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

Abstract

WhatNext Vision Motors, a leader in the automotive industry, embarked on a Salesforce-based digital transformation project to streamline vehicle ordering, test drives, and service management.

The project focuses on improving customer experience, ensuring accurate stock availability, and automating core business processes using Salesforce's CRM capabilities.

The report outlines the structure, objects, relationships, and automation mechanisms implemented to achieve these objectives, along with real-world applicability.

Objective

The primary objectives of this project are:

- To enhance the vehicle ordering experience by suggesting the nearest dealer automatically.
- To prevent customers from placing orders for out-of-stock vehicles.
- To manage test drive bookings and service requests effectively.
- To automate the order status update process.
- To create a centralized, scalable, and efficient CRM system using Salesforce.

Technology Description

The project uses Salesforce CRM as the core platform, leveraging:

- Custom Objects for managing vehicles, customers, dealers, orders, test drives, and services.
- Automation Tools like Workflow Rules, Process Builder, and Apex Triggers.
- Batch Apex Classes to update bulk order records.
- Geolocation-based logic for dealer suggestion.
- App Manager and Tabs for streamlined user navigation.
- Validation Rules to enforce business logic and data quality.

Detailed Execution of Project Phases

Phase 1: Environment Setup

1. **Create Salesforce Developer Account**

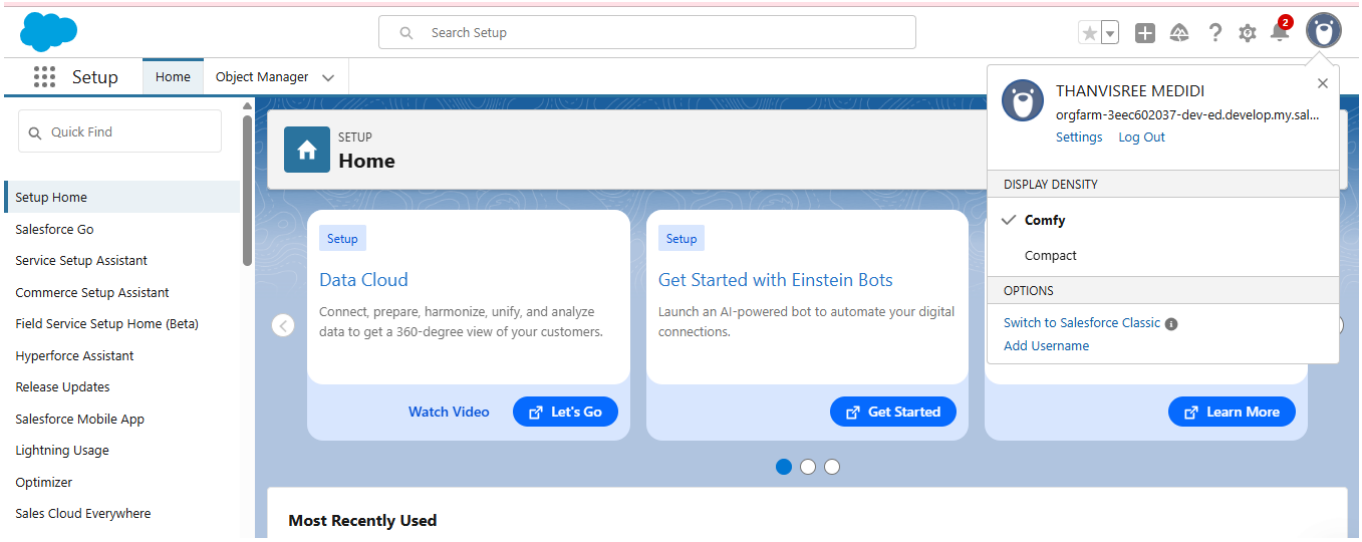
   Sign up at https://developer.salesforce.com for full platform access.

2. **Generate Salesforce Credentials**

   Admin and developer logins were created for secure access.

