Jancee Joy T. Dolosa

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

Project Overview

The WhatNext Vision Motors project establishes a comprehensive Salesforce CRM solution designed to elevate the company's position in the automotive mobility sector by transforming its customer lifecycle management, thereby addressing the core business need for a seamless and error-free ordering experience. The system serves as a unified platform to store and manage critical assets, including vehicle details, stock availability, and dealer information, while also efficiently tracking the entire customer journey across orders, test drives, and service requests. To ensure efficiency and data integrity, the CRM incorporates key automation features: it uses Apex logic to proactively prevent the placement of orders if the vehicle is out of stock, automatically assigns new orders to the nearest dealer based on the customer’s location, and enhances communication by sending automated email reminders for scheduled test drives and notifications for stock replenishment. This comprehensive implementation ensures improved fulfillment accuracy, streamlined operational efficiency, and an elevated customer experience.

Objectives

The primary goals for the development of the WhatNext Vision Motors CRM solution are to fundamentally improve the customer ordering experience and maximize internal process efficiency.

These objectives are:

- **Enhance customer convenience and efficiency.** the CRM's first objective is to automate the suggestion of the nearest dealer based on the customer’s address, which significantly reduces the time and effort required from the customer, resulting in a more customer-friendly experience.
- **Ensure order integrity and fulfillment accuracy.** the system's second objective is to implement proactive validation that strictly blocks customers from placing orders for vehicles that are currently out of stock, thereby eliminating potential confusion and directly improving customer satisfaction.
- **Streamline operational workflows and communication.** the final objective is to establish a scheduled process that accurately and transparently updates bulk order statuses (to 'Confirmed' or 'Pending') based on real-time stock availability, which reduces the administrative burden on staff and provides clear communication to customers regarding their order status.

Phase 1: Requirement Analysis & Planning

Understanding Business Requirements:

The planning phase commenced with a thorough analysis of the current state and identification of key pain points within WhatsNext Vision Motors' mobility and sales operations. The CRM is designed to provide immediate solutions to critical user needs, moving the company from manual, error-prone processes to automated, high-accuracy workflows.

Defining Project Scope and Objectives

- Implementation of the core data model (Custom Objects for Vehicles, Orders, Dealers, Test Drives, and Service Requests).
- Creation of a custom Lightning App and related record pages.
- Deployment of all Process Automation (Flows and Apex Triggers) for stock validation, dealer assignment, and reminder emails.
- Development of Batch and Scheduled Apex jobs for periodic stock level checks and notification processing.
- Configuration of a robust Security Model using Profiles, Permission Sets, and Field-Level Security.

Design Data Model and Security Model

- Key data relationships, primarily Lookup relationships, are established to link Orders to both the specific Vehicle being purchased and the assigned Dealer Location.
- The security architecture is established using a layered approach starting with a Role Hierarchy to define organizational structure and control data sharing across roles.
- Profiles are used to set the base level of access for groups like Sales Agents and Fulfillment Managers.

Stakeholders Mapping

- Executive Leadership (Executives): Serves as the Project Sponsor and provides final approval, primarily interested in the project's strategic alignment and overall return on investment (ROI).
- Sales Team (Users): These End Users are focused on the efficiency of the order creation process and the automated dealer assignment feature, and will participate actively in User Acceptance Testing (UAT).
- Fulfillment Team (Users): As Process Owners for inventory and fulfillment, they are critically interested in the stock validation and the accuracy of the automated status updates.
- IT/Development Team (Admins): They are the Solution Architects and Developers responsible for system architecture, code quality, and long-term system maintenance.

- Customer Support (Users): This team is primarily concerned with the tracking features for customer engagements, specifically Test Drive reminders and the efficient management of Service Requests.

