WhatNext Vision Motors: Shaping the Future of Mobility with Innovation and Excellence

Project Overview

WhatNext Vision Motors is a Salesforce-based Customer Relationship Management (CRM) project created to improve the way vehicle orders and customer services are managed. The system helps the company work more efficiently by automating important tasks such as finding the nearest dealer, checking vehicle availability, processing customer orders, scheduling test drives, and handling service requests. Through these automated processes, the system reduces manual work, speeds up transactions, and improves overall service delivery.

All important information such as customer records, dealer information, vehicle details, orders, test drives, and service requests is stored in one centralized system. This allows the company to easily monitor its operations and manage every stage of the sales and service process more effectively. The system also includes useful features such as automatic dealer assignments, real-time stock checking, scheduled notifications, and batch updating of order statuses. These functions help reduce errors, improve accuracy, and ensure smoother workflow management.

Overall, this CRM system increases productivity, improves communication between customers and dealers, and leads to better customer satisfaction. It also supports the company’s goal of providing reliable and modern mobility solutions through a well-organized and technology-driven service approach.

Objectives

The main objective of this project is to design and implement a customized Salesforce CRM system for WhatNext Vision Motors in order to improve the vehicle ordering process, automate important business tasks, and increase overall service efficiency. The system is intended to support the company in managing its operations more effectively by using a centralized and well-structured platform.

By developing a unified system to manage customer information, dealer records, vehicle data, orders, test drives, and service requests, this project aims to:

- simplify the vehicle ordering process through automatic dealer assignment, order checking, and status updates;
- improve the accuracy and visibility of vehicle stock to avoid ordering unavailable units and reduce delays in processing;
- enhance the customer experience by providing faster transactions, automatic reminders, and clearer communication;
- increase operational efficiency by combining essential processes into one integrated Salesforce system; and
- promote customer satisfaction and loyalty by delivering a smooth, transparent, and dependable service experience.

Phase 1: Requirement Analysis & Planning

- **Understanding Business Requirements**

  During this phase, the project focuses on collecting and analyzing the business needs of WhatNext Vision Motors in order to design a CRM system that supports its operations effectively. This includes identifying existing issues in the vehicle ordering process, dealer management, and service request handling. The goal is to ensure that the system addresses current challenges and improves overall workflow efficiency.

  The system is required to:

  i. manage user information including name, email address, phone number, and physical address;
  ii. identify weaknesses and delays in vehicle ordering, dealer management, and service request processes;

iii. ensure that customer, vehicle, and order data are recorded accurately and remain easy to access; and

iv. apply secure, role-based access for sales staff, dealership managers, IT administrators, and other authorized users to protect data and control system usage.

- **Defining Project Scope and Objectives**

Project Scope

The scope of this project is to design and develop a Salesforce-based Customer Relationship Management (CRM) system for WhatNext Vision Motors. The system will serve as a centralized platform for managing vehicle orders, customer information, and service-related processes. It will focus on improving business operations through automation and better data management.

Project Objectives:

The objectives of this project are to:

i. simplify and improve the vehicle ordering process;

ii. increase the accuracy and visibility of vehicle stock information;

iii. improve the overall customer experience through faster and clearer service processes;

iv. enhance operational efficiency by reducing manual tasks and organizing workflows; and

v. strengthen customer satisfaction and loyalty by providing reliable and transparent services.

- **Design Data Model and Security Model:**

During this stage, the data model and security structure of the system are carefully designed to support accurate data management and secure access control.

The project includes:

i. defining clear relationships among system objects such as Vehicle, Dealer, Customer, Order, Test Drive, and Service Request to ensure organized data storage and easy information retrieval; and
ii. implementing security controls using user profiles, roles, and permission sets to manage access rights and protect sensitive information based on user responsibilities.

- **Stakeholders Mapping:**

Stakeholders are identified and grouped based on their level of involvement in the CRM system.

The project identifies:

i. **Primary stakeholders**, which include the sales team, dealership managers, customers, and IT administrators who directly use and depend on the system for daily operations; and
ii. **Secondary stakeholders**, which include senior management and support staff who are indirectly involved and rely on reports and system outputs for decision-making and support functions.

- **Execution Roadmap:**

The execution roadmap outlines the overall plan for managing the project from the initial analysis stage to the final system deployment.

The roadmap includes:

i. a detailed, phase-by-phase plan covering requirement analysis, system design, development, testing, and deployment; and
ii. clearly defined milestones for each major activity, including development completion, system testing, and final deployment.

