**Title of the project:**

**Vehicle Management System**

* **Introduction:**

The Vehicle Management System is a software tool designed to help organizations manage their vehicles more easily. It keeps all vehicle-related information in one place, making it simple to organize and update records. This system is helpful for companies that rely on multiple vehicles for their daily operations, offering a more efficient way to handle vehicle data instead of using manual methods.

* **Scope**:

It stores information about each vehicle, the drivers, the trips they take, how much fuel they use, and any repairs or maintenance done. This system is useful for companies that use many vehicles for work, like in transport or delivery. It helps reduce the need for paper records and makes it easier to find and manage vehicle information.

* **Functionalities:**

1. **Vehicle Class Functionality:**

Stores vehicle information like vehicleID, model, type, and registration number.

1. **Driver Class Functionality:**

Manages driver details such as driverID, name, licenseNumber, and contactInfo.

1. **Trip Class Functionality:**

Handles trip details including tripID, vehicleID, driverID, startLocation, endLocation, date, and distance.

1. **FuelRecord Class Functionality:**

Keeps records of fuel usage using attributes like fuelID, vehicleID, date, fuelType, quantity, and cost.

1. **MaintenanceRecord Class Functionality:**

Manages maintenance data such as maintenanceID, vehicleID, date, description, and cost.

1. **Compliance class Functionality:**

Allows adding, viewing, updating, deleting compliance records and checking for expiry**.**

1. **Reservation class Functionality:**

Enables vehicle booking, viewing, updating, canceling reservations, and checking availability.

* **System Requirements:**
* A relational database management system (e.g., MySQL, PostgreSQL)
* SQL-compatible client or interface (like MySQL Workbench, phpMyAdmin)
* For Front end html and java Compiler(like Vscode)
* **Challenges Faced:**

1. **Database Design:**  
    Creating a schema with all needed tables and relationships, like standalone Inventory, was tough.
2. **Data Consistency:**  
    Ensuring sample data (e.g., VehicleID across tables) was correct and consistent was hard.
3. **Front-End Integration:**  
    Linking HTML forms to the database for adding/editing records was challenging.
4. **Reservation Conflicts:**  
    Preventing overlapping reservations for vehicles without constraints was difficult.

