VEHICLE SERVICE MANAGEMENT SYSTEM

Mayuresh-PES1UG22CS340

Mohammed Jazaa-PES1UG22CS353

1. Purpose of the project:

The Vehicle Service Management System (VSMS) is designed to streamline and optimize the management of vehicle service operations. It serves as a comprehensive platform for service centers to handle tasks such as tracking service records, managing customer payments, and monitoring revenue.

Key Objectives:

1. **Service History Management**: To maintain a detailed history of vehicle services, enabling quick access to service dates, descriptions, costs, and technician information.
2. **Revenue Tracking**: To calculate and display the total revenue generated by the service center, providing financial insights and aiding in strategic planning.
3. **Database Integration**: To store and manage data securely using a database, ensuring reliable and efficient retrieval of information.
4. **User Accessibility**: To provide a user-friendly interface for administrators and service staff to interact with the system seamlessly.

Impact: The system minimizes manual errors, enhances operational efficiency, and improves customer satisfaction by ensuring timely and accurate service management. It is particularly valuable for service centers aiming to digitize their workflows and offer a modern service experience.

2. Scope of the project:

The Vehicle Service Management System (VSMS) aims to modernize and streamline the operations of vehicle service centers. Its scope encompasses a wide range of functionalities and applications, designed to enhance the efficiency, accuracy, and overall customer experience.

Scope Highlights

1. **Core Functionalities**:
    - **Service Record Management**: Store and retrieve comprehensive service records, including vehicle details, service descriptions, costs, and technician assignments.

    - **Revenue Analysis**: Track and calculate revenue generated from services and payments, providing financial insights to the service center.
    - **Customer Interaction**: Manage customer data, vehicle information, and service schedules to improve engagement and satisfaction.
2. **Administrative Capabilities**:
    - Facilitate data entry and management for vehicle service details.
    - Provide an intuitive interface for administrators to monitor service history and revenue.
3. **Technician Coordination**:
    - Assign services

3. Detailed description of the project:

The **Vehicle Service Management System (VSMS)** is a robust and efficient platform designed to digitize and streamline the operations of vehicle service centers. By integrating service management, financial tracking, and customer interaction functionalities, the system aims to enhance operational efficiency, reduce errors, and provide a better customer experience.

System Overview

The VSMS is a web-based application built with a combination of technologies like PHP, MySQL, and a user-friendly frontend interface. It serves as a central hub for managing vehicle services, customer interactions, and administrative tasks.

Key Modules and Features

1. Database Management

- **Centralized Data Storage**: The system uses a MySQL database to securely store and manage all data, including customer details, vehicle information, service records, and payment history.
- **Data Integrity**: Ensures consistent and accurate data storage and retrieval, minimizing manual errors.

2. Service Management

- **Service Records**: Allows the storage of service details such as service date, description, cost, and the assigned technician.
- **Service History Retrieval**: Provides detailed service history for a particular vehicle, enabling quick decision-making and improved customer service.
- **Service Scheduling**: Facilitates the scheduling of services, ensuring optimized time management and resource allocation.

3. Technician Management

- **Technician Assignment**: Assign specific technicians to services based on expertise and availability.
- **Performance Monitoring**: Track services completed by each technician, allowing for performance analysis.

4. Revenue Tracking

- **Real-Time Revenue Monitoring**: Calculate and display the total revenue generated from service payments.
- **Financial Reporting**: Generate reports to help the management evaluate business performance and profitability.

5. User Roles and Accessibility

- **Admin Panel**: A dedicated panel for administrators to monitor all system activities, manage records, and generate reports.
- **Technician Access**: Limited access for technicians to view and update their assigned services.

6. Customer Interaction

- **Customer Database**: Maintain detailed records of customers and their vehicles for personalized service.
- **Service Notifications**: Notify customers about service updates, payments, or upcoming maintenance schedules.

4. Functional Requirements of the project:

The functional requirements define the core functionalities and operations that the Vehicle Service Management System must support to achieve its objectives. These are categorized into modules for better clarity.

1. Service Management

- **Add Service Records**: The system should allow administrators to add detailed service records, including:
    - Service date.
    - Description of the service.
    - Service cost.
    - Technician assigned.
- **Retrieve Service History**: Provide a feature to retrieve and display the complete service history of a specific vehicle, sorted by date.
- **Update Service Records**: Allow modifications to service records when needed, such as updating costs or descriptions.

- **Delete Service Records**: Enable the removal of outdated or incorrect service records.

2. Vehicle Management

- **Add Vehicle Details**: Capture essential details of vehicles, including:
    - Vehicle ID.
    - Make and model.
    - Year of manufacture.
    - Owner information.
- **Search Vehicle Records**: Support search functionality to find specific vehicles using IDs or other criteria.
- **Update Vehicle Information**: Allow administrators to modify vehicle details as required.
- **Delete Vehicle Records**: Provide the ability to remove vehicles no longer associated with the service center.

3. Technician Management

- **Assign Technicians to Services**: Facilitate the assignment of technicians based on expertise or availability.
- **Technician Database**: Maintain a record of technicians, including:
    - Technician ID.
    - Name.
    - Contact information.
    - Area of expertise.
- **Monitor Technician Performance**: Generate reports on the number of services completed by each technician.

4. Customer Management

- **Customer Database**: Store customer information such as:
    - Name.
    - Contact details.
    - Associated vehicles.

Service Feedback: Allow customers to provide feedback on services rendered.

5. Payment and Revenue Management

- **Add Payment Details**: Record payment information for services, including:
    - Payment date.
    - Amount paid.

    - Payment method (cash, card, online, etc.).
- **Revenue Tracking**: Summarize the total revenue generated over a specific period.
- **Generate Payment Receipts**: Provide printable or digital payment receipts for customers.

6. Reporting and Analytics

- **Generate Service Reports**: Create detailed reports on services rendered, including:
    - Number of services per vehicle.
    - Services categorized by type (maintenance, repair, etc.).
- **Revenue Reports**: Generate periodic financial reports to monitor business performance.
- **Technician Activity Reports**: Provide insights into the workload and performance of each technician.

7. User Roles and Access Control

- **Administrator Access**: Full access to manage all system functionalities.
- **Technician Access**: Limited access to view assigned services and update service status.
- **Customer Access (Optional)**: Allow customers to view service history for their vehicles and provide feedback.