3. **Set Up Sandbox**

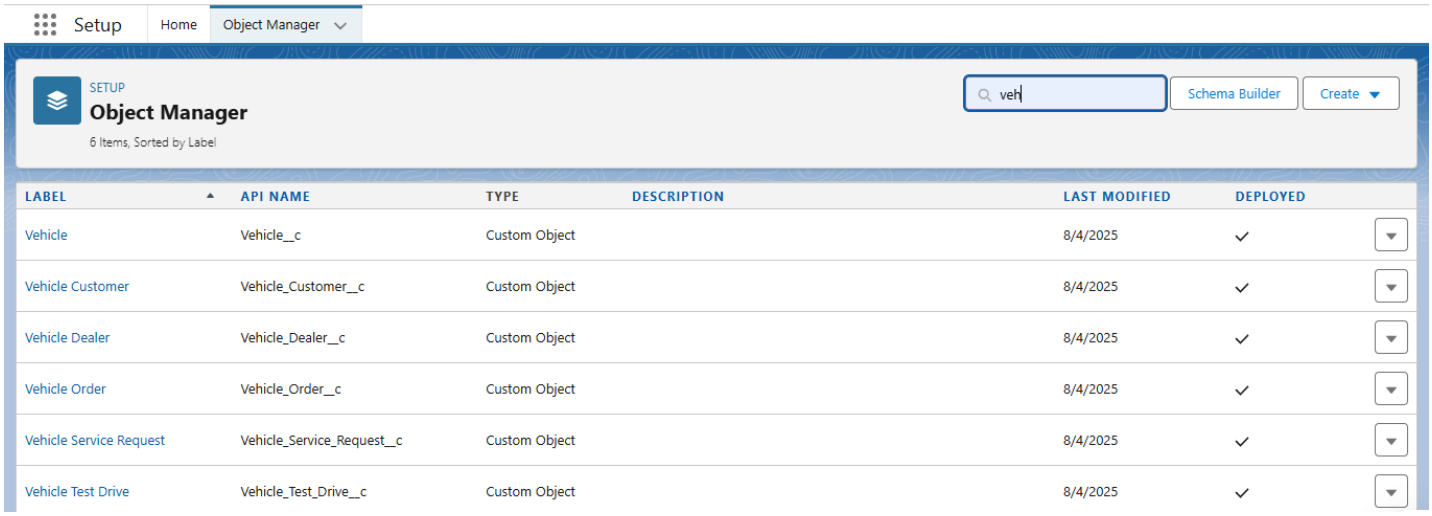
   A sandbox environment was prepared to safely test and develop features before moving to production.



Phase 2: Data Modelling

4. **Identify Core Entities** – The following objects were identified:
    - Vehicles
    - Dealers
    - Customers
    - Orders
    - Test Drives
    - Service Requests
5. **Create Custom Objects** :-

- From the setup page → Click on Object Manager → Click on Create → Click on Custom Object.



6. **Define Relationships** :-

   Lookup relationships were established:

   - Orders linked to Vehicles and Customers
   - Test Drives linked to Customers and Vehicles
   - Dealers linked to Vehicles

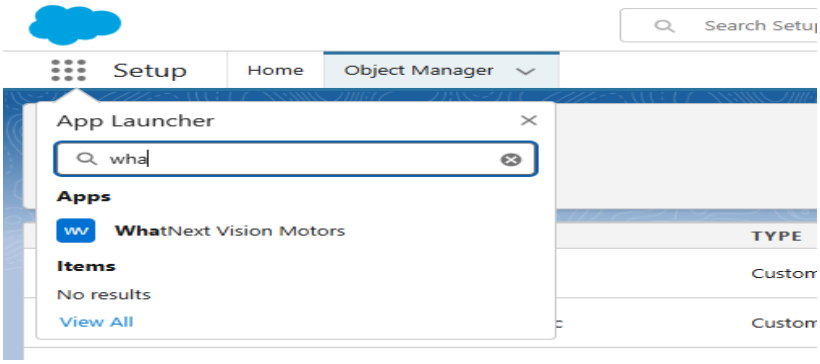
Phase 3: Fields & Relationships

7. **Add Standard and Custom Fields** – Each object was enriched with relevant fields such as:
   - Status, Stock_Quantity, Preferred_Vehicle_Type, etc.
8. **Apply Validation Rules** – To ensure data quality (e.g., stock must be >0 to place an order).
9. **Set Picklist Values** – For fields like Vehicle Model (SUV, Sedan, EV) and Status (Pending, Confirmed).

Phase 4: App Creation & User Interface

- To create a lightning app page:
  Go to setup page → search “app manager” in quick find → select “app manager” → click on New lightning App.

10. **Create App in App Manager** – App named **" WhatNext Vision Motors"** was built.
11. **Add Tabs** – Custom tabs for all objects (Vehicles, Orders, Test Drives, etc.) were added for easier navigation.
12. **Customise Page Layouts – Page layouts were designed for a** better user experience based on roles (Sales, Service, Admin).