Execution Roadmap

- Analysis & Planning, Core Development, Programmatic Build, Testing & Deployment

Phase 2: Salesforce Development - Backend & Configurations

Setup Environment & DevOps Workflow

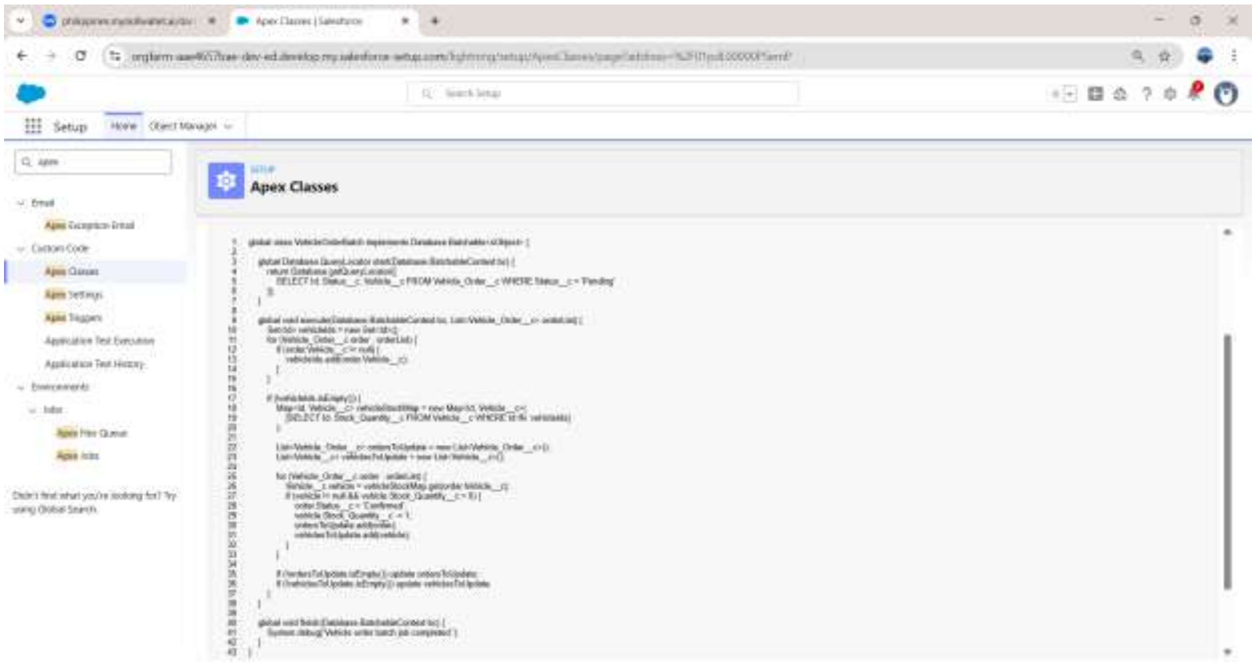
- All development activities, including object creation, flow building, and Apex coding, were executed within a dedicated Salesforce Developer Sandbox environment to isolate changes and prevent disruption to the production org.