8. Security and Data Integrity

- **Authentication**: Implement login systems for administrators and technicians to ensure secure access.
- **Data Validation**: Validate inputs for consistency and accuracy, such as ensuring valid vehicle IDs and service costs.
- **Backup and Recovery**: Provide options to back up the database and recover data in case of system failures.

9. System Scalability

- **Multi-Branch Support**: Allow the system to be extended for managing multiple service center locations.
- **Integration Options**: Support future integration with external systems, such as IoT-enabled diagnostics or CRM tools.

These functional requirements ensure the VSMS meets the operational needs of vehicle service centers while providing a scalable and efficient solution.

5. Deliverables for the Vehicle Service Management System (VSMS)

1. Title of the Problem Statement with Team Details

Title: Vehicle Service Management System (VSMS)
Team Details:

- **Member 1**: Database Design, Documentation, and Testing
- **Member 2**: Frontend Development, Backend Development

2. Abstract

The **Vehicle Service Management System (VSMS)** is a comprehensive application designed to digitize and enhance the operational efficiency of vehicle service centers. The system focuses on automating service tracking, customer management, and revenue monitoring while providing a user-friendly platform for administrators and technicians.

With dynamic database integration, the platform supports vehicle and service record management, technician assignment, payment processing, and detailed reporting. By integrating advanced features such as service history retrieval, revenue analysis, and payment gateway support, the VSMS ensures reliable, transparent, and efficient operations. The system also enhances customer satisfaction by providing timely notifications, accurate billing, and personalized service recommendations.

3. User Requirement Specification

The **Vehicle Service Management System** aims to streamline vehicle service operations for service centers. It offers a secure and efficient way to manage service records, customers, and financial data.

- **Administrator Features**:
    - Add, view, update, and delete vehicle and service records.
    - Assign technicians to specific tasks based on their expertise.
    - Monitor service history and revenue reports.
    - Manage customer information and send service notifications.
- **Technician Features**:
    - Access assigned tasks with details such as vehicle, service description, and deadlines.
    - Update service status and provide feedback on task completion.
- **Customer Features** (optional):
    - View vehicle service history.
    - Receive notifications about service schedules, payments, and completion.
- **Payment Processing**:
    - Track payment history and issue receipts for completed transactions.

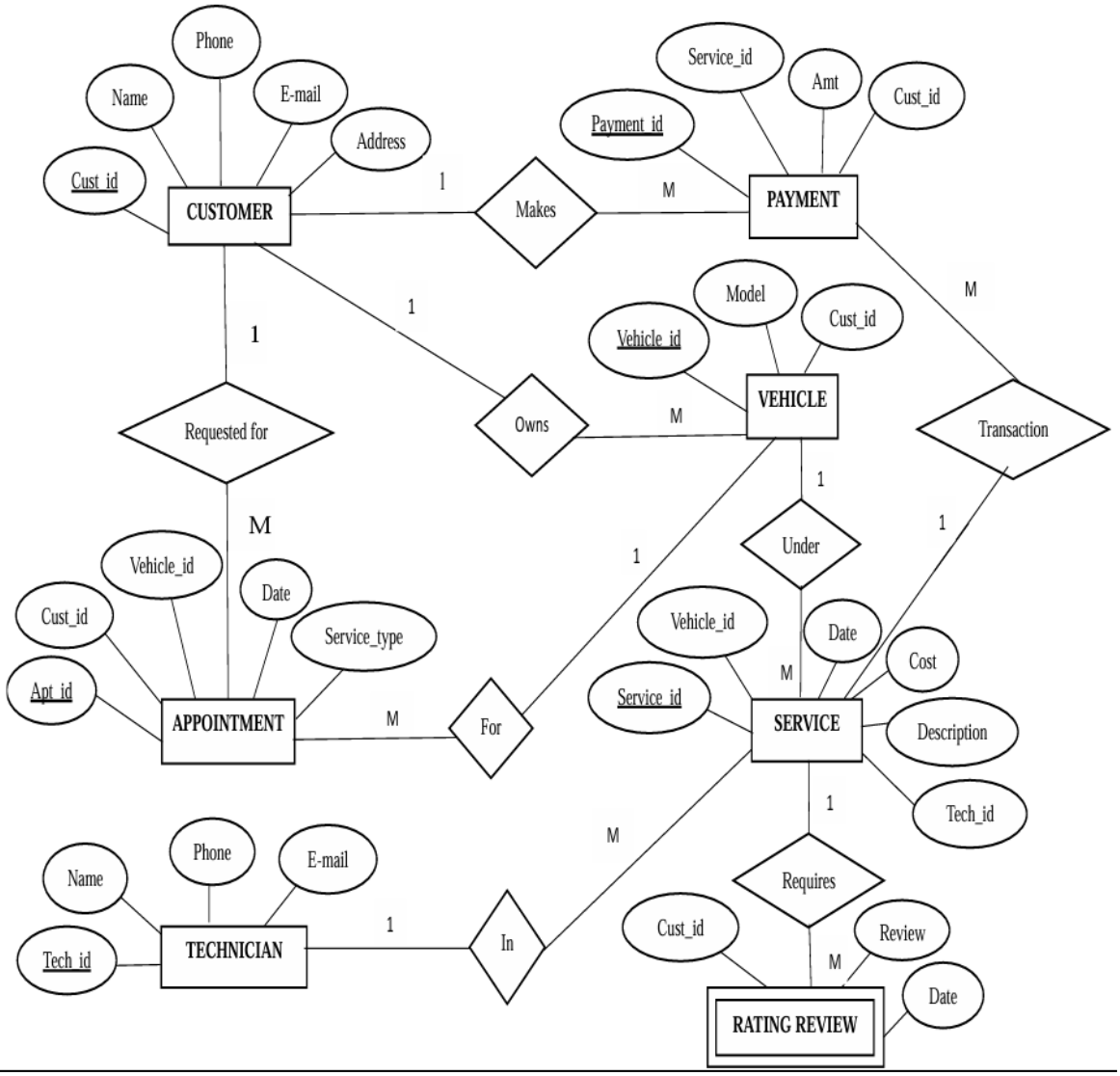
- **Data Security**:
    - Provide role-based access to protect sensitive information.
    - Maintain a secure database for storing vehicle, customer, and financial records.