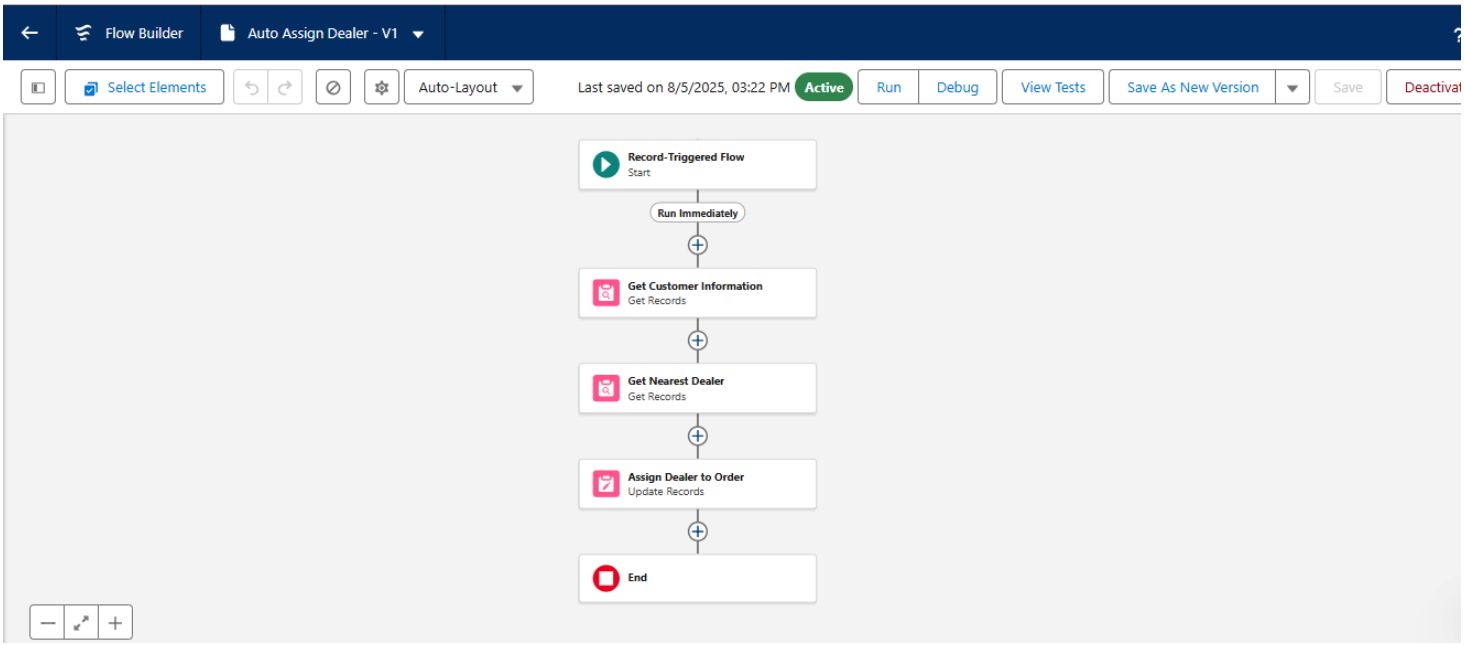
Phase 5: Automation

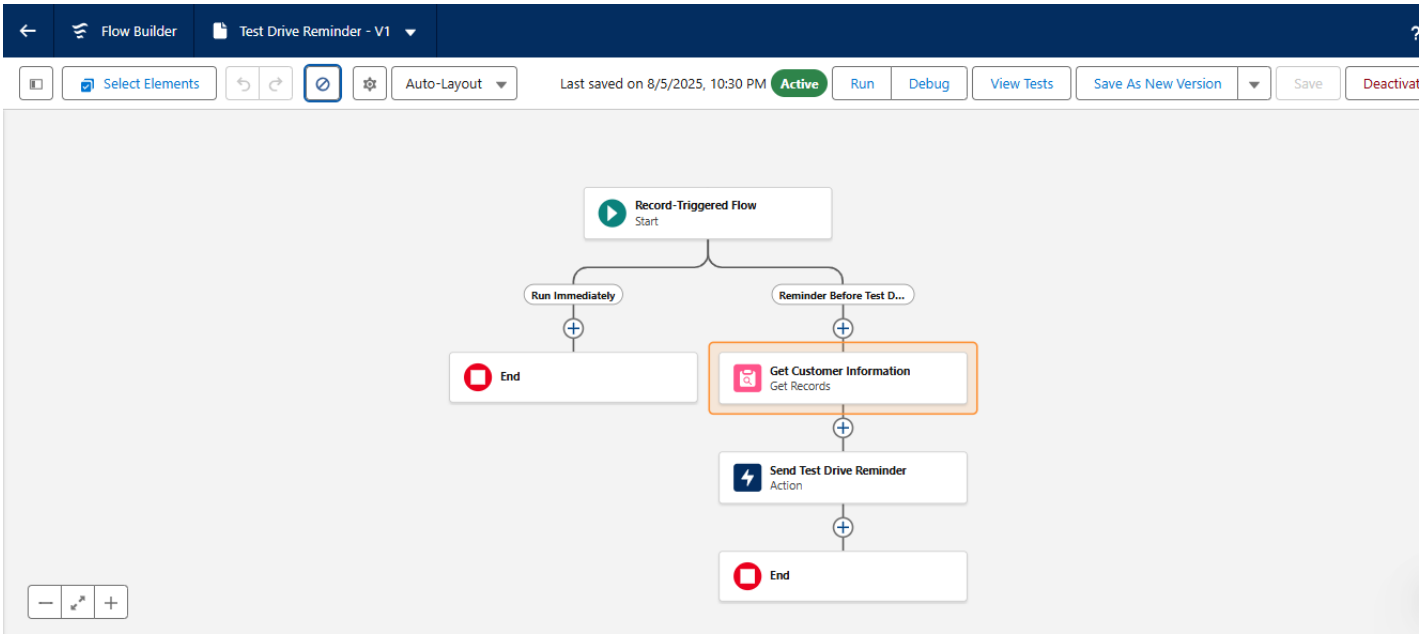
13. **Auto-Suggest Nearest Dealer** – Implemented using Apex and geolocation logic.
14. **Block Out-of-Stock Orders** – Apex Trigger ensures orders are placed only for available vehicles.
15. **Schedule Order Status Update** – Batch Apex class runs daily to:
    - Update order status to **"Confirmed"** if stock is available.
    - Update status to **"Pending"** if stock is zero.