Customization of Objects and Fields

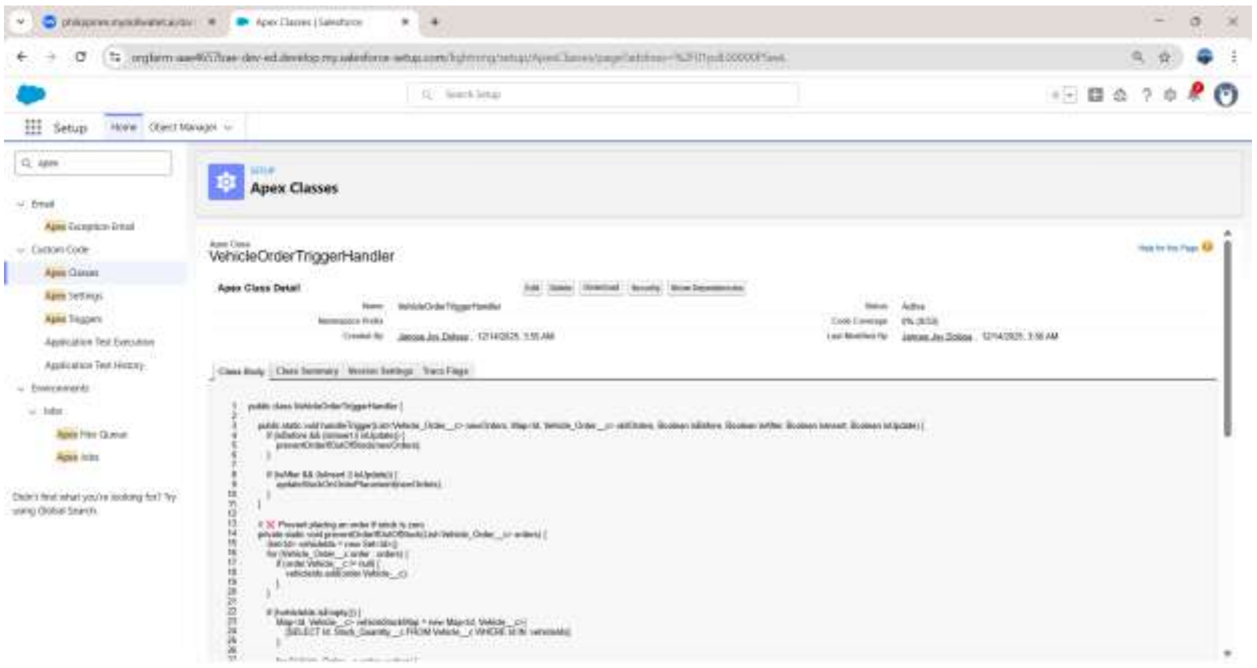
- The workflow rules, Process Builder, and Flows were implemented to automating the repetitive tasks.

Apex Classes, Triggers, and Asynchronous Apex Classes

- Apex Triggers (OrderTrigger and OrderTriggerHandler) were implemented to enforce synchronous, high-performance business rules, specifically to prevent order placement if the vehicle is out of stock and to perform complex distance calculations to auto-assign orders to the nearest dealer upon creation.
- For asynchronous processing of large data volumes, a Batch Apex Class was developed to periodically check vehicle stock levels across the entire database and update availability.
- A corresponding Scheduled Apex Class (ScheduledStockNotifier) was configured to run daily to check for vehicles needing replenishment after the batch job completes and to



```
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.');
    }
}
```

