	0.00
deal_price	DECIMAL (10,2)	Total deal amount	0.00

Owner

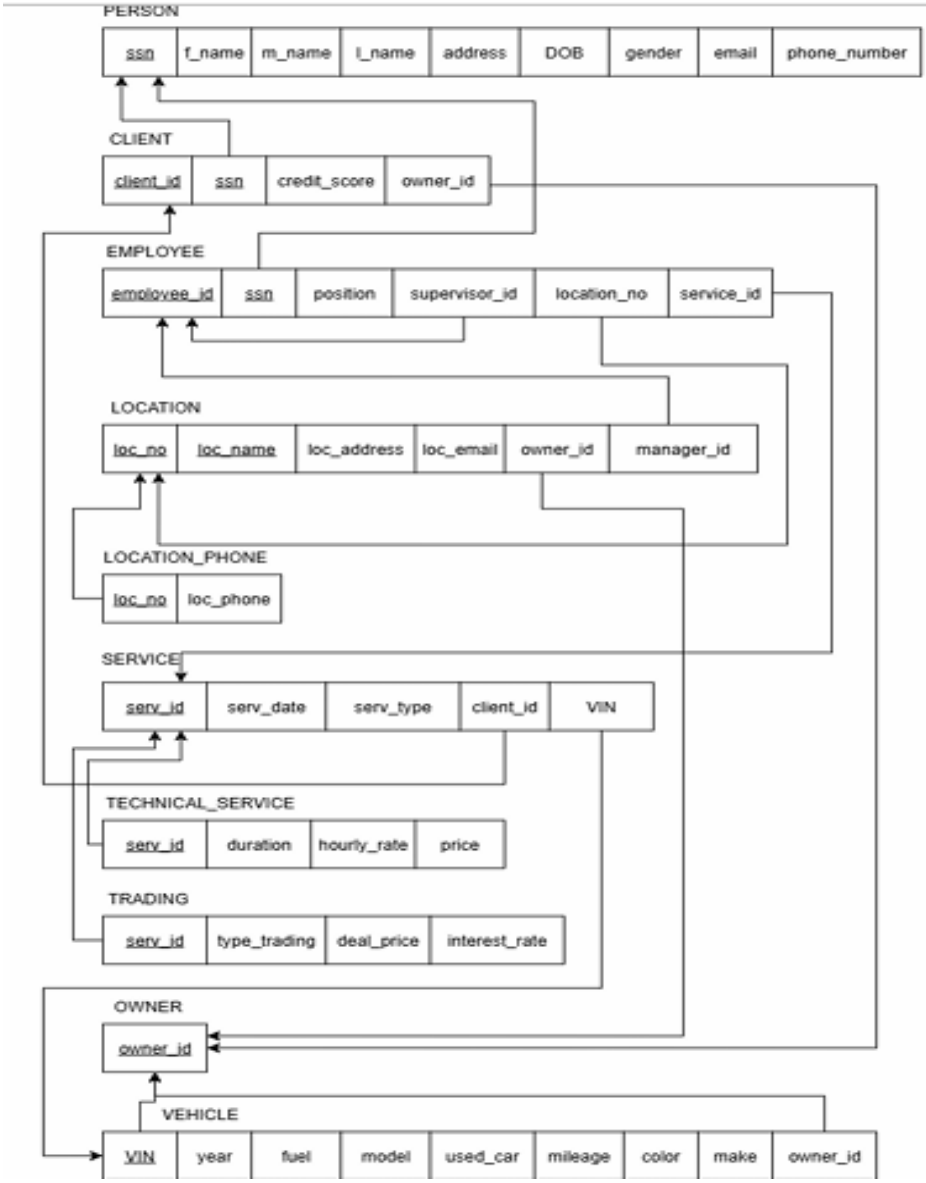
Attribute	Data Type	Purpose	Default Value
Owner_id	INT	Identifier of the owner of the vehicle (client / dealership)	not null

Business Rules

- A Person may be a Client or Employee.
- Each Client has an owner_id; deleting a Person cascades.
- A Vehicle belongs to an Owner, not directly a Person.
- A Service is linked to both a Client and a Vehicle.
- A Technical Service or Trading record is created only when a Service occurs.
- A Location may have multiple employees and phone numbers.
- Deleting a Person removes them as client or employee automatically.