---

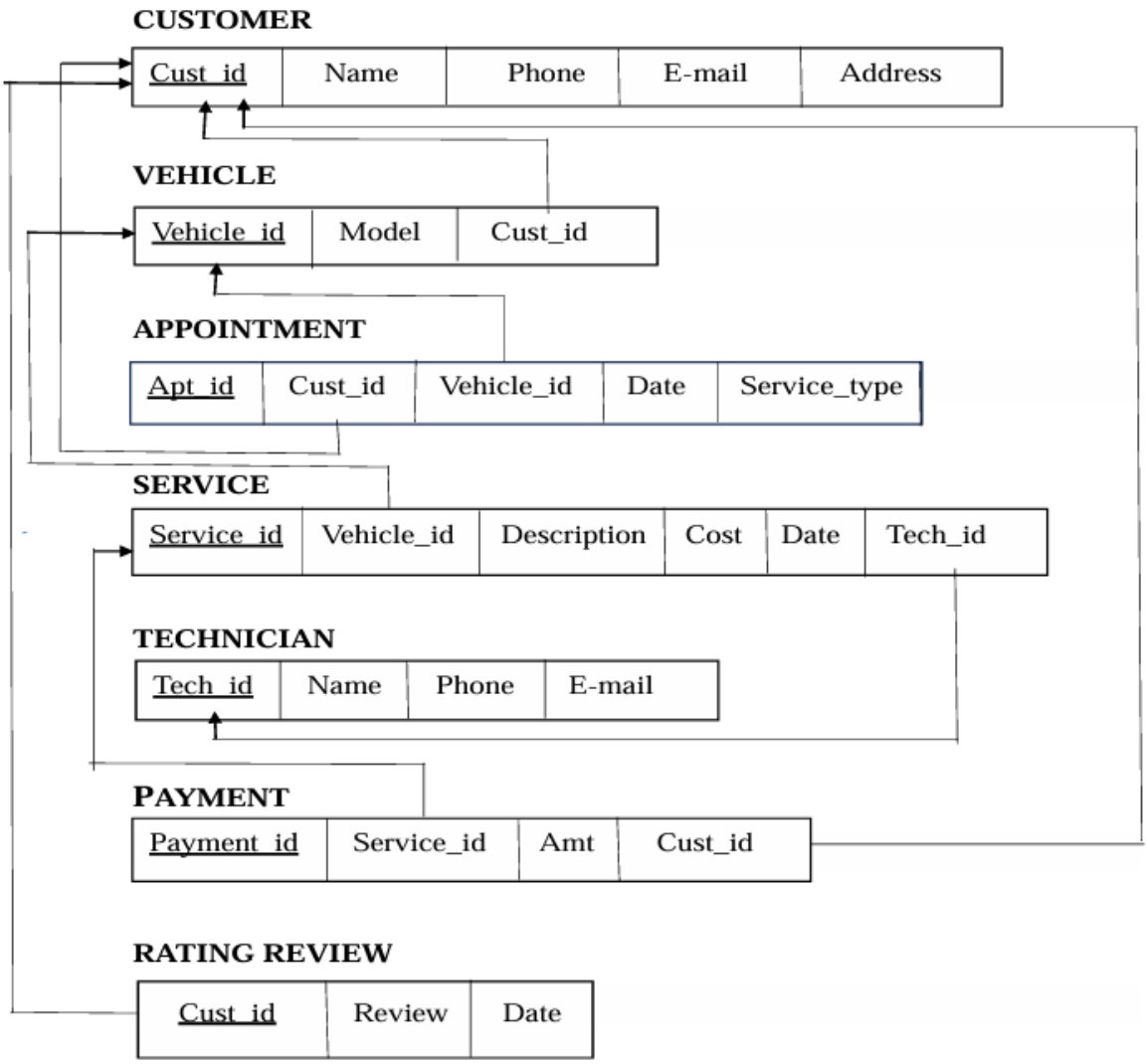
4. List of Software/Tools/Programming Languages Used

- **Frontend**:
    - **HTML5, CSS3**: For building the structure, style, and behavior of the system.
- **Backend**:
    - **PHP**: Handles server-side processing for user authentication, database interactions, and API integration.
- **Database**:
    - **MySQL**: Used to store and manage all data, including vehicle records, customer profiles, service details, and payment history.
- **Tools**:
    - **Visual Studio Code (VS Code)**: The primary IDE for coding and debugging.
    - **Git**: Ensures version control and team collaboration.
    - Apache server (XAMPP).