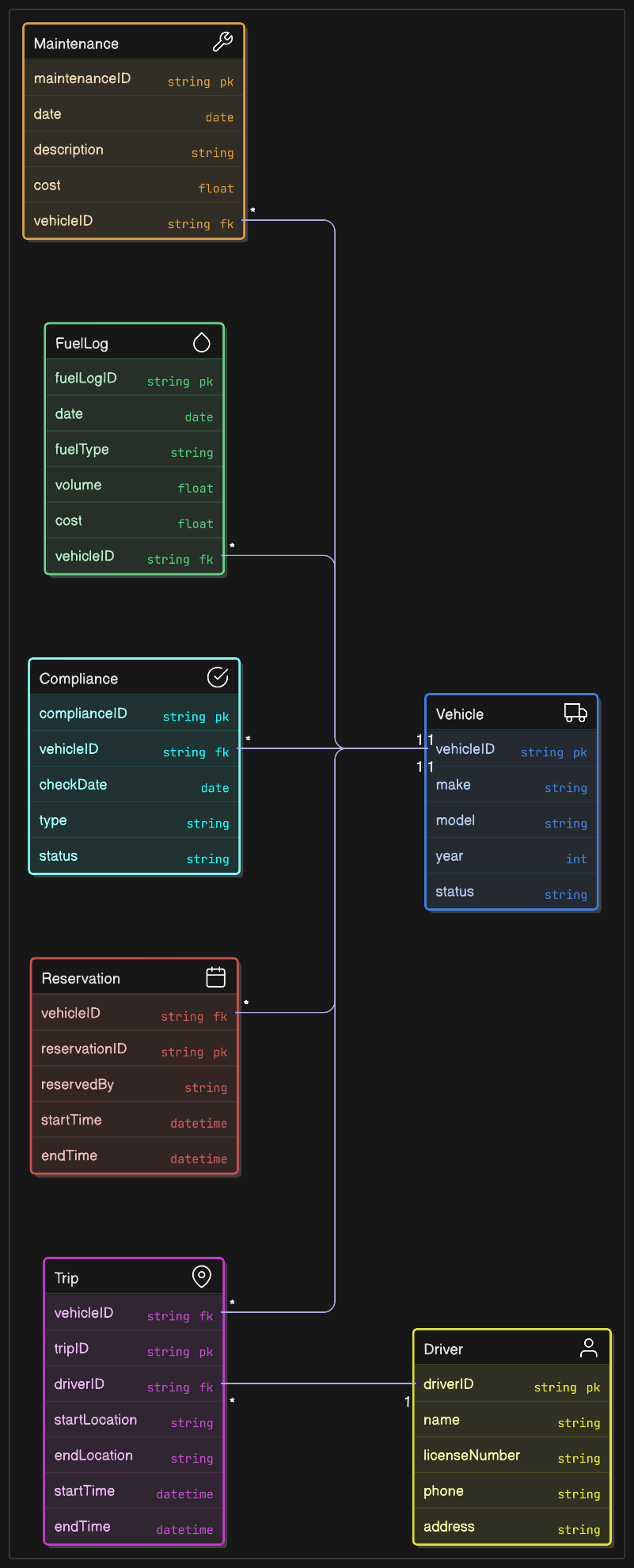
* **Lesson Learned:**

The vehicle management system taught us to plan table relationships, like linking Inventory to Maintenance, test data early, add indexes for faster queries, validate HTML form inputs, prevent reservation overlaps, use clear APIs, and keep ER diagrams for clarity.

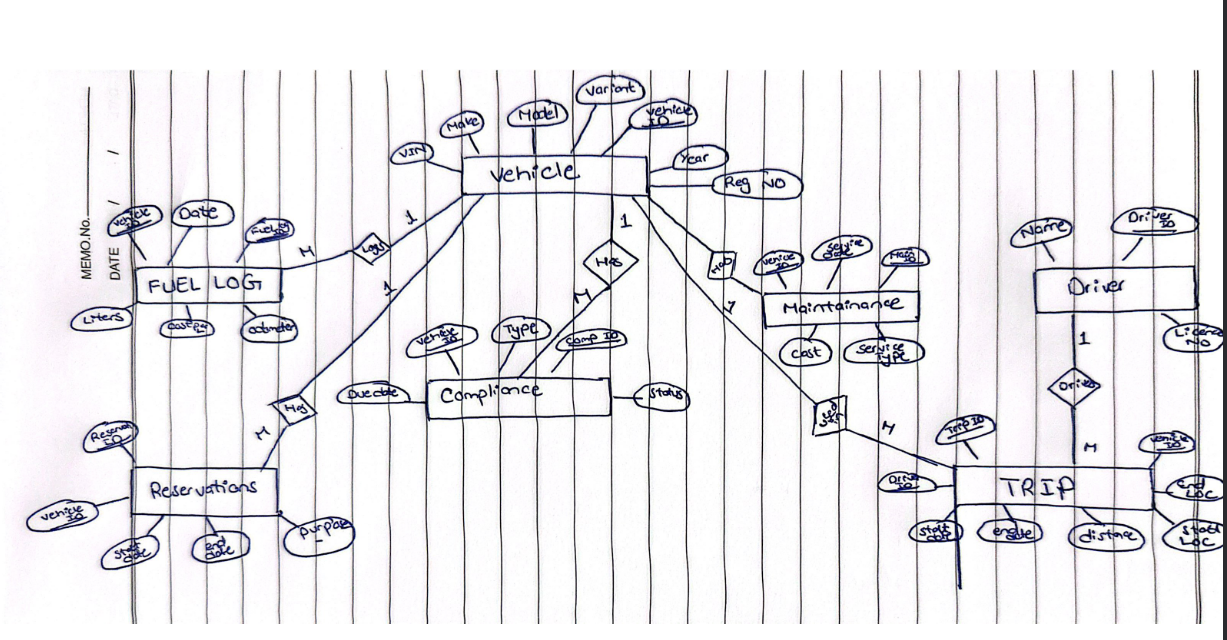
* **Conclusion:**

The vehicle management system project was a success, building a great database and easy-to-use interface for managing vehicles, maintenance, fuel, trips, inventory, compliance, and reservations. We fixed challenges like missing table links and slow queries by planning relationships, testing data, adding indexes, checking forms, avoiding reservation conflicts, using APIs, and keeping clear diagrams, creating a reliable system and learning for future projects.