Logical Database Schema

Relational Schema



SQL statements

```
CREATE DATABASE IF NOT EXISTS car_dealership;
USE car_dealership;

-- Create Person table
CREATE TABLE IF NOT EXISTS Person (
    ssn VARCHAR(11) PRIMARY KEY,
    f_name VARCHAR(50),
    m_intial VARCHAR(1),
    l_name VARCHAR(50),
    address VARCHAR(100),
    state VARCHAR(2),
```

```
    DOB VARCHAR(10),
    gender VARCHAR(1),
    email VARCHAR(100),
    phone VARCHAR(12)
);

-- Create owner table
CREATE TABLE IF NOT EXISTS owner (
    owner_id INT PRIMARY KEY
);

-- Create Location table
CREATE TABLE IF NOT EXISTS Location (
    loc_name VARCHAR(50),
    loc_no INT PRIMARY KEY,
    loc_email VARCHAR(100),
    loc_address VARCHAR(100),
    state VARCHAR(2),
    manager_id INT,
    owner_id INT
);

-- Create Location_Phone table
CREATE TABLE IF NOT EXISTS Location_Phone (
    loc INT,
    loc_phone VARCHAR(12),
    FOREIGN KEY (loc) REFERENCES Location(loc_no)
);

-- Create Client table
CREATE TABLE IF NOT EXISTS Client (
    client_id INT PRIMARY KEY,
    ssn VARCHAR(11),
    credit_score INT,
    owner_id INT,
    FOREIGN KEY (ssn) REFERENCES Person(ssn),
    FOREIGN KEY (owner_id) REFERENCES owner(owner_id)
);

-- Create Employee table
CREATE TABLE IF NOT EXISTS Employee (
    Employee_id INT PRIMARY KEY,
    ssn VARCHAR(11),
    position VARCHAR(20),
```

```
    supervisor_id INT,
    location_no INT,
    service_id INT,
    FOREIGN KEY (ssn) REFERENCES Person(ssn),
    FOREIGN KEY (location_no) REFERENCES Location(loc_no)
);

-- Create Vehicle table
CREATE TABLE IF NOT EXISTS Vehicle (
    VIN VARCHAR(17) PRIMARY KEY,
    fuel VARCHAR(20),
    make VARCHAR(50),
    model VARCHAR(1),
    year INT,
    color VARCHAR(20),
    milage INT,
    used BOOLEAN,
    owner_id INT,
    FOREIGN KEY (owner_id) REFERENCES owner(owner_id)
);

-- Create Service table
CREATE TABLE IF NOT EXISTS Service (
    serv_id INT PRIMARY KEY,
    serv_type VARCHAR(20),
    serv_date VARCHAR(10),
    client_id INT,
    VIN VARCHAR(17),
    FOREIGN KEY (client_id) REFERENCES Client(client_id),
    FOREIGN KEY (VIN) REFERENCES Vehicle(VIN)
);

-- Create technical_service table
CREATE TABLE IF NOT EXISTS technical_service (
    serv_id INT PRIMARY KEY,
    duration INT,
    hourly_rate DECIMAL(10,2),
    price DECIMAL(10,2),
    FOREIGN KEY (serv_id) REFERENCES Service(serv_id)
);

-- Create trading table
CREATE TABLE IF NOT EXISTS trading (
    serv_id INT PRIMARY KEY,
```

```
    type_trading VARCHAR(10),
    interest_rate DECIMAL(5,2),
    deal_price DECIMAL(10,2),
    FOREIGN KEY (serv_id) REFERENCES Service(serv_id)
);
```

The Summit Motors database system is designed primarily for supporting daily dealership operations. The expected operations are categorized by their CRUD (Create, Read, Update, Delete) nature:

- INSERT Operations (High Frequency):
    - Service Logging: Creating new service records (Service, Technical_Service) each time a customer brings a vehicle in for maintenance.
    - Client Registration: Adding new customers (Person, Client) during vehicle purchase or service intake.
    - Inventory Management: Adding new vehicles (Vehicle) to the database when shipments arrive from the manufacturer.
- UPDATE Operations (Medium Frequency):
    - Vehicle Status: Updating vehicle details (e.g., changing mileage after a test drive or updating condition from "New" to "Used").
    - Ownership Transfer: Updating the owner_id on a vehicle record when a car is sold to a client.
    - Profile Management: Updating client contact information (phone, email) or employee job titles (Employee position).
- DELETE Operations (Low Frequency):
    - Inventory Removal: Removing vehicle records that are auctioned off or scrapped.
    - Data Correction: Removing erroneous entries created by administrative mistakes.
    - Personnel Changes: Removing employee records upon termination (handled via cascading deletes from the Person table).
- SELECT / QUERY Operations (Very High Frequency):
    - Inventory Search: Customers and staff searching for vehicles by Make, Model, or VIN.
    - History Retrieval: Retrieving the complete service history for a specific VIN during a maintenance visit.
    - Authentication: Verifying user credentials (email and password) during the login process.

Based on the operational scale of a mid-sized regional dealership with multiple locations, we estimate the following data growth rates:

- Clients: Expected to handle an initial load of 2,000 client records, with a growth rate of approximately 50–100 new clients per month.

- Vehicles: The active inventory will maintain approximately 300–500 vehicles at any given time. However, the historical table of sold vehicles will grow by 40–60 records per month.
- Service Records: This will be the fastest-growing segment of the database. Assuming 15 service appointments per day across two locations, we anticipate adding ~4,500 service records annually.
- Employees & Locations: These tables are relatively static, containing fewer than 100 records combined, with infrequent updates.