6. E R Diagram:



7. Relational Schema:



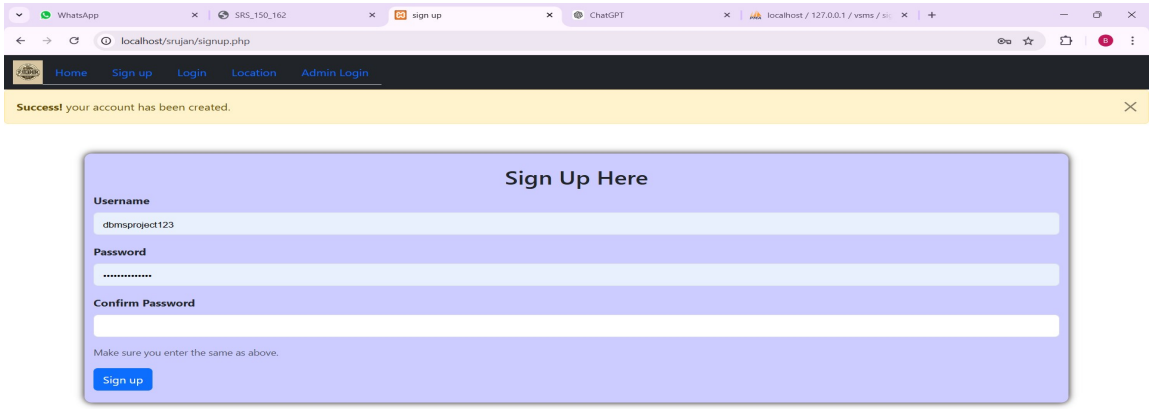
---

8. DDL Commands:

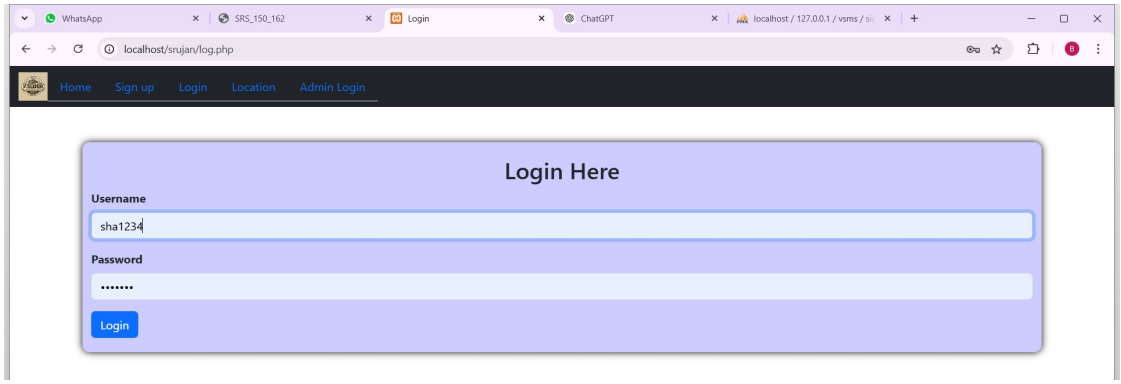
```
<!-- creation of tables -->
create database vsms;

create table customer
(cust_id int primary key,
name varchar (10),
phone number (10),
email varchar (20),
address varchar (15));

create table vehicle
(vehicle_id int primary key,
model varchar (10),
cust_id int,
foreign key (cust_id) references customer(cust_id));

create table appointment
(apt_id int primary key,
cust_id int,
vehicle_id int,
date date,
service_type varchar (20),
foreign key (cust_id) references customer (cust_id),
foreign key (vehicle_id) references vehicle (vehicle_id));

create table technician
(tech_id int primary key,
name varchar (55),
phone number (10),
email varchar (20));

create table service
(service_id int primary key,
vehicle_id int,
tech_id int,
description text,
cost decimal (10,2),
date date,
foreign key (vehicle¬_id) references vehicle (vehicle_id),
foreign key (tech_id) references technician (tech_id));

create table payment
(payment_id int (11) primary key,
service_id int (11),
cost int (10),
cust_id int (11),
foreign key (service_id) references service (service_id),
foreign key (cust_id) references customer (cust_id));

create table rating_review
(customer_id int (11),
review varchar (255),
date date,
foreign key (customer_id) references customer (customer_id));
```

9. CRUD Operation Screenshots

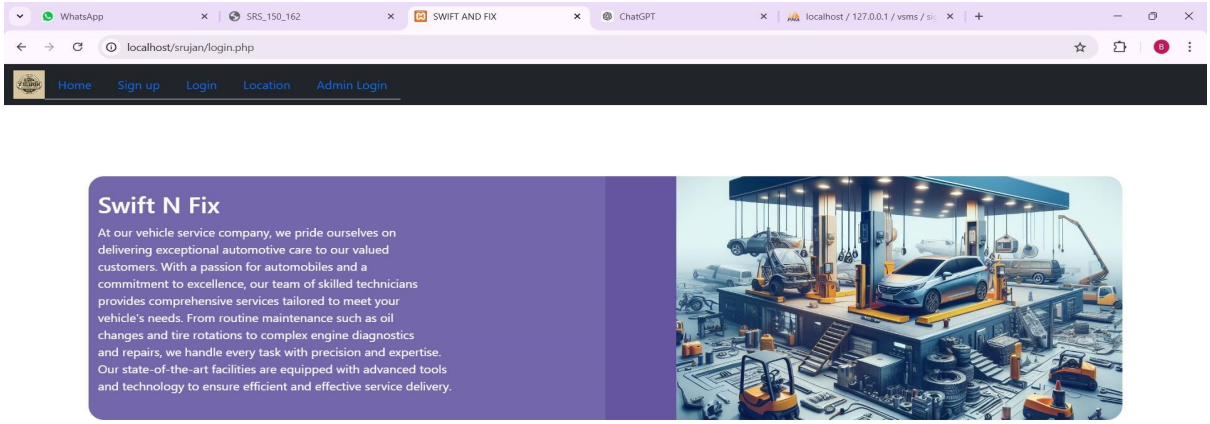
Show actions like:
- Registering a new user.



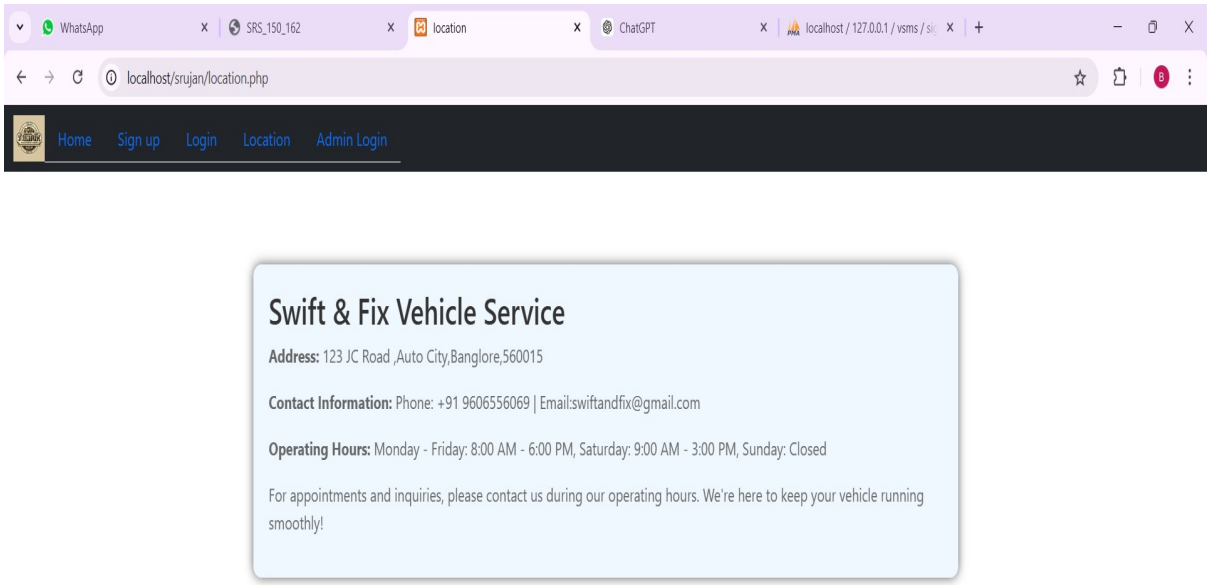
-Login page:



After Login: Home Page



Displaying location of the garage:



The list of tables in our backend are....