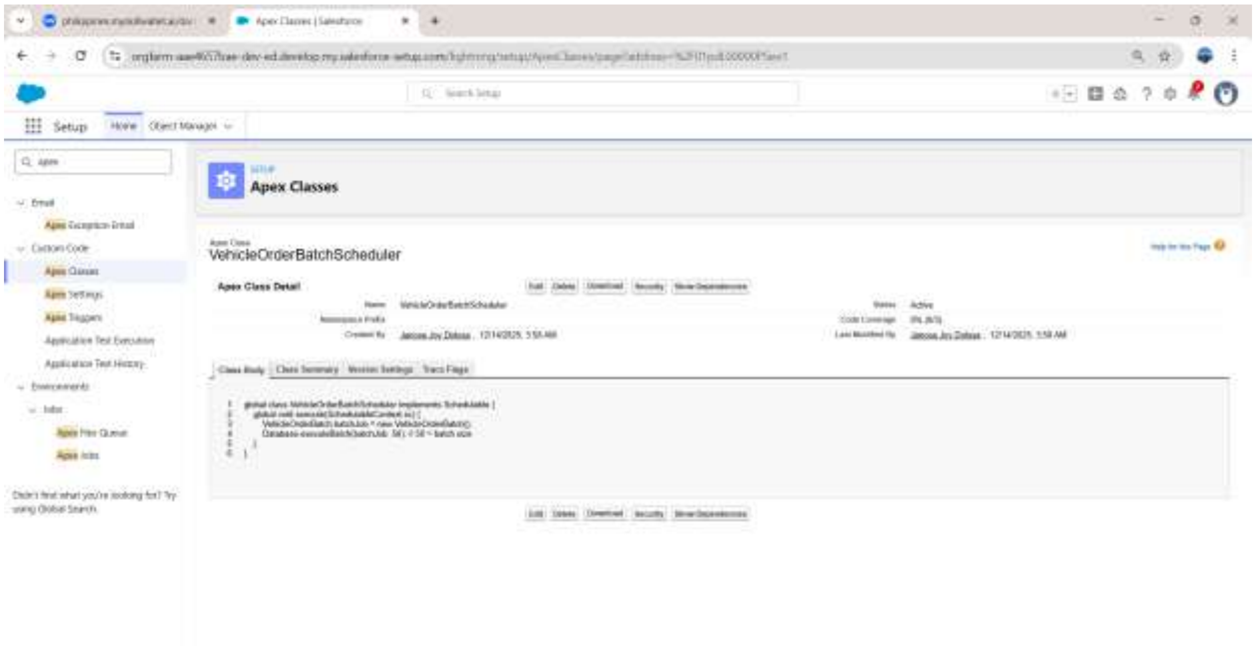


Apex Class
VehicleOrderTriggerHandler

Apex Class Detail

Name: VehicleOrderTriggerHandler
Namespace Prefix:
Created By: Jancea Joy Dalisay, 12/14/2025, 3:55 AM
Status: Active
Code Coverage: 0% (0/53)
Last Modified By: Jancea Joy Dalisay, 12/14/2025, 3:56 AM

```
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) {
```