Functional Dependencies and Database Normalization

Person
PK: ssn

FD: ssn → f_name, m_initial, l_name, address, DOB, gender, email, phone_no
Analysis:
1NF: All values are atomic
2NF: Non-composite PK, partial dependencies are impossible
3NF: No transitive dependencies exist

Client
PK: client_id
FD: client_id → ssn, Credit_score, owner_id
ssn (candidate key, Person specialization) → client_id, Credit_score, owner_id
Analysis:
1NF: All values are atomic
2NF: Non-composite PK, partial dependencies are impossible
3NF: Only non-prime attributes are Credit_score and FKs. No non-prime attribute determines another non-prime attribute

Employee
PK: Employee_id
FD: Employee_id → ssn, position, supervisor_id, location_no, service_id
ssn (candidate key, Person specialization) → Employee_id, position, supervisor_id, location_no, service_id
Analysis:
1NF: All valuates are atomic
2NF: Non-composite PK, partial dependencies are impossible
3NF: Only non-prime attributes are position and FKs. No non-prime attribute determines another non-prime attribute

Location
PK: loc_no
FD: loc_no → loc_name, loc address, loc_email, manager_id, owner_id
Analysis:
1NF: All values are atomic
2NF: Non-composite PK, partial dependencies are impossible
3NF: No transitive dependencies exist

Location_phone
PK: loc_no
loc_phone
FD: none
Analysis:
1NF: All values are atomic
2NF: No non-prime attributes, partial dependencies are impossible
3NF: No non-prime attributes, transitive dependencies are impossible

Vehicle
PK: VIN
FD: VIN → year, fuel, model, used_car, mileage, color, make, owner_id
Analysis:
1NF: All values are atomic
2NF: Non-composite PK, partial dependencies are impossible
3NF: No transitive dependencies exist

Owner
PK:owner_id
FD: None
Analysis:
1NF: All values are atomic
2NF: No non-prime attributes, partial dependencies are impossible
3NF: No non-prime attributes, transitive dependencies are impossible

Service
PK: service_id
FD: serv_id → serv_date, serv_type, client_id, VIN
Analysis:
1NF: All values are atomic
2NF: Non-composite PK, partial dependencies are impossible
3NF: Only non-prime attributes are FKs (VIN, client_id) which don't
depend on each other. No non-prime attribute determines another non-prime attribute

Technical_service
PK: serv_id
FD: serv_id → duration, hourly_rate, price
Analysis:
1NF: All attributes are atomic
2NF: Non-composite PK, partial dependencies are impossible
3NF: No transitive dependencies exist

Trading
PK: serv_id
FD: serv_id → type_trading, deal_price, interest_rate
Analysis:
1NF: All attributes are atomic
2NF: Non-composite PK, partial dependencies are impossible
3NF: No transitive dependencies exist

The Database System

System Installation: The database backend is built on MySQL. To install the system, the schema is initialized using a master SQL script (summit.sql). This script performs the following actions:

- Schema Creation: Initializes the summit_motors_db database.
- Table Definition: Creates the 10 core relational entities (including Person, Vehicle, Location, and Service) with Foreign Key constraints to maintain referential integrity.

- Data Seeding: Populates the tables with initial dummy data (e.g., dealership locations, demo users, and vehicle inventory) to ensure the system is ready for immediate testing.

Configuration Connectivity between the user interface and the database is managed by the db_connect.php module. This file contains the specific credentials (hostname, username, and password) required to authenticate the PHP application against the local MySQL instance.

System Invocation: The system is invoked by launching a local PHP development server.

Command: php -S localhost:8000 (on Mac Terminal)

Access: Once the server is listening, the full database system is accessible via a standard web browser at http://localhost:8000/login.html.

Operation: The application remains active as long as the terminal window running the PHP server remains open. Closing the terminal terminates the system invocation.

Screenshot Dump:

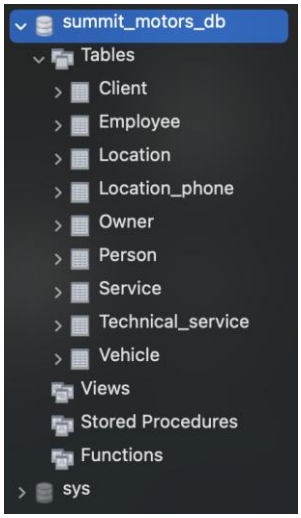


Figure 1: MySQL Schema after running summit.sql

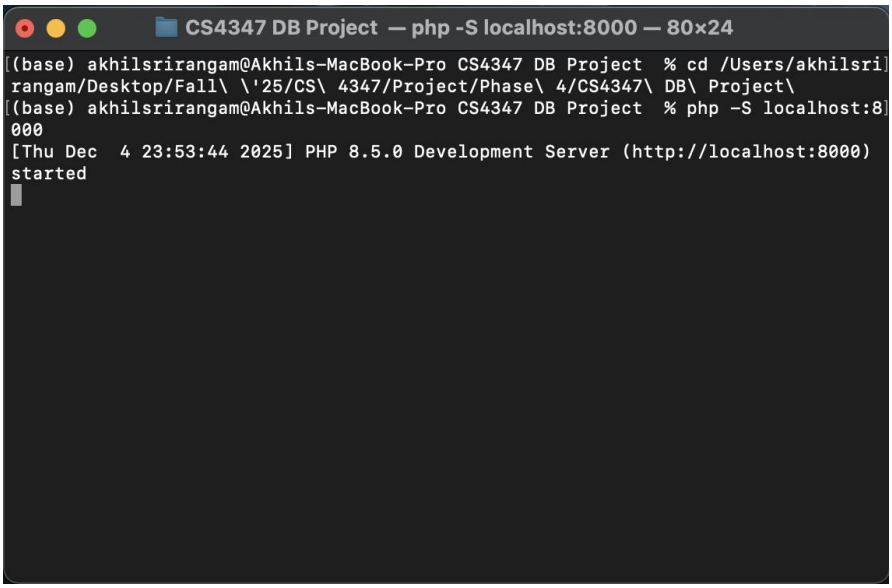


Figure 2: Initializing the local PHP development server via Terminal.

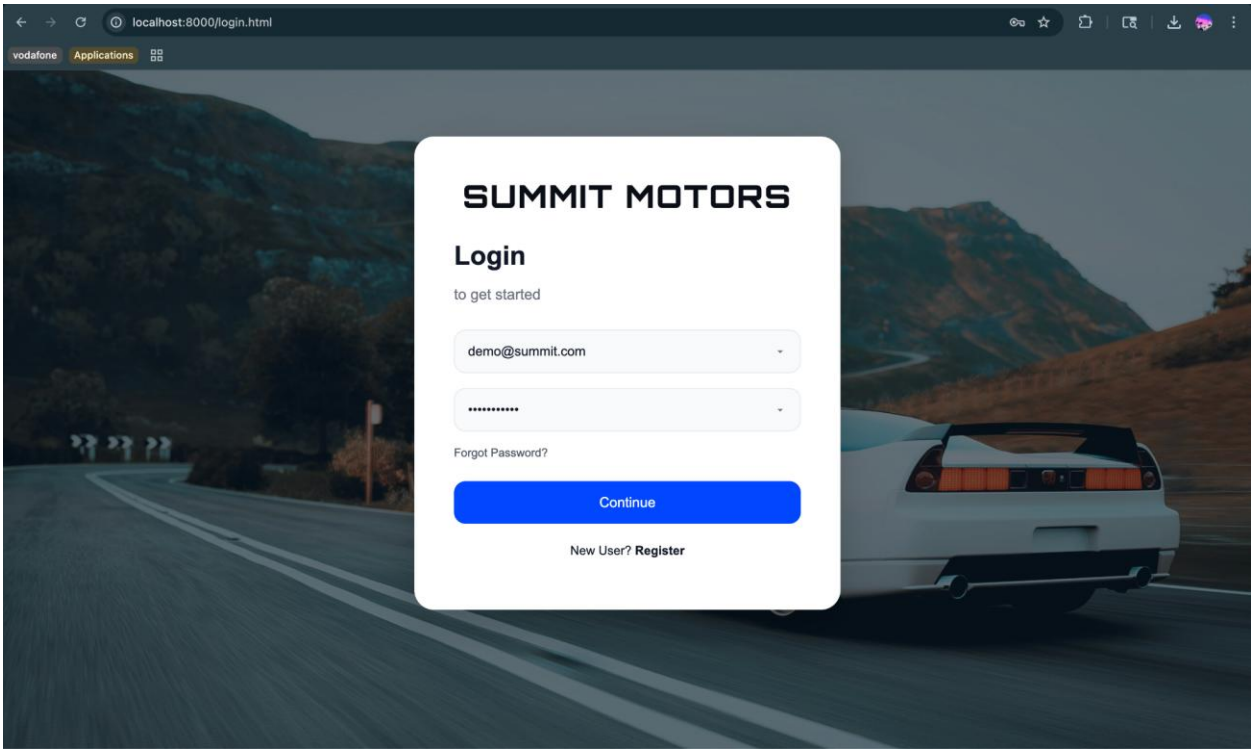


Figure 3: The secure user login interface.
Login with username: demo@summit.com, password: password123