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	appointment	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	customer	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	payment	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	rating_review	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	service	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	signup	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	technician	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	technician_revenue	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	vehicle	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	32.0 KiB	-
	9 tables	Sum	17	InnoDB	utf8mb4_general_ci	288.0 KiB	0 B

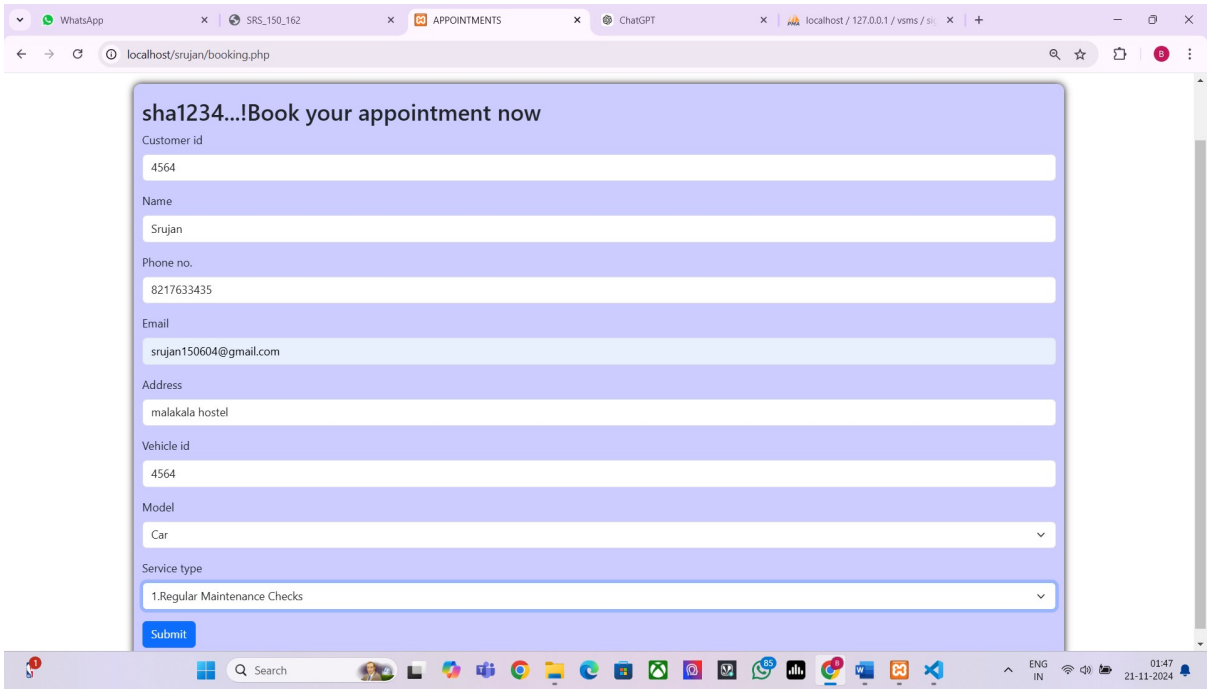
After signing up, the details of the signed user is stored…

		username	password	dt
☐	Edit Copy Delete	sha1234	sha1234	2024-11-19 18:03:05
☐	Edit Copy Delete	sharan	sharan9874	2024-11-21 01:35:40

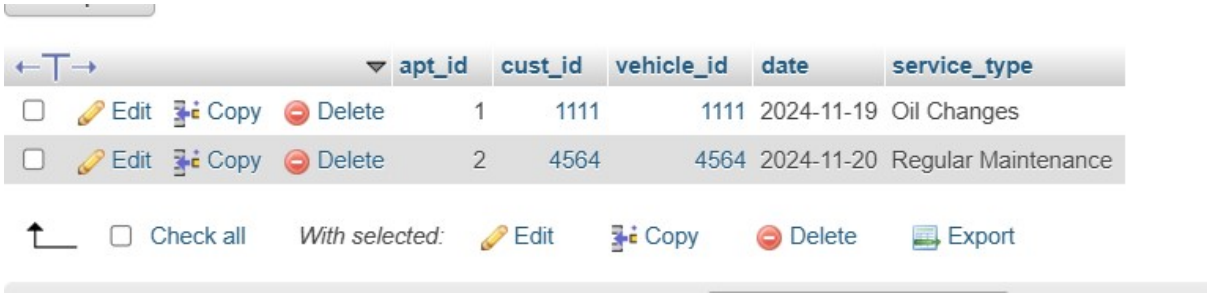
Check all With selected: Edit Copy Delete Export

10. Application Functionality Screenshots:

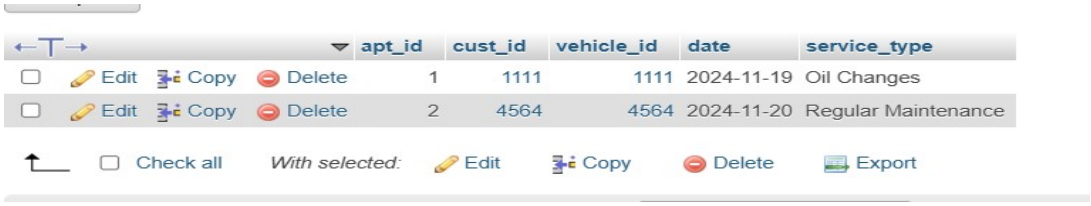
Booking an appointment for a service:



After an appointment is booked , the appointment details gets updated…

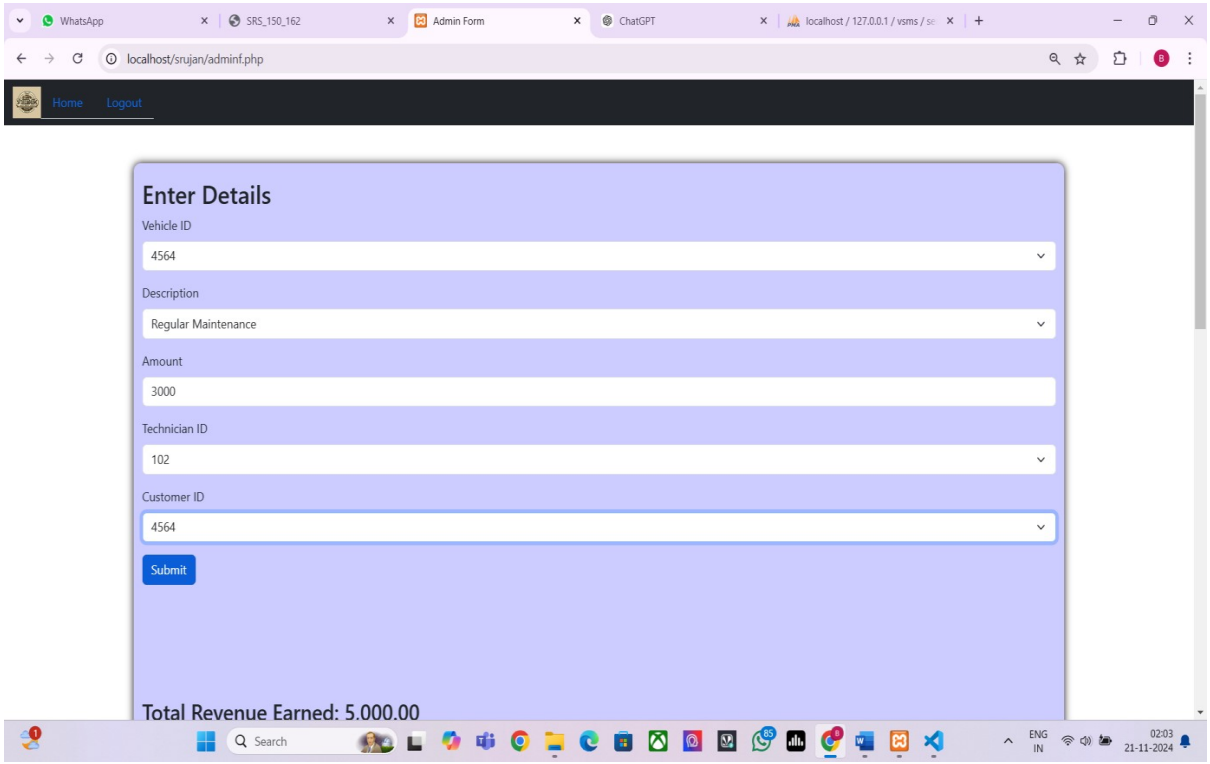


Customer details gets updated…



Vehicle details gets updated…..

Generating bill for the service requested….



After the bill is generated….

Payment table gets updated

	payment_id	service_id	cost	cust_id
Edit Copy Delete	1	0	5000	1111
Edit Copy Delete	2	2	3000	4564
Edit Copy Delete	4	4	3000	4564

Service table also gets updated

	service_id	vehicle_id	tech_id	description	cost	date
Edit Copy Delete	1	1111	100	Oil Changes	5000.00	2024-11-19
Edit Copy Delete	2	4564	102	Regular Maintenance	3000.00	2024-11-20
Edit Copy Delete	4	45 car	102	Regular Maintenance	3000.00	2024-11-20

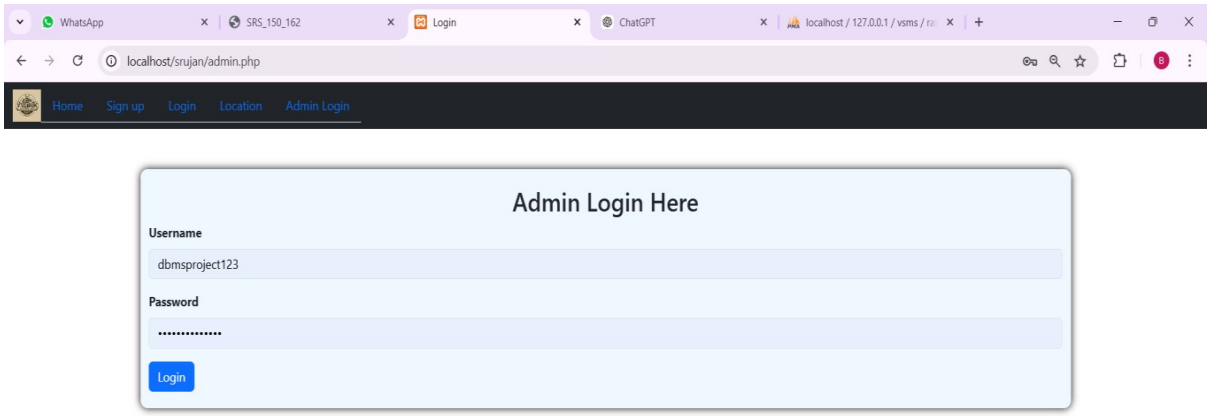
Check all *With selected:* Edit Copy Delete Export

This is the list of technicians working in the garage….