Phase 2: Salesforce Development – Backend & Configurations

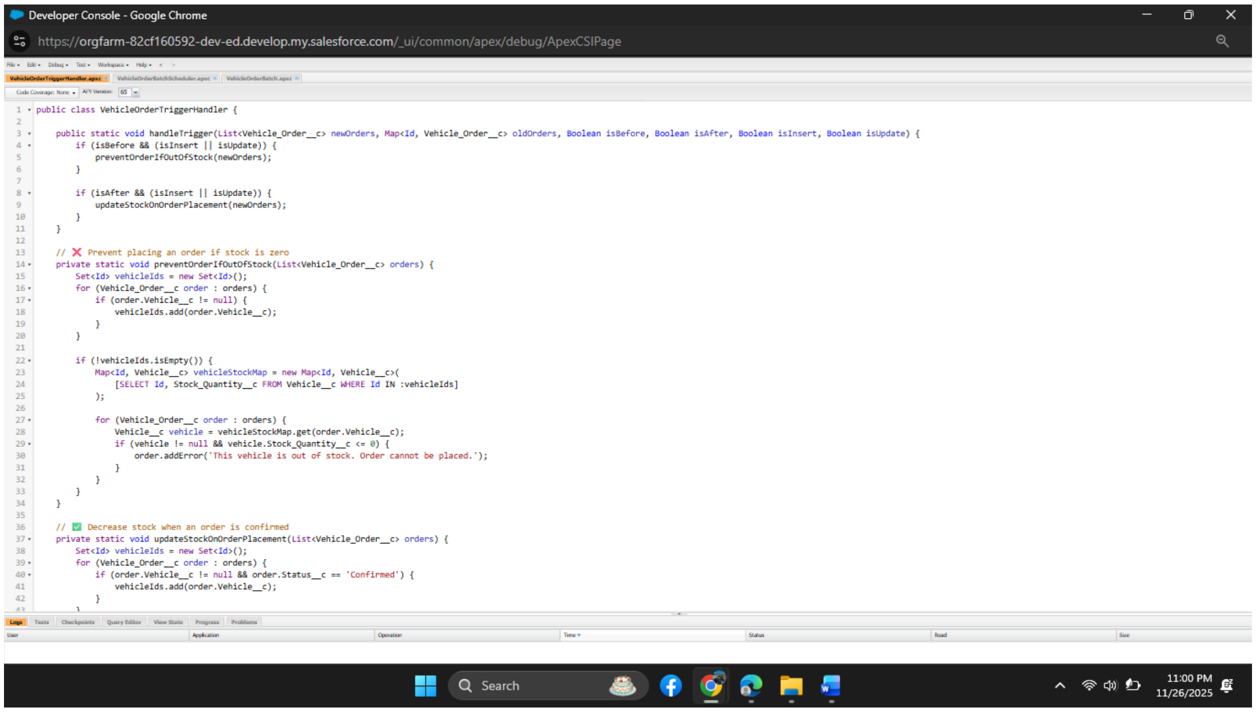
- **Environment Setup and DevOps:**
  - Configured a Salesforce sandbox environment to support system development and testing.
  - Established version control and deployment workflows using tools such as Visual Studio Code and Git to manage code changes efficiently and safely.

- **Customization of Objects, Fields, Validation Rules, and Automation:**
  - Created custom objects including Vehicle, Dealer, Customer, Order, Test Drive, and Service Request to support core business operations.
  - Customized object fields to capture necessary data. For example, the Vehicle object includes:
    - Vehicle_Name__c (Text)
    - Vehicle_Model__c (Picklist: Sedan, SUV, EV, etc.)
    - Stock_Quantity__c (Number)
    - Price__c (Currency)
    - Dealer__c (Lookup to Dealer__c)
    - Status__c (Picklist: Available, Out of Stock, Discontinued)
  - Configured workflow rules, process builders, and flows to automate key business processes and reduce manual tasks.

- **Apex Classes, Triggers, and Asynchronous Apex:**
  - Developed Apex classes and triggers to handle backend operations and ensure smooth automation of complex business logic.
  - Implemented asynchronous Apex where necessary to improve performance and manage large-scale processes efficiently.

- **Documentation:**
  - Screenshots of Apex Class and Apex Triggers.

VehicleTriggerHandler:

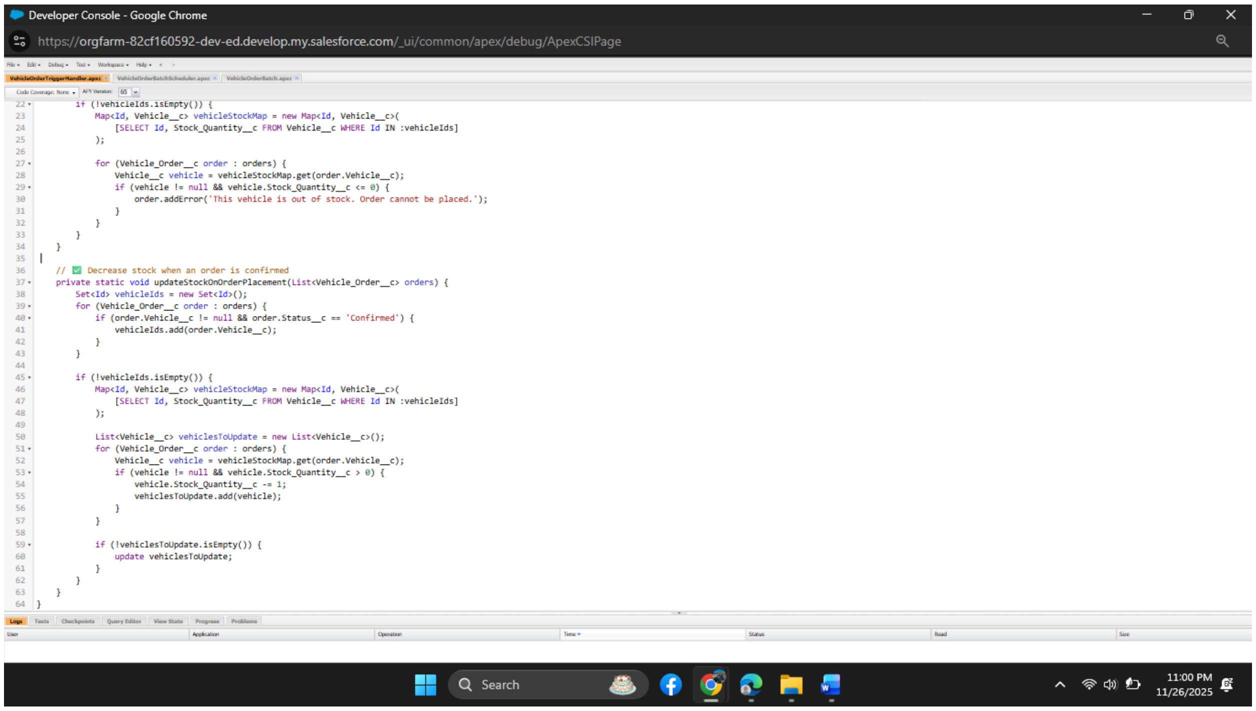


Developer Console - Google Chrome

https://orgfarm-82cf160592-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

```
public class VehicleOrderTriggerHandler {

    public static void handleTrigger(List<Vehicle_Order__c> newOrders, Map<Id, Vehicle_Order__c> oldOrders, Boolean isBefore, Boolean isAfter, Boolean isInsert, Boolean isUpdate) {
        if (isBefore && (isInsert || isUpdate)) {
            preventOrderIfOutOfStock(newOrders);
        }

        if (isAfter && (isInsert || isUpdate)) {
            updateStockOnOrderPlacement(newOrders);
        }
    }

    // ❌ Prevent placing an order if stock is zero
    private static void preventOrderIfOutOfStock(List<Vehicle_Order__c> orders) {
        Set<Id> vehicleIds = new Set<Id>();
        for (Vehicle_Order__c order : orders) {
            if (order.Vehicle__c != null) {
                vehicleIds.add(order.Vehicle__c);
            }
        }

        if (!vehicleIds.isEmpty()) {
            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
            );

            for (Vehicle_Order__c order : orders) {
                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
                if (vehicle != null && vehicle.Stock_Quantity__c <= 0) {
                    order.addError('This vehicle is out of stock. Order cannot be placed.');
                }
            }
        }
    }

    // ✅ Decrease stock when an order is confirmed
    private static void updateStockOnOrderPlacement(List<Vehicle_Order__c> orders) {
        Set<Id> vehicleIds = new Set<Id>();
        for (Vehicle_Order__c order : orders) {
            if (order.Vehicle__c != null && order.Status__c == 'Confirmed') {
                vehicleIds.add(order.Vehicle__c);
            }
```

VehicleTriggerHandler:

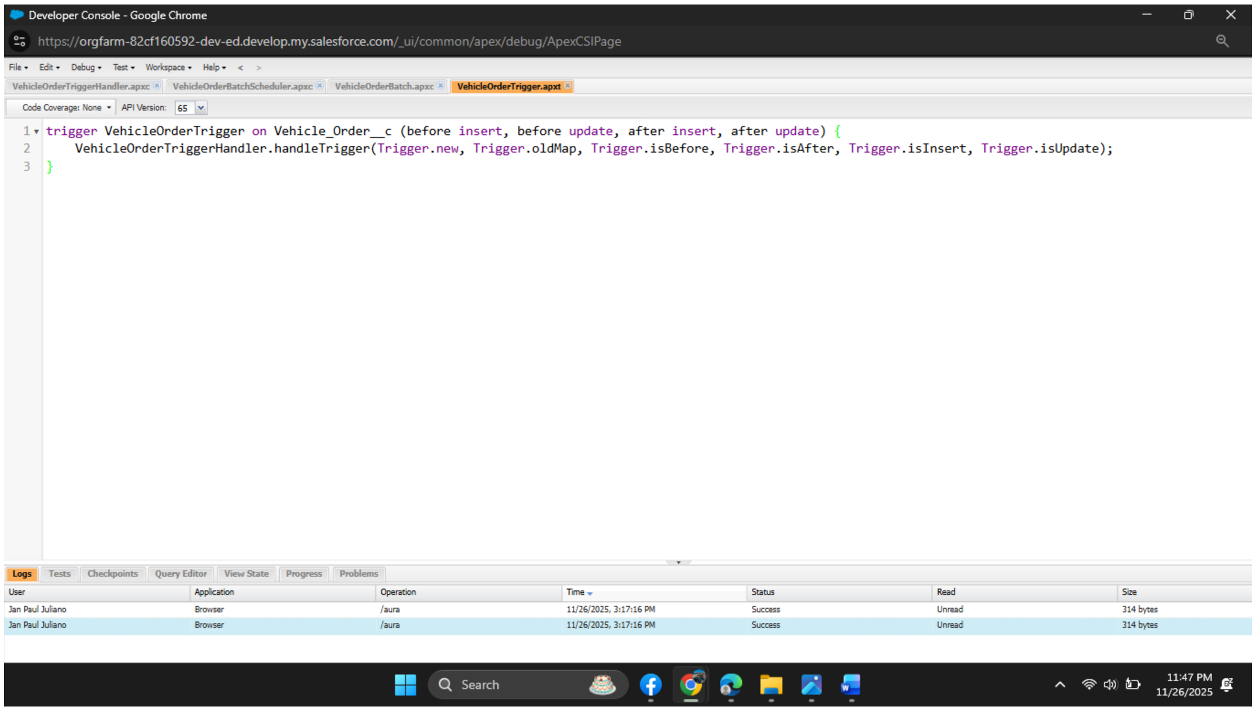


Developer Console - Google Chrome

https://orgfarm-82cf160592-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

```
        if (!vehicleIds.isEmpty()) {
            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
            );

            for (Vehicle_Order__c order : orders) {
                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
                if (vehicle != null && vehicle.Stock_Quantity__c <= 0) {
                    order.addError('This vehicle is out of stock. Order cannot be placed.');
                }
            }
        }
    }

    // ✅ Decrease stock when an order is confirmed
    private static void updateStockOnOrderPlacement(List<Vehicle_Order__c> orders) {
        Set<Id> vehicleIds = new Set<Id>();
        for (Vehicle_Order__c order : orders) {
            if (order.Vehicle__c != null && order.Status__c == 'Confirmed') {
                vehicleIds.add(order.Vehicle__c);
            }
        }

        if (!vehicleIds.isEmpty()) {
            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
            );

            List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();
            for (Vehicle_Order__c order : orders) {
                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
                if (vehicle != null && vehicle.Stock_Quantity__c > 0) {
                    vehicle.Stock_Quantity__c -= 1;
                    vehiclesToUpdate.add(vehicle);
                }
            }

            if (!vehiclesToUpdate.isEmpty()) {
                update vehiclesToUpdate;
            }
        }
    }
}
```