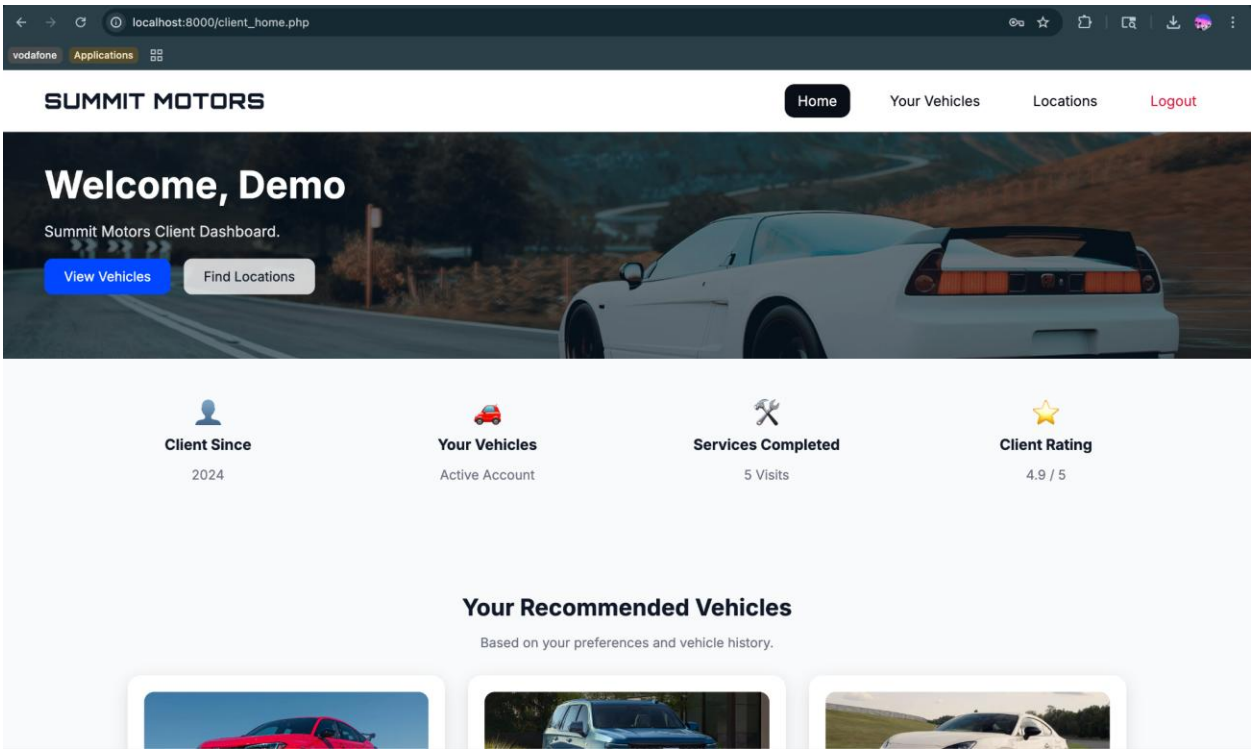


Figure 4: Client Dashboard displaying personalized user session data.