* **Relational Schema:**



* **ER DIAGRAM:**

****

* **SQL Code:**

-- Creating the database

CREATE DATABASE vehicle\_management;

USE vehicle\_management;

-- Creating tables (DDL)

CREATE TABLE Vehicle (

VehicleID INT PRIMARY KEY AUTO\_INCREMENT,

Make VARCHAR(50) NOT NULL,

Model VARCHAR(50) NOT NULL,

Variant VARCHAR(50),

Year INT NOT NULL,

VIN VARCHAR(17) UNIQUE NOT NULL,

RegistrationNo VARCHAR(20) UNIQUE NOT NULL

);

CREATE TABLE Driver (

DriverID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL,

LicenseNo VARCHAR(20) UNIQUE NOT NULL

);

CREATE TABLE Maintenance (

MaintenanceID INT PRIMARY KEY AUTO\_INCREMENT,

VehicleID INT,

ServiceDate DATE NOT NULL,

Cost DECIMAL(10,2) NOT NULL,

ServiceType VARCHAR(50) NOT NULL,

FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID)

);

CREATE TABLE FuelLog (

FuelLogID INT PRIMARY KEY AUTO\_INCREMENT,

VehicleID INT,

Date DATE NOT NULL,

Liters DECIMAL(10,2) NOT NULL,

CostPerLiter DECIMAL(10,2) NOT NULL,

Odometer INT NOT NULL,

FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID)

);

CREATE TABLE Trip (

TripID INT PRIMARY KEY AUTO\_INCREMENT,

VehicleID INT,

DriverID INT,

StartDate DATETIME NOT NULL,

EndDate DATETIME NOT NULL,

Distance DECIMAL(10,2) NOT NULL,

StartLocation VARCHAR(100),

EndLocation VARCHAR(100),

FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID),

FOREIGN KEY (DriverID) REFERENCES Driver(DriverID)

);

CREATE TABLE Inventory (

PartID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL,

Quantity INT NOT NULL,

ReorderLevel INT NOT NULL,

Supplier VARCHAR(100)

);

CREATE TABLE Compliance (

ComplianceID INT PRIMARY KEY AUTO\_INCREMENT,

VehicleID INT,

Type VARCHAR(50) NOT NULL,

DueDate DATE NOT NULL,

Status VARCHAR(20) NOT NULL,

FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID)

);

CREATE TABLE Reservation (

ReservationID INT PRIMARY KEY AUTO\_INCREMENT,

VehicleID INT,

StartDate DATETIME NOT NULL,

EndDate DATETIME NOT NULL,

Purpose VARCHAR(100),

FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID)

);

-- Populating tables with sample data (10+ records per table)

INSERT INTO Vehicle (Make, Model, Variant, Year, VIN, RegistrationNo) VALUES

('Toyota', 'Camry', 'LE', 2020, '4T1BF1FK0LU123456', 'ABC123'),

('Honda', 'Civic', 'EX', 2021, '2HGFC2F75MH543210', 'XYZ789'),

('Ford', 'F-150', 'XLT', 2019, '1FTEW1E43KFA98765', 'LMN456'),

('Chevrolet', 'Malibu', 'LT', 2022, '1G1ZD5STXNF123456', 'DEF789'),

('Nissan', 'Altima', 'SV', 2020, '1N4BL4DVXLC654321', 'GHI123'),

('Hyundai', 'Tucson', 'SEL', 2021, '5NMJB3AE0MH678901', 'JKL456'),

('BMW', 'X3', 'sDrive30i', 2023, '5UXWX9C58P0A12345', 'MNO789'),

('Mercedes', 'C-Class', 'C300', 2020, 'W1KWF8DBXLR987654', 'PQR123'),

('Tesla', 'Model 3', 'Standard', 2022, '5YJ3E1EA7NF456789', 'STU456'),

('Volkswagen', 'Tiguan', 'SE', 2021, '3VV3B7AX8MM123456', 'VWX789'),

('Kia', 'Sportage', 'LX', 2020, '5XYK33AF0LG987654', 'YZA123'),

('Subaru', 'Outback', 'Limited', 2021, '4S4BTANC4M3123456', 'BCD456');

INSERT INTO Driver (Name, LicenseNo) VALUES

('John Doe', 'D1234567'),

('Jane Smith', 'D7654321'),

('Mike Johnson', 'D9876543'),

('Sarah Williams', 'D4567890'),

('David Brown', 'D3210987'),

('Emily Davis', 'D6543210'),

('Michael Wilson', 'D7890123'),

('Laura Taylor', 'D2109876'),

('James Anderson', 'D5432109'),

('Lisa Martinez', 'D8765432'),

('Robert Clark', 'D1098765'),

('Anna Lewis', 'D4321098');

INSERT INTO Maintenance (VehicleID, ServiceDate, Cost, ServiceType) VALUES

(1, '2025-01-10', 75.00, 'Oil Change'),

(1, '2025-03-15', 150.00, 'Tire Rotation'),

(2, '2025-02-20', 200.00, 'Brake Repair'),

(2, '2025-04-10', 80.00, 'Oil Change'),

(3, '2025-01-05', 300.00, 'Transmission Service'),

(4, '2025-03-01', 90.00, 'Oil Change'),

(5, '2025-02-15', 120.00, 'Battery Replacement'),

(6, '2025-04-05', 250.00, 'Brake Repair'),

(7, '2025-01-20', 100.00, 'Tire Rotation'),

(8, '2025-03-10', 85.00, 'Oil Change'),

(9, '2025-02-25', 400.00, 'AC Repair'),

(10, '2025-04-01', 110.00, 'Tire Rotation');