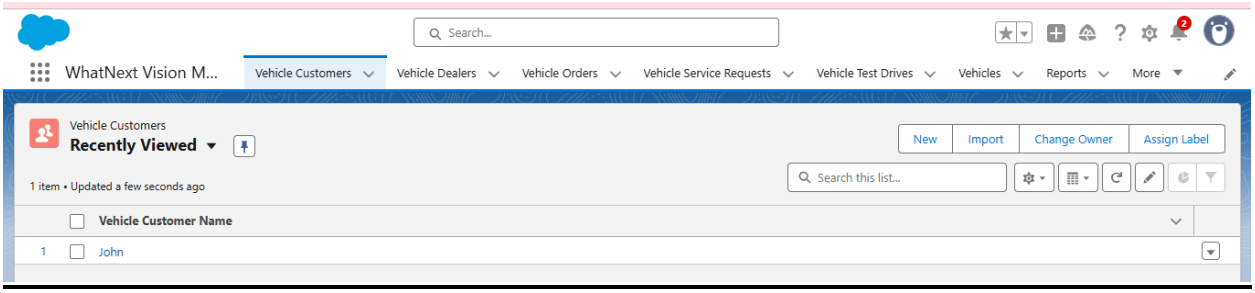
Phase 6: Testing and Deployment

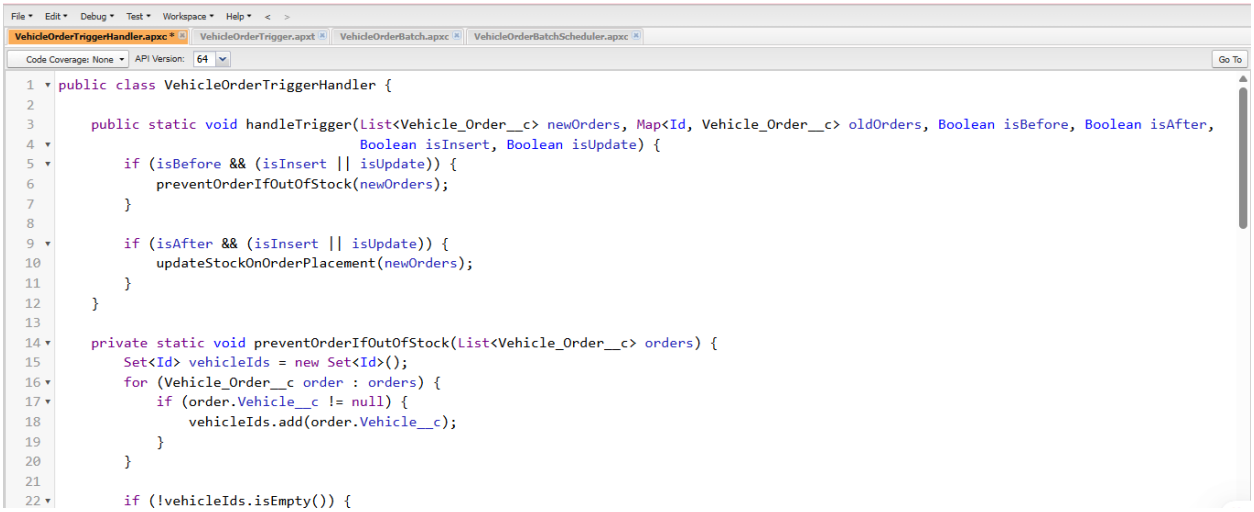
16. **Unit Testing** – Apex classes and triggers were tested with multiple test records.
17. **User Testing** – Simulated real-life scenarios (e.g., ordering, booking test drives).
18. **Bug Fixes & Improvements** – Based on feedback from stakeholders and testers.
19. **Deployment to Production** – Once validated, changes were deployed from the sandbox to production.

Screenshots:-

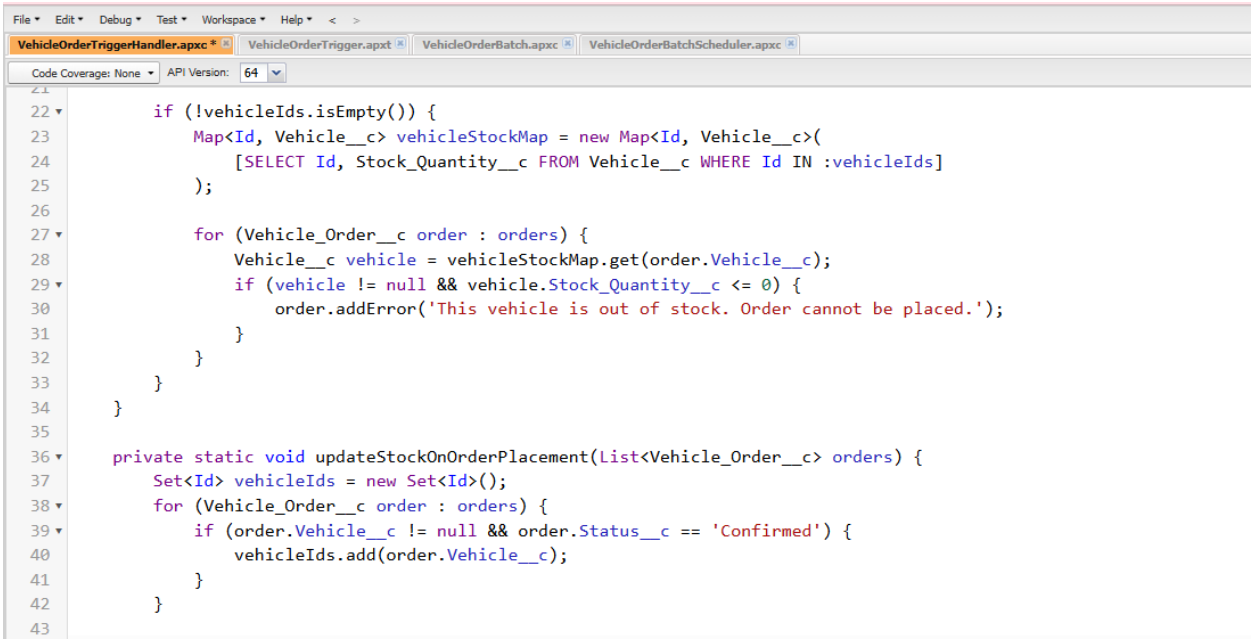