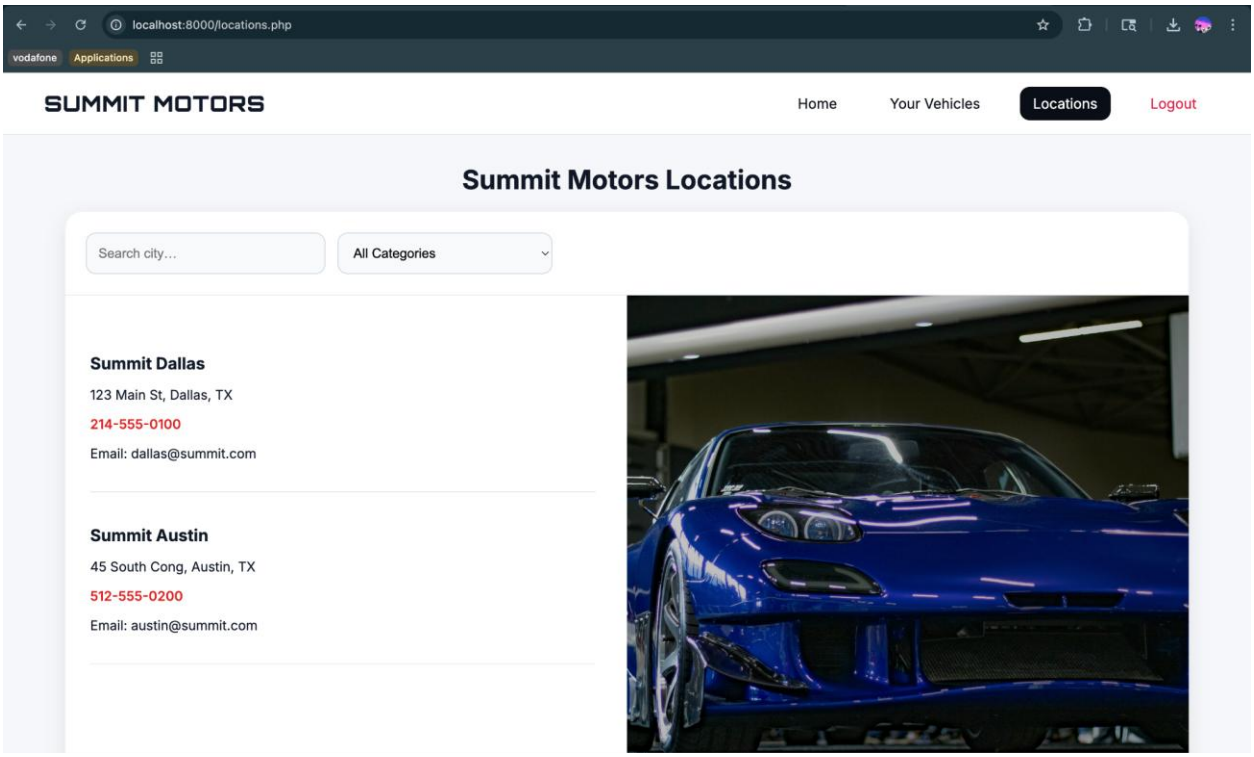


Figure 5: Locations page retrieving dealership information dynamically from the database.

CRUD Operations:

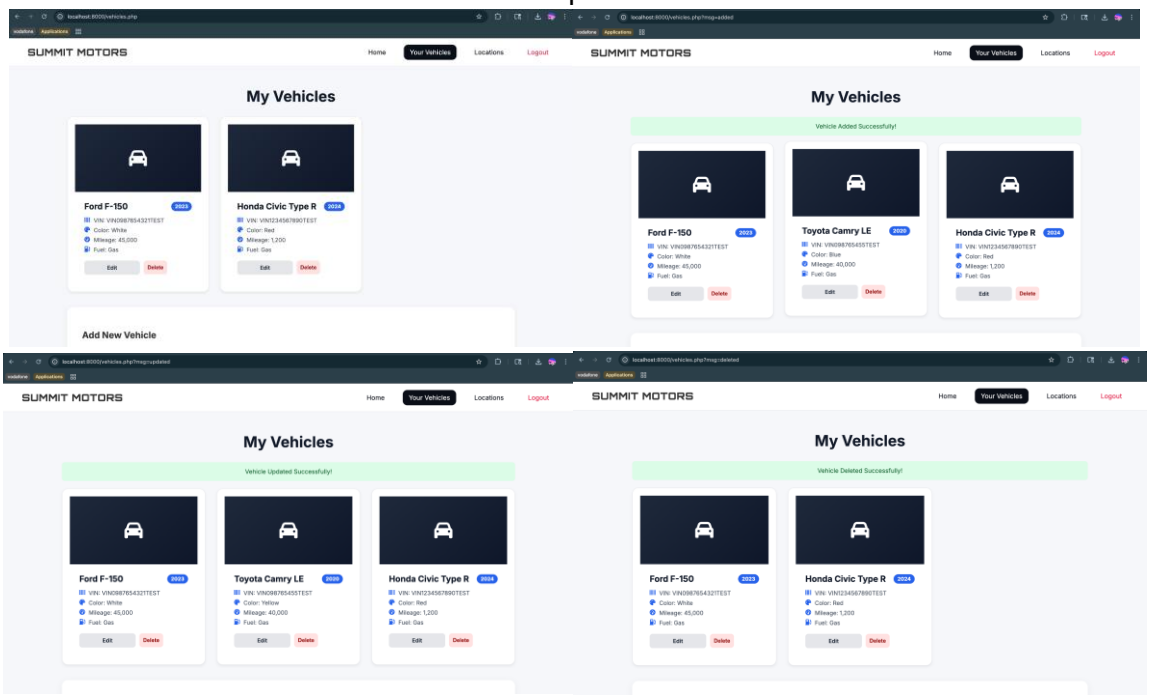


Figure 6: User's vehicle inventory retrieved from the database.

Figure 7: The "Add Vehicle" used to insert a new record into the database.

Figure 8: Interface for updating existing vehicle details (Blue → Yellow).

Figure 9: Deletion confirmation ensuring data integrity before removal.

Suggestions on Database Tuning

List of suggestions for our database:

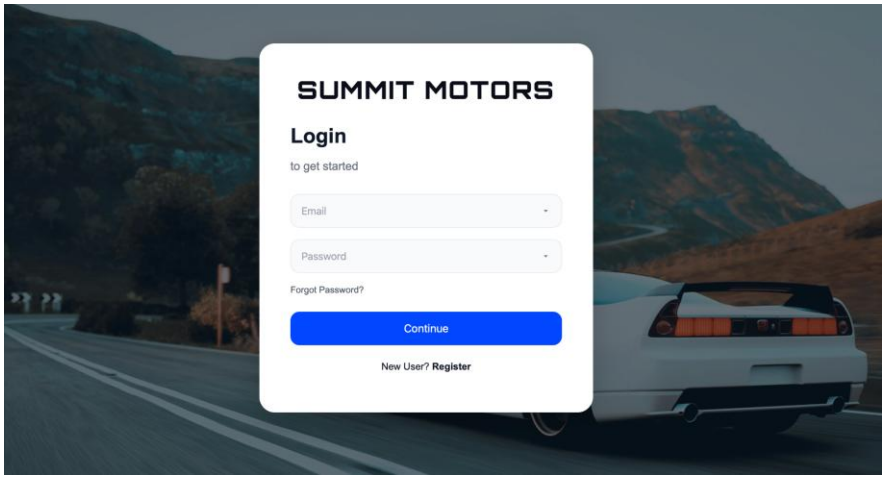
- Use appropriate data types such as use CHAR(17) for VIN (fixed length) and use YEAR for car manufacturing year
- Reduce NULL usage when possible such as Vehicle.make, Vehicle.model, and email
- Because the database stores SSNs, client financial data, vehicle ownership, and employee information, the following security enhancements are recommended:
    - Add user roles and permissions
    - Encrypt sensitive
    - Enforce strict password policies fields