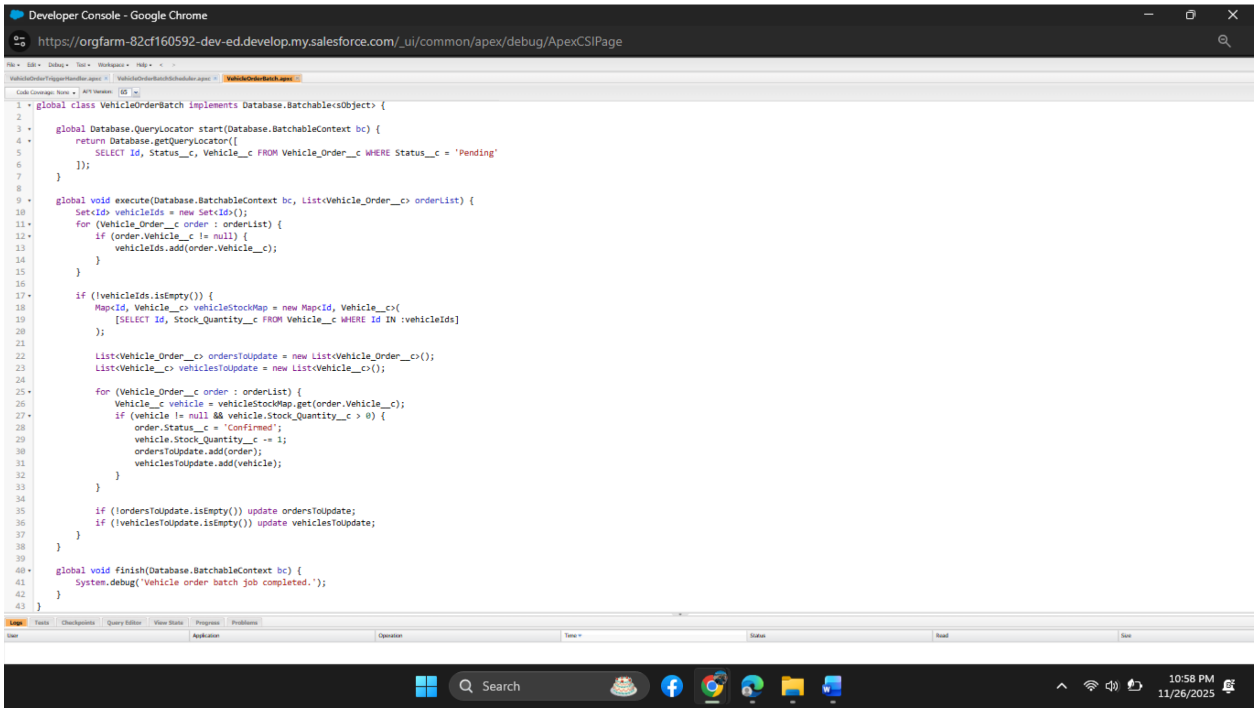
VehicleOrderTrigger:



```
trigger VehicleOrderTrigger on Vehicle_Order__c (before insert, before update, after insert, after update) {
    VehicleOrderTriggerHandler.handleTrigger(Trigger.new, Trigger.oldMap, Trigger.isBefore, Trigger.isAfter, Trigger.isInsert, Trigger.isUpdate);
}
```

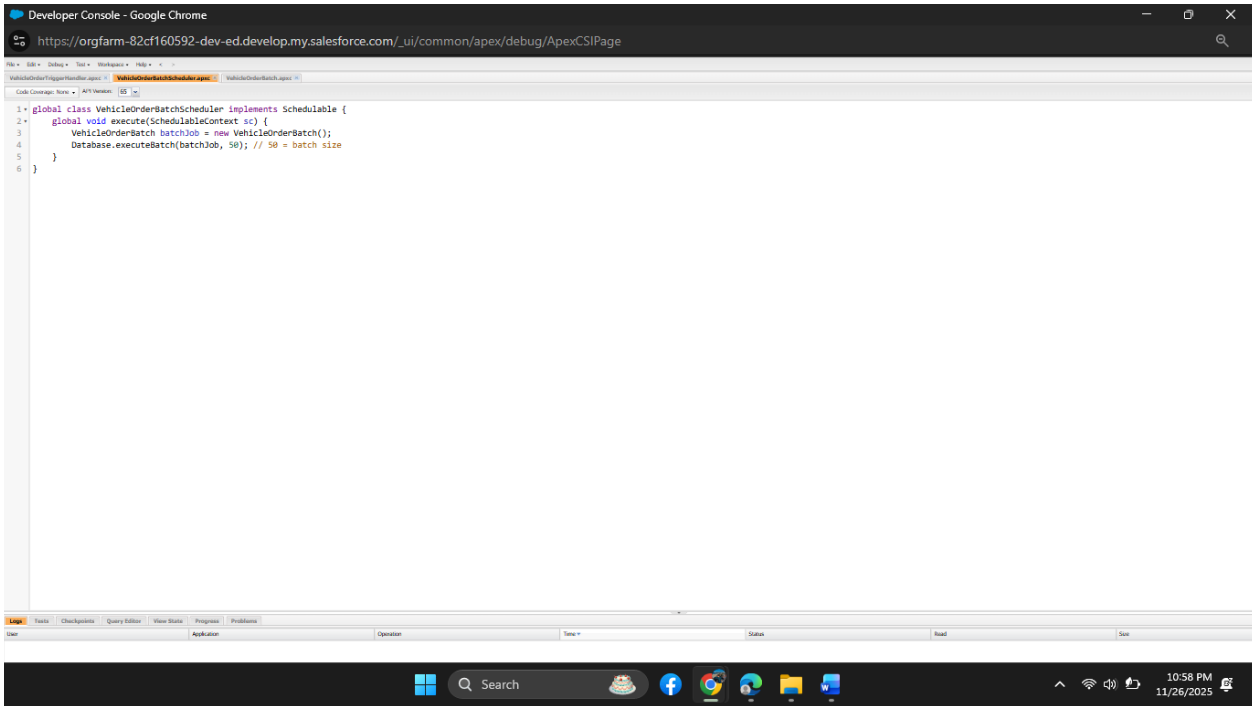
User	Application	Operation	Time	Status	Read	Size
Jan Paul Juliano	Browser	/aura	11/26/2025, 3:17:16 PM	Success	Unread	314 bytes
Jan Paul Juliano	Browser	/aura	11/26/2025, 3:17:16 PM	Success	Unread	314 bytes