INSERT INTO FuelLog (VehicleID, Date, Liters, CostPerLiter, Odometer) VALUES

(1, '2025-01-05', 40.0, 1.50, 50000),

(1, '2025-02-10', 45.0, 1.55, 51000),

(2, '2025-01-15', 35.0, 1.60, 45000),

(2, '2025-03-01', 38.0, 1.58, 46000),

(3, '2025-02-05', 50.0, 1.65, 60000),

(4, '2025-01-20', 42.0, 1.50, 30000),

(5, '2025-03-10', 37.0, 1.55, 42000),

(6, '2025-02-15', 39.0, 1.60, 35000),

(7, '2025-01-25', 41.0, 1.58, 25000),

(8, '2025-03-05', 43.0, 1.62, 28000),

(9, '2025-02-20', 36.0, 1.59, 20000),

(10, '2025-04-01', 40.0, 1.61, 32000);

INSERT INTO Trip (VehicleID, DriverID, StartDate, EndDate, Distance, StartLocation, EndLocation) VALUES

(1, 1, '2025-01-10 08:00:00', '2025-01-10 12:00:00', 100.5, 'New York', 'Boston'),

(1, 2, '2025-02-15 09:00:00', '2025-02-15 15:00:00', 120.0, 'Boston', 'Philadelphia'),

(2, 3, '2025-01-20 07:00:00', '2025-01-20 11:00:00', 80.0, 'Chicago', 'Detroit'),

(2, 4, '2025-03-01 10:00:00', '2025-03-01 14:00:00', 90.5, 'Detroit', 'Cleveland'),

(3, 5, '2025-02-10 08:30:00', '2025-02-10 13:30:00', 150.0, 'Los Angeles', 'San Diego'),

(4, 6, '2025-01-15 09:00:00', '2025-01-15 12:00:00', 70.0, 'Miami', 'Orlando'),

(5, 7, '2025-03-05 07:00:00', '2025-03-05 11:00:00', 85.0, 'Seattle', 'Portland'),

(6, 8, '2025-02-20 08:00:00', '2025-02-20 14:00:00', 110.0, 'Dallas', 'Houston'),

(7, 9, '2025-01-25 09:30:00', '2025-01-25 13:30:00', 95.0, 'Atlanta', 'Charlotte'),

(8, 10, '2025-03-10 08:00:00', '2025-03-10 12:00:00', 100.0, 'Phoenix', 'Tucson'),

(9, 11, '2025-02-25 07:30:00', '2025-02-25 11:30:00', 88.0, 'Denver', 'Boulder'),

(10, 12, '2025-04-01 09:00:00', '2025-04-01 13:00:00', 105.0, 'San Francisco', 'San Jose');

INSERT INTO Inventory (Name, Quantity, ReorderLevel, Supplier) VALUES

('Oil Filter', 50, 10, 'AutoParts Inc.'),

('Brake Pads', 30, 5, 'BrakeTech'),

('Tires', 20, 8, 'TireWorld'),

('Air Filter', 40, 15, 'AutoParts Inc.'),

('Spark Plugs', 60, 20, 'EngineTech'),

('Battery', 15, 5, 'PowerCorp'),

('Wiper Blades', 25, 10, 'ClearView'),

('Headlight Bulb', 35, 12, 'LightTech'),

('Brake Fluid', 45, 10, 'BrakeTech'),

('Coolant', 50, 15, 'CoolSys'),

('Transmission Fluid', 30, 8, 'AutoParts Inc.'),

('Fuel Filter', 25, 10, 'EngineTech');

INSERT INTO Compliance (VehicleID, Type, DueDate, Status) VALUES

(1, 'Insurance', '2025-12-01', 'Active'),

(1, 'Inspection', '2025-06-15', 'Pending'),

(2, 'Insurance', '2025-11-30', 'Active'),

(2, 'Tax', '2025-04-30', 'Paid'),

(3, 'Insurance', '2025-10-01', 'Active'),

(4, 'Inspection', '2025-07-20', 'Pending'),

(5, 'Insurance', '2025-09-15', 'Active'),

(6, 'Tax', '2025-05-31', 'Paid'),

(7, 'Insurance', '2025-08-01', 'Active'),

(8, 'Inspection', '2025-06-30', 'Pending'),

(9, 'Tax', '2025-04-15', 'Paid'),

(10, 'Insurance', '2025-11-01', 'Active');