User Application Interface

The Summit Motors dealership interface was built using HTML for structure of each page, CSS to provide visual layout and responsive design, and PHP backend functionality for user authentication, data retrieval, and integration with the MySQL database. Users interact with the system through dedicated pages for login, registration, vehicle and service management, and employee tools/analytics.

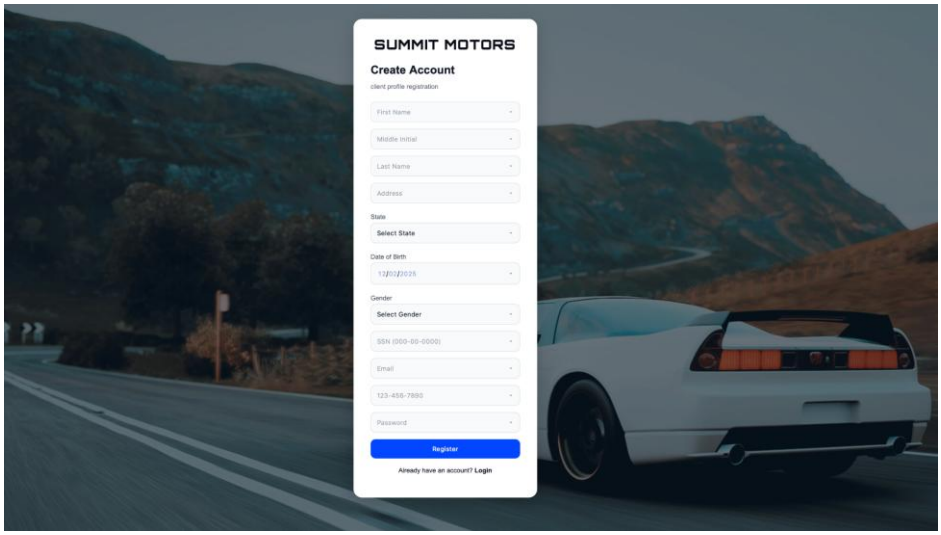
How Users Use the Summit Motors System

To access the Summit Motors site dashboard, a user must first log in through the login page, where they must enter their email address and password. The system then verifies these credentials through the backend PHP authentication process to ensure that the given user email and password are found in the Summit Motors database. If the credentials do not match or if the user is not found, they can create an account by using the register page, otherwise, the system logs them in and redirects them to the main dashboard of Summit Motors. The following figure shows the layout and composition of the login page.



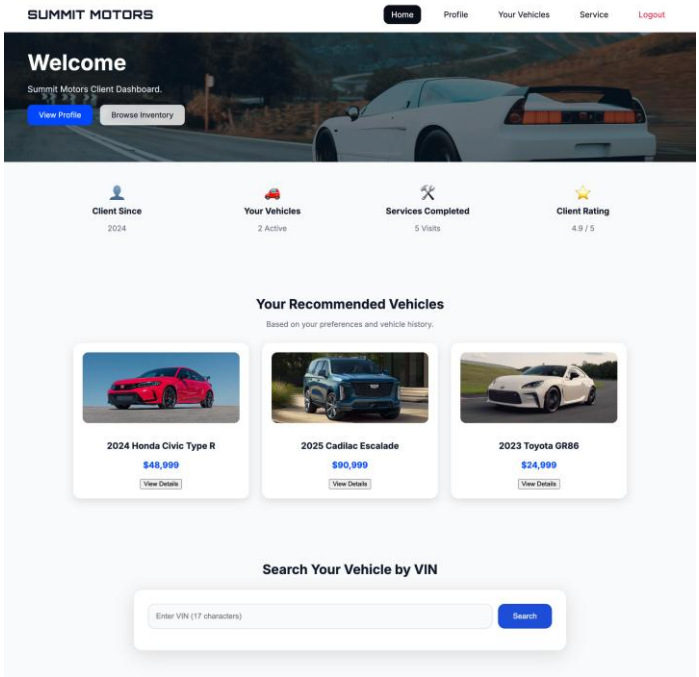
Login Page of SUMMIT MOTORS

Now if a user wants to create an account, they can access the registration page through the “Register” link located next to the “New User?” message. On this page, the user is asked to provide information such as their full name, address, state, date of birth, gender, social security number, email, phone number, and a password. Once they submit the form, the PHP program will validate all fields by checking valid formats and verifying that the social security number and email do not already exist in the database. After all inputs are verified and validated, the system will insert the new user’s information to the Summit Motors database, and then the user will be moved to the dealer dashboard.