Apex Classes

Apex Class
VehicleOrderBatchScheduler

Apex Class Detail

Name: VehicleOrderBatchScheduler
Status: Active
Namespace Prefix:
Code Coverage: 0% (0/3)
Created By: Jancee Joy Delosa, 12/14/2025, 3:58 AM
Last Modified By: Jancee Joy Delosa, 12/14/2025, 3:58 AM

```
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 & Customization

- Lightning App Setup: A custom Lightning App is set up for WhatNext Vision Motors employees to manage customers, vehicle inventory, and orders.
- Page Layouts: Customer and Order page layouts are customized to highlight key information like customer address and order status.
- Lightning Pages: Custom Lightning Pages are created to present data clearly and intuitively.

User Management

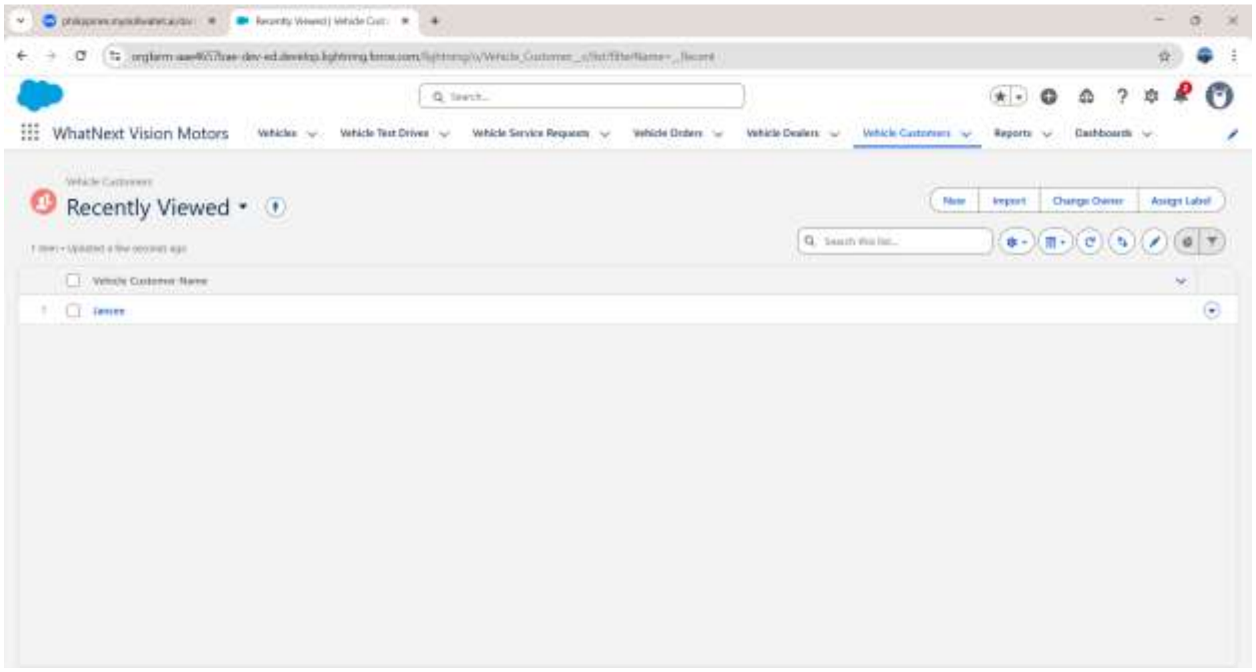
- Application Visibility: The "Vision Motors Order Management" app was restricted via Permission Sets to only be visible to internal Sales and Fulfillment staff, maintaining a clean App Launcher for other users.
- Record Page Assignment: Different versions of the Lightning Record Pages were assigned to the various profiles to ensure each user sees only the actions and related lists necessary for their specific role.

Reports and Dashboards

- Summarizes all orders currently stuck in 'Pending' status, grouped by the Suggested Dealer to identify fulfillment restrictions.

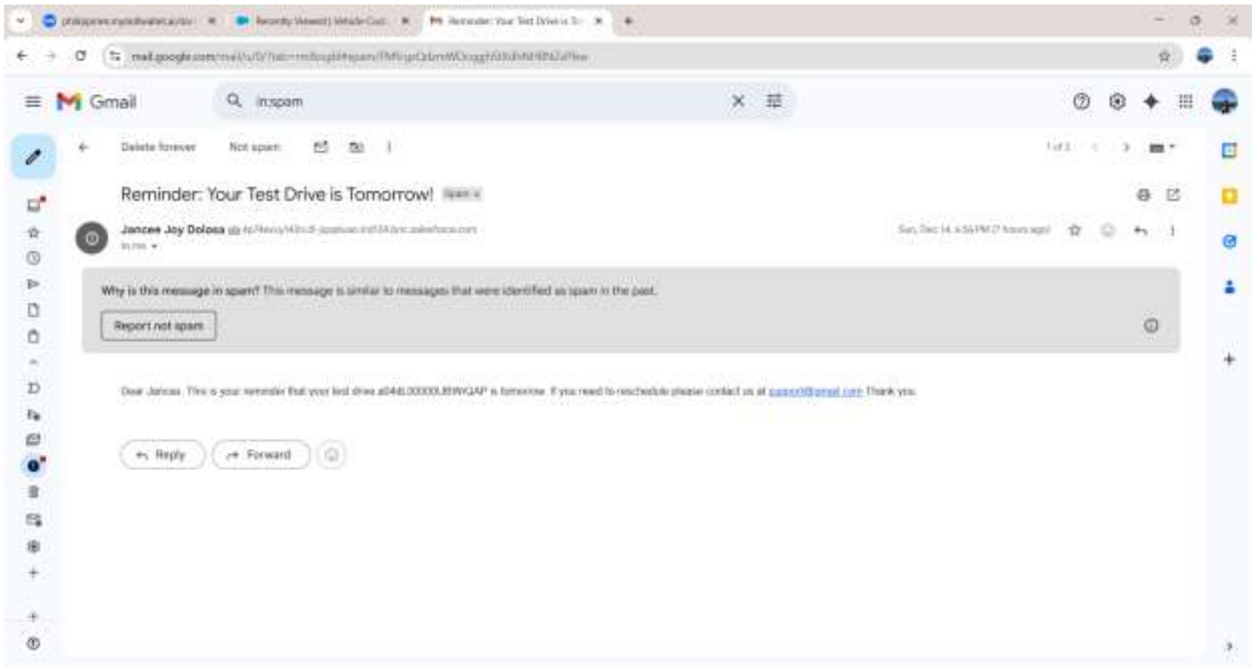
LWC Development

- A custom Lightning Web Component (LWC) was developed to enhance the dealer assignment process visually.



Phase 4: Data Migration, Testing & Security

- Data Loading Process: Initial vehicle inventory and customer records will be loaded using the Data Import Wizard or Data Loader.
- Field History Tracking: Field History Tracking is enabled on the Order Status field to maintain an audit trail of order fulfillment progress.
- Profiles, Roles, and Permission Sets: Profiles and Permission Sets are configured to ensure that only authorized employees can create orders and view sensitive customer data.
- Test Cases must be prepared and submitted for all features, including the stock validation rule and the scheduled flow's status update logic.
- Testing Approach: The automated flow logic (Pending/Confirmed status update) was tested using sample data and debug logs to ensure accurate status changes based on mocked stock availability.


Reminder: Your Test Drive is Tomorrow!

Jancee Joy Dolosa

Why is this message in spam? This message is similar to messages that were identified as spam in the past.

Dear Jancee, This is your reminder that your test drive a04dL00000JBWGAP is tomorrow. If you need to reschedule please contact us at support@gmail.com Thank you.

Phase 5: Deployment, Documentation & Maintenance

Deployment Strategy

- The primary method for migrating the solution will utilize a hybrid approach combining Change Sets and the Salesforce Command Line Interface (CLI).