VehicleOrderBatch:



```
global class VehicleOrderBatch implements Database.Batchable<sObject> {

    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator([
            SELECT Id, Status__c, Vehicle__c FROM Vehicle_Order__c WHERE Status__c = 'Pending'
        ]);
    }

    global void execute(Database.BatchableContext bc, List<Vehicle_Order__c> orderList) {
        Set<Id> vehicleIds = new Set<Id>();
        for (Vehicle_Order__c order : orderList) {
            if (order.Vehicle__c != null) {
                vehicleIds.add(order.Vehicle__c);
            }
        }

        if (!vehicleIds.isEmpty()) {
            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
            );

            List<Vehicle_Order__c> ordersToUpdate = new List<Vehicle_Order__c>();
            List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();

            for (Vehicle_Order__c order : orderList) {
                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
                if (vehicle != null && vehicle.Stock_Quantity__c > 0) {
                    order.Status__c = 'Confirmed';
                    vehicle.Stock_Quantity__c -= 1;
                    ordersToUpdate.add(order);
                    vehiclesToUpdate.add(vehicle);
                }
            }

            if (!ordersToUpdate.isEmpty()) update ordersToUpdate;
            if (!vehiclesToUpdate.isEmpty()) update vehiclesToUpdate;
        }
    }

    global void finish(Database.BatchableContext bc) {
        System.debug('Vehicle order batch job completed.');
    }
}
```

VehicleOrderBatchScheduler:



```
global class VehicleOrderBatchScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        VehicleOrderBatch batchJob = new VehicleOrderBatch();
        Database.executeBatch(batchJob, 50); // 50 = batch size
    }
}
```

Phase 3: UI/UX Development and Customization

- **Lightning App Setup:**
    - Developed a custom Salesforce Lightning App named **WhatNext Vision Motors** to serve as the main interface for all users.

- **Page Layouts and Dynamic Forms:**
    - Organized fields, buttons, and related lists for each object to ensure clear and efficient workflows.
    - Configured page layouts according to different user roles for objects such as Vehicle, Dealer, Customer, Order, Test Drive, and Service Request.
    - Implemented dynamic forms to improve usability and provide a more responsive user experience.

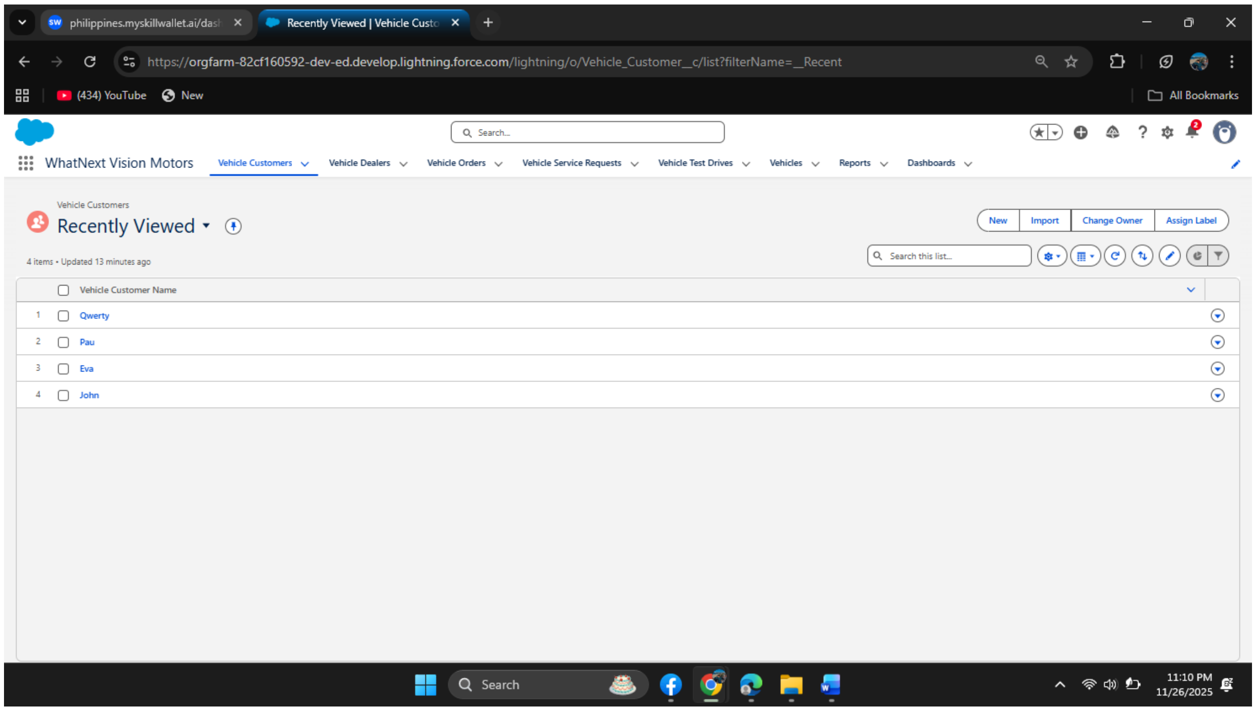
- **User Management:**
  - Set up user profiles, roles, and permission sets to control access and define what each user can view or modify within the system.