Create Account Page of SUMMIT MOTORS

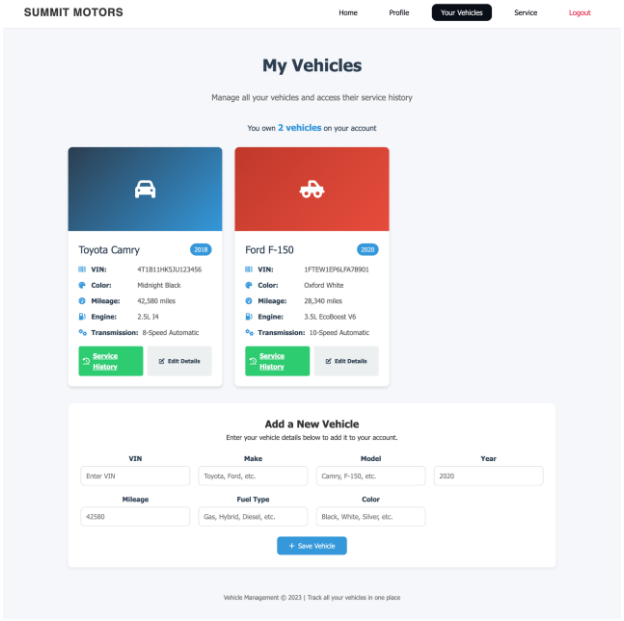
After an account is registered to the Summit Motors database, a user will now be granted to access the Summit Motors client dashboard. On this page, the user can see general account information such as the total number of services completed, the number of vehicles associated with their account, and a list of recommended vehicles for sale. This page also includes a vin search tool to allow users to find more information relating to a vehicle.



Client Dashboard of SUMMIT MOTORS

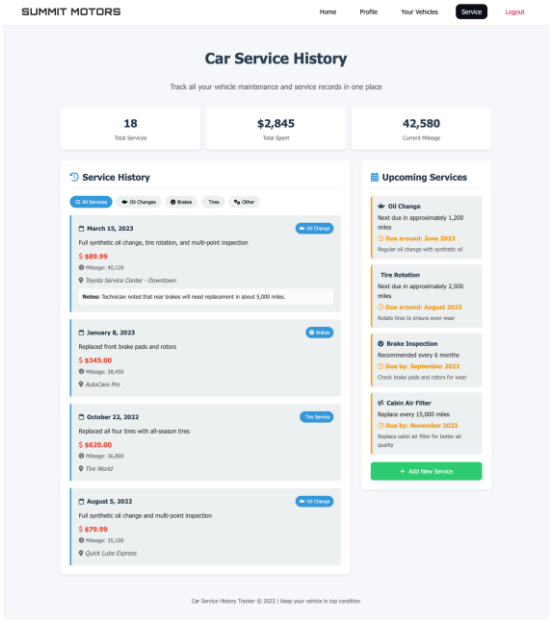
In the vehicle page, a user will be able to view all the vehicles linked to their account, showing details such as the make, model, vin number, color, mileage, engine, and transmission. Users will also be able to access the vehicle's complete service history performed at Summit Motors.

At the bottom of the page, there is the Add Vehicle section that allows users to register a new vehicle to their account by entering their car details.



Vehicle Dashboard of SUMMIT MOTORS

The service page is where a user can view and manage all service appointments that have been completed or are upcoming for their vehicles. This page provides an overview of total services completed, total amount spent, and a detailed breakdown of each service. Each service entry includes the type of work performed, date, location, price charged, and mileage at service to allow the users to track their vehicle maintenance history.



Service Dashboard of SUMMIT MOTORS

Conclusion and Future Work

This project was an introduction of the full process of familiarizing ourselves with going from business rules to a finished, integrated web application. Starting with defining a use case, target audience, and having a problem statement gave our team a frame of reference to work off of when defining business rules. Then, those business rules being translated into an EER diagram allowed us to frame our business rules in the mind of a DBA, and taking it to the next step and converting them into relations allowed us to expose any complications and solidify interaction between the tables. Finally, normalizing and implementing these diagrams and relations helped our group go from concept to practice and have fun with creating interaction between dynamic web-forms and our database. In the future, implementing some of the suggestions mentioned on “Suggestions in Database Tuning” would be our first step. Then, we would find ways of automating database updates on detection triggers to lower human error.

References

Class slides, lectures, textbooks

Appendix

https://drive.google.com/file/d/1cXsCErbdCPIFPAkN13pnxJfYZIdnG-oh/view?usp=sharing