INSERT INTO Reservation (VehicleID, StartDate, EndDate, Purpose) VALUES

(1, '2025-05-01 08:00:00', '2025-05-01 17:00:00', 'Business Trip'),

(2, '2025-05-02 09:00:00', '2025-05-02 18:00:00', 'Client Meeting'),

(3, '2025-05-03 07:00:00', '2025-05-03 15:00:00', 'Delivery'),

(4, '2025-05-04 08:00:00', '2025-05-04 16:00:00', 'Field Work'),

(5, '2025-05-05 09:00:00', '2025-05-05 17:00:00', 'Staff Transport'),

(6, '2025-05-06 08:00:00', '2025-05-06 18:00:00', 'Business Trip'),

(7, '2025-05-07 07:00:00', '2025-05-07 15:00:00', 'Client Meeting'),

(8, '2025-05-08 09:00:00', '2025-05-08 17:00:00', 'Delivery'),

(9, '2025-05-09 08:00:00', '2025-05-09 16:00:00', 'Field Work'),

(10, '2025-05-10 09:00:00', '2025-05-10 17:00:00', 'Staff Transport'),

(1, '2025-05-11 08:00:00', '2025-05-11 17:00:00', 'Business Trip'),

(2, '2025-05-12 09:00:00', '2025-05-12 18:00:00', 'Client Meeting');

* **DML Queries:**

-- INSERT: Add a new vehicle

INSERT INTO Vehicle (Make, Model, Variant, Year, VIN, RegistrationNo)

VALUES ('Toyota', 'Corolla', 'SE', 2023, '4T1BF1FK3NU654321', 'EFG456');

-- UPDATE: Update maintenance cost

UPDATE Maintenance SET Cost = 100.00 WHERE MaintenanceID = 1;

-- DELETE: Delete a fuel log entry

DELETE FROM FuelLog WHERE FuelLogID = 1;

-- SELECT: Get all vehicles manufactured after 2020

SELECT \* FROM Vehicle WHERE Year > 2020;

-- Joins

-- INNER JOIN: Get vehicle and maintenance details

SELECT v.Make, v.Model, m.ServiceDate, m.ServiceType, m.Cost

FROM Vehicle v

INNER JOIN Maintenance m ON v.VehicleID = m.VehicleID;

-- LEFT JOIN: Get all vehicles and their trips (if any)

SELECT v.Make, v.Model, t.TripID, t.Distance

FROM Vehicle v

LEFT JOIN Trip t ON v.VehicleID = t.VehicleID;

-- RIGHT JOIN: Get all drivers and their assigned trips (if any)

SELECT d.Name, t.TripID, t.StartDate

FROM Driver d

RIGHT JOIN Trip t ON d.DriverID = t.DriverID;

-- Aggregate Functions

-- COUNT: Number of maintenance records per vehicle

SELECT v.Make, v.Model, COUNT(m.MaintenanceID) AS ServiceCount

FROM Vehicle v

LEFT JOIN Maintenance m ON v.VehicleID = m.VehicleID

GROUP BY v.VehicleID, v.Make, v.Model;

-- SUM and AVG: Total and average fuel cost per vehicle

SELECT v.Make, v.Model, SUM(f.Liters \* f.CostPerLiter) AS TotalFuelCost, AVG(f.Liters \* f.CostPerLiter) AS AvgFuelCost

FROM Vehicle v

LEFT JOIN FuelLog f ON v.VehicleID = f.VehicleID

GROUP BY v.VehicleID, v.Make, v.Model;

-- Subquery: Vehicles with maintenance costs above average

SELECT v.Make, v.Model

FROM Vehicle v

WHERE v.VehicleID IN (

SELECT VehicleID

FROM Maintenance

GROUP BY VehicleID

HAVING AVG(Cost) > (SELECT AVG(Cost) FROM Maintenance)

);

-- View: Fuel efficiency per vehicle

CREATE VIEW FuelEfficiency AS

SELECT v.VehicleID, v.Make, v.Model,

t.Distance / f.Liters AS KmPerLiter

FROM Vehicle v

INNER JOIN FuelLog f ON v.VehicleID = f.VehicleID

INNER JOIN Trip t ON v.VehicleID = t.VehicleID

WHERE t.Distance > 0 AND f.Liters > 0;

-- View: Low inventory alert

CREATE VIEW LowInventory AS

SELECT Name, Quantity, ReorderLevel

FROM Inventory

WHERE Quantity <= ReorderLevel;

* **FRONT END:**