- **Reports and Dashboards:**
  - Designed real-time dashboards and reports to monitor sales performance, track orders, and provide management with actionable insights.

- **Lightning Web Components (LWC) Development:**
  - Developed Lightning Web Components for advanced user interface functionality where required, improving interaction and responsiveness.

- **Lightning Pages:**
  - Customized Lightning Pages to simplify navigation and enhance the overall usability of the CRM system.

- **Documentation:**



Vehicle Customers
Recently Viewed
4 items • Updated 13 minutes ago

	Vehicle Customer Name
1	Qwerty
2	Pau
3	Eva
4	John

Phase 4: Data Migration, Testing, and Security

- **Data Loading Process:**
  - Performed data migration through manual record entry, as demonstrated in the project video, to ensure all relevant information was accurately captured in the new system.

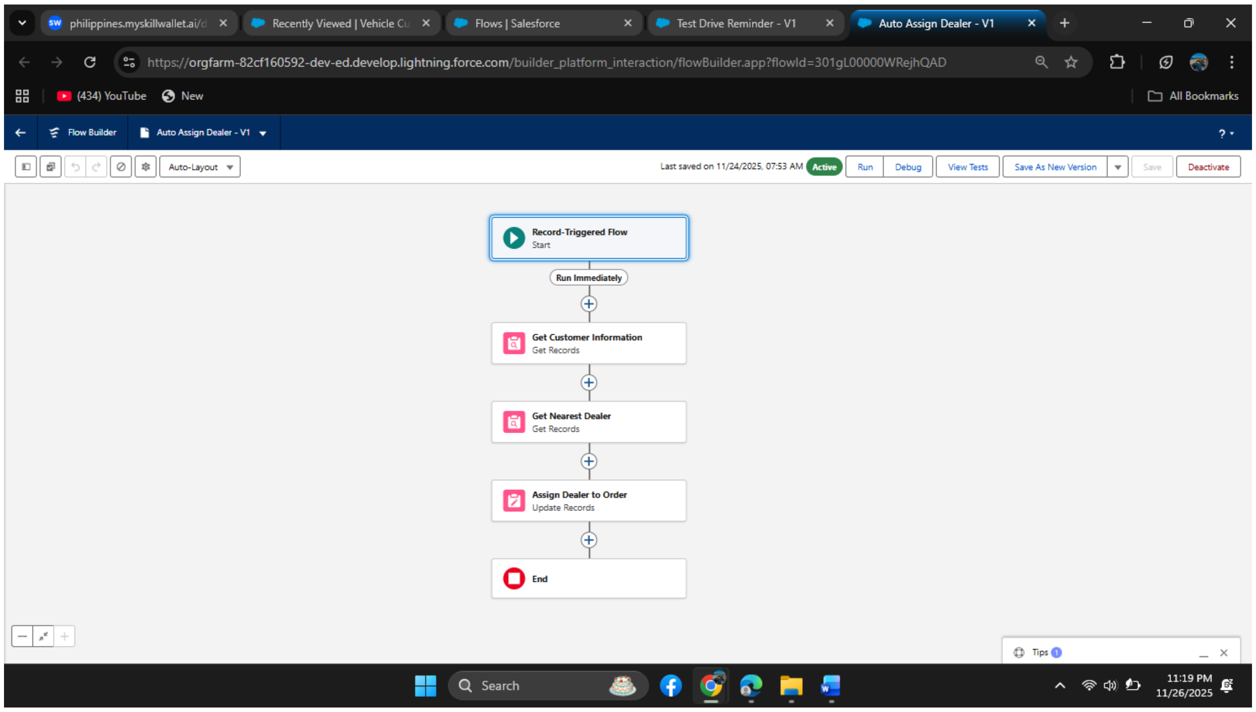
- **Field History Tracking, Duplicate, and Matching Rules:**
  - Implemented field history tracking to monitor changes in critical data fields.
  - Configured duplicate and matching rules to alert users when a new Lead matches an existing Contact based on email or phone number, helping maintain data integrity.