- Change Sets: Used for migrating declarative components such as Custom Objects, Fields, Validation Rules, Page Layouts, Profiles, and Flows.
- Salesforce CLI: Preferred for deploying programmatic metadata, including all Apex Classes, Apex Triggers, Batch Apex, and their corresponding Test Classes, as the CLI offers greater control and easier script-based automation.

System Maintenance and Monitoring

- Effective maintenance and monitoring are essential to ensure the long-term health and performance of the CRM solution.
- Flow and Trigger Monitoring: Any runtime errors related to the Record-Triggered Flow (Test Drive Reminders) or the Apex Triggers will be captured and reviewed using the standard Debug Logs.

Documentation of Troubleshooting Approach

- Issue Triage: All reported issues will be logged and categorized based on severity and affected component (Order, Stock, Test Drive).

- Declarative Review: For issues related to orders or field updates, the first step involves reviewing Validation Rules (e.g., if an order is improperly blocked) and Flow Debug Logs (e.g., if a reminder email failed to send).
- Audit Trail Check: For issues involving unintended data changes, the Field History Tracking audit trail on the Order__c and Vehicle__c objects will be analyzed to determine the sequence of events and the user responsible for the change.
- Programmatic Debugging: For issues related to stock validation or dealer assignment, the developer team will reproduce the error in a sandbox environment and utilize the Apex Debug Logs to trace the execution path within the OrderTriggerHandler class and the Batch Apex job.
- Data Correction: If a data error is confirmed, the Data Loader will be used for safe, bulk data correction in non-critical situations, following strict change management protocols.

Conclusion

The WhatNext Vision Motors CRM capstone project successfully delivered a foundational Salesforce solution, fully achieving all objectives for optimizing the customer ordering experience and maximizing internal process efficiency. The system ensures superior Order Integrity by using both declarative and programmatic validation to prevent the placement of out-of-stock orders, while enhancing Customer Convenience through the automated assignment of orders to the nearest dealer via advanced Apex logic. Operational Efficiency is significantly improved by the deployment of Batch Apex jobs, which reliably handle bulk stock updates and scheduled notifications for the Fulfillment Team. Built on a scalable architecture utilizing best practices like trigger handlers and a dedicated Lightning App, the CRM is stable, maintainable, and ready for future integrations. This project successfully demonstrates the application of comprehensive Salesforce development skills to deliver substantial and measurable business value.