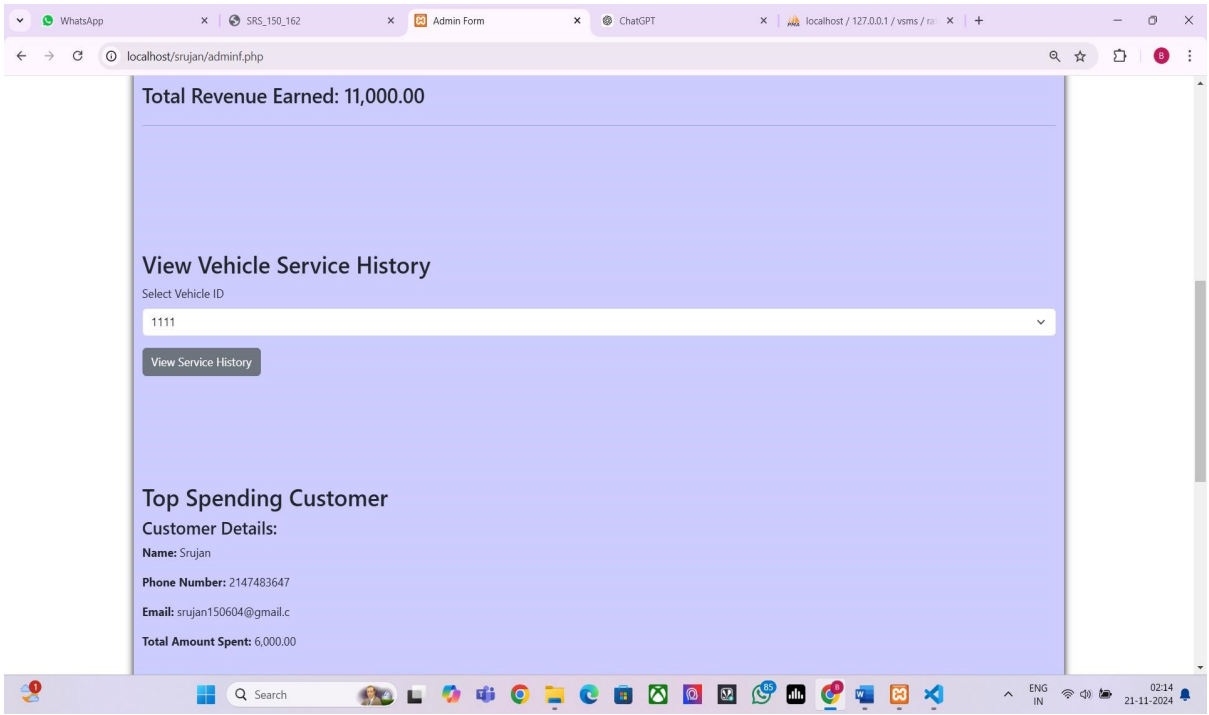
	tech_id	name	phone_no	email
Edit Copy Delete	100	dhanush	111	sr@gmail.com
Edit Copy Delete	101	manish	222	qw@gmail.com
Edit Copy Delete	102	sharan	333	er@gmail.com
Edit Copy Delete	104	moksh	444	yu@gmail.com
Edit Copy Delete	105	jyothi	555	tu@gmail.com

The following is the screenshots of administration page:

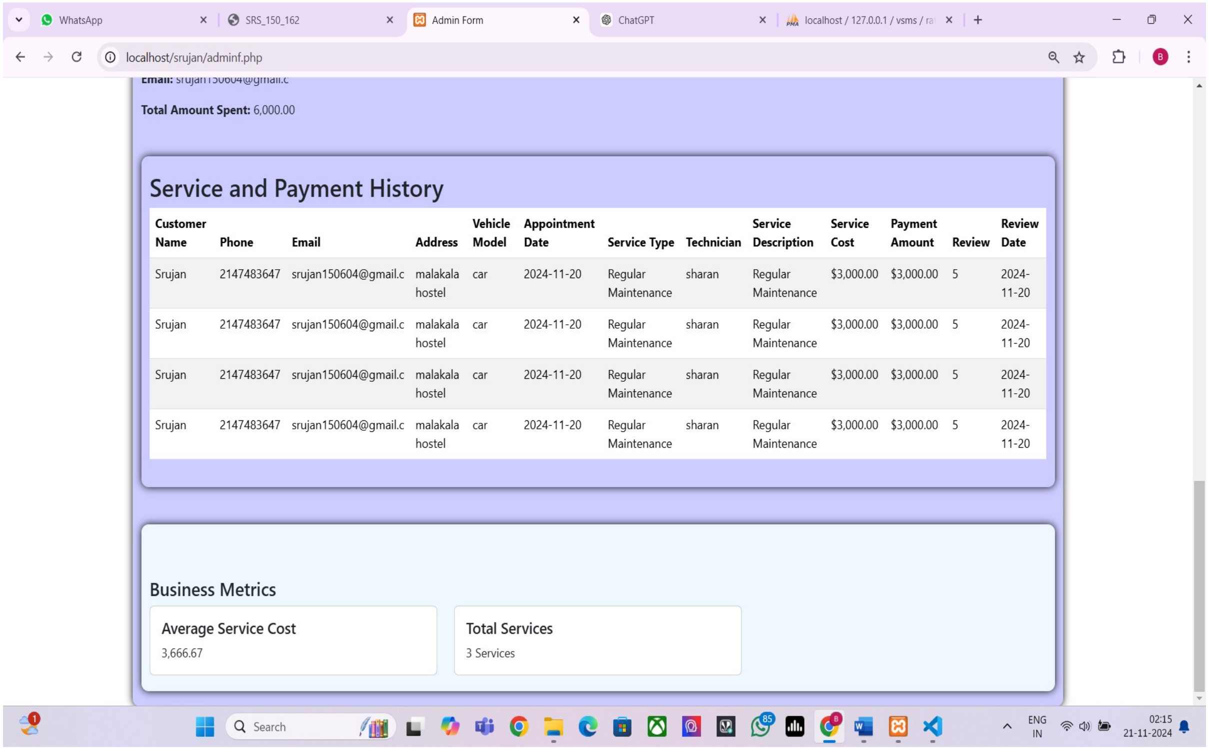
Below screenshot is the login page of admin,



The Admin would be able to look at details like total revenue earned , view vehicle service history and also find out abouttop spending customer.

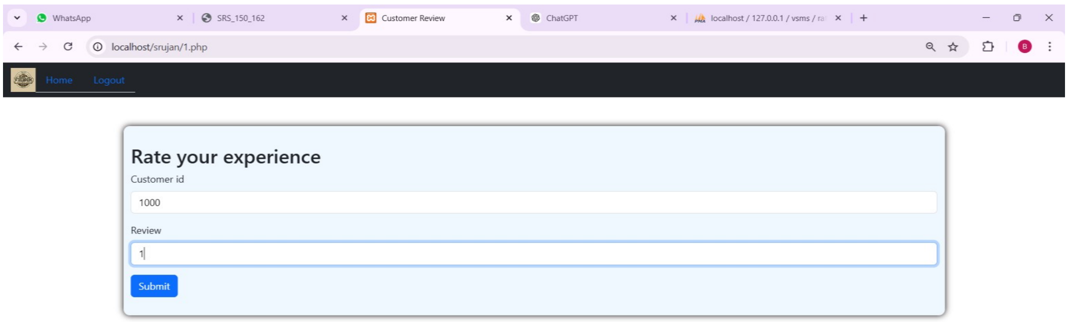


The other functionalities of admin page are it gives a detailed summary about customer details, vehicle details, payment details, technician details, customer feedback, appointment details i.e.,it is a collection of all the tables and also find the business metrics such as average service cost and total no of services rendered till that moment.