- **Profiles, Roles, Permission Sets, and Sharing Rules:**
  - Ensured secure access by configuring user profiles, roles, permission sets, and sharing rules, providing appropriate visibility and control based on user responsibilities.

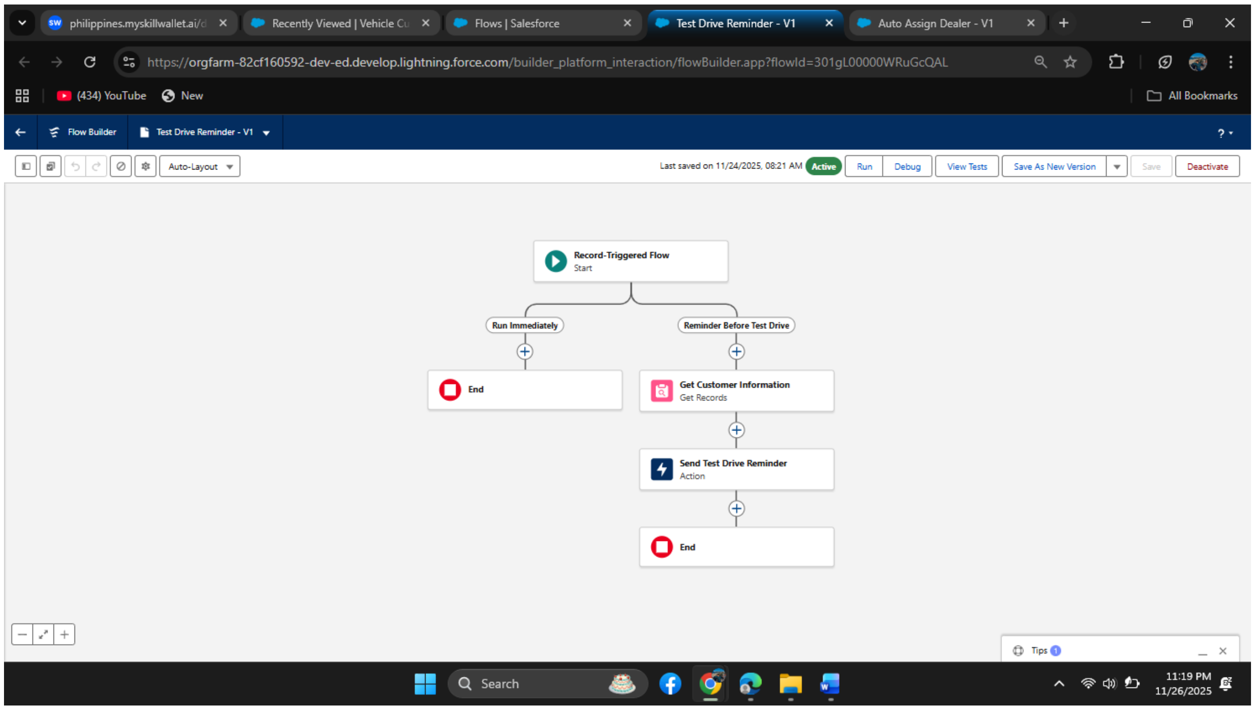
- **Creation of Test Classes:**
  - Developed test classes for Apex code to verify proper execution, maintain code quality, and ensure reliable system performance.

- **Documentation:**
  - Screenshot of the flows which are Auto Assign Dealer and Test Drive Reminder.

Auto Assign Dealers Flow:



Test Drive Reminder Flow:



Phase 5: Deployment, Documentation, and Maintenance

- **Deployment Strategy**
  - Deployed system changes using Salesforce change sets and metadata deployment tools to ensure smooth implementation and minimal disruption to business operations.

- **System Maintenance and Monitoring**
  - Conducted regular system updates, backups, and performance monitoring to maintain reliability and efficiency.
  - Provided ongoing support for users, addressing any issues promptly to ensure continuous operation.

- **Troubleshooting Documentation**
  - Prepared detailed documentation outlining step-by-step procedures for identifying and resolving common system issues.
  - Included guidance on reviewing flows, automation processes, and examining Apex debug logs to assist administrators in troubleshooting effectively.

Conclusion

The Salesforce CRM system for WhatNext Vision Motors has successfully modernized the vehicle ordering process, automated essential operations, and improved overall efficiency. By centralizing information about customers, dealers, and vehicles, the system has enhanced decision-making, minimized manual errors, and provided a scalable platform to support future growth.

Looking ahead, potential improvements could include integrating advanced marketing automation, using AI-powered analytics for sales forecasting, developing a mobile-friendly interface for on-the-go access, and expanding reporting features to gain deeper insights into customer behavior and dealer performance. These enhancements will help ensure that the CRM system continues to meet the evolving needs of the company and supports its goal of providing innovative and seamless mobility solutions.