Customer Name	Phone	Email	Address	Vehicle Model	Appointment Date	Service Type	Technician	Service Description	Service Cost	Payment Amount	Review	Review Date
Srujan	2147483647	srujan150604@gmail.c	malakala hostel	car	2024-11-20	Regular Maintenance	sharan	Regular Maintenance	$3,000.00	$3,000.00	5	2024-11-20
Srujan	2147483647	srujan150604@gmail.c	malakala hostel	car	2024-11-20	Regular Maintenance	sharan	Regular Maintenance	$3,000.00	$3,000.00	5	2024-11-20
Srujan	2147483647	srujan150604@gmail.c	malakala hostel	car	2024-11-20	Regular Maintenance	sharan	Regular Maintenance	$3,000.00	$3,000.00	5	2024-11-20
Srujan	2147483647	srujan150604@gmail.c	malakala hostel	car	2024-11-20	Regular Maintenance	sharan	Regular Maintenance	$3,000.00	$3,000.00	5	2024-11-20

Customer feedback:



Storing customer feedback in the database,

customer_id	review	date
4564	5	2024-11-20
4564	5	2024-11-20
1000	1	2024-11-20

11. Advanced SQL Features(Code snippets):

All functions,procedures and triggers mentioned below performs their functionality as showed in above screenshots in the frontend interface,

- Functions

Function used to get total revenue:

```php
// Function to get total revenue
function getTotalRevenue($conn) {
    $sql = "SELECT SUM(cost) AS total_revenue FROM payment";
    $result = $conn->query($sql);

    if ($result && $row = $result->fetch_assoc()) {
        return $row['total_revenue'];
    } else {
        return 0;
    }
}
```

Function to get service history of a vehicle:

```php
// Function to get service history for a vehicle
function getServiceHistory($conn, $vehicle_id) {
    $sql = "SELECT s.date AS service_date, s.description AS service_description, s.cost AS service_cost, t.name AS technician_name
            FROM service s
            JOIN technician t ON s.tech_id = t.tech_id
            WHERE s.vehicle_id = ?
            ORDER BY s.date DESC";
    $stmt = $conn->prepare($sql);

    if (!$stmt) {
        die("Error preparing statement: " . $conn->error);
    }

    $stmt->bind_param("s", $vehicle_id);
    $stmt->execute();
    return $stmt->get_result();
}
```

- Stored Procedures

Procedure to get services by technician,

```sql
***Procedure to get services by technician

DELIMITER $$
CREATE PROCEDURE getServicesByTechnician(IN tech_id INT)
BEGIN
    SELECT s.date, s.description, s.cost, c.name AS customer_name, SUM(s.cost) AS total_revenue
    FROM service s
    JOIN customer c ON s.vehicle_id = c.vehicle_id
    WHERE s.tech_id = tech_id
    GROUP BY s.date, s.description, s.cost, c.name
    ORDER BY s.date DESC;
END $$
DELIMITER ;
```

- Triggers

Trigger to prevent appointment overlap,

```
***TRIGGER to prevent appointment overlap
DELIMITER $$

CREATE TRIGGER prevent_duplicate_service_on_same_day
BEFORE INSERT ON service
FOR EACH ROW
BEGIN
    -- Declare a variable to store the count of services assigned to the technician on the same day
    DECLARE service_count INT;

    -- Check if the technician is already assigned to a service on the same date
    SELECT COUNT(*)
    INTO service_count
    FROM service
    WHERE tech_id = NEW.tech_id
    AND DATE(date) = DATE(NEW.date);  -- Compare only the date part, not the time part

    -- If the technician already has a service on the same date, prevent the insert
    IF service_count > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Technician is already assigned to a service on this date.';
    END IF;
END $$

DELIMITER ;
```

12. SQL Queries(Code)

All the below queries perform their functionality as displayed in the above scrrenshots in the frontend interface.

a)Nested query to get the top spending customer:

```
// Nested query to get the top spending customer
function getTopSpendingCustomer($conn) {
    $sql = "
        SELECT c.cust_id, c.name, c.phone, c.email, SUM(p.cost) AS total_spent
        FROM customer c
        JOIN payment p ON c.cust_id = p.cust_id
        GROUP BY c.cust_id
        HAVING SUM(p.cost) = (
            SELECT MAX(total_spent)
            FROM (
                SELECT SUM(cost) AS total_spent
                FROM payment
                GROUP BY cust_id
            ) AS subquery
        )
    ";

    $result = $conn->query($sql);
    if ($result && $row = $result->fetch_assoc()) {
        return $row;
    } else {
        return null;
    }
}
```

b) Sql query using join operation to fetch the complete data of all tables.

```
// SQL query to fetch the required data
$sql = "
    SELECT
        c.name AS customer_name,
        c.phone AS customer_phone,
        c.email AS customer_email,
        c.address AS customer_address,
        v.model AS vehicle_model,
        a.date AS appointment_date,
        a.service_type AS service_type,
        t.name AS technician_name,
        s.description AS service_description,
        s.cost AS service_cost,
        p.cost AS payment_amount,
        r.review AS customer_review,
        r.date AS review_date
    FROM
        customer c
    JOIN
        vehicle v ON c.cust_id = v.cust_id
    JOIN
        appointment a ON v.vehicle_id = a.vehicle_id
    JOIN
        service s ON a.vehicle_id = s.vehicle_id
    JOIN
        technician t ON s.tech_id = t.tech_id
    JOIN
        payment p ON p.service_id = s.service_id
    LEFT JOIN
        rating_review r ON c.cust_id = r.customer_id
    ORDER BY
        a.date DESC;
";
```

c)Aggregate query to find the Business Metrics such as average service cost and total number of services rendered.

```
// Updated aggregate query (without SUM of payments)
$aggregateSql = "
    SELECT
        AVG(s.cost) AS average_service_cost,
        COUNT(DISTINCT s.service_id) AS total_services
    FROM
        service s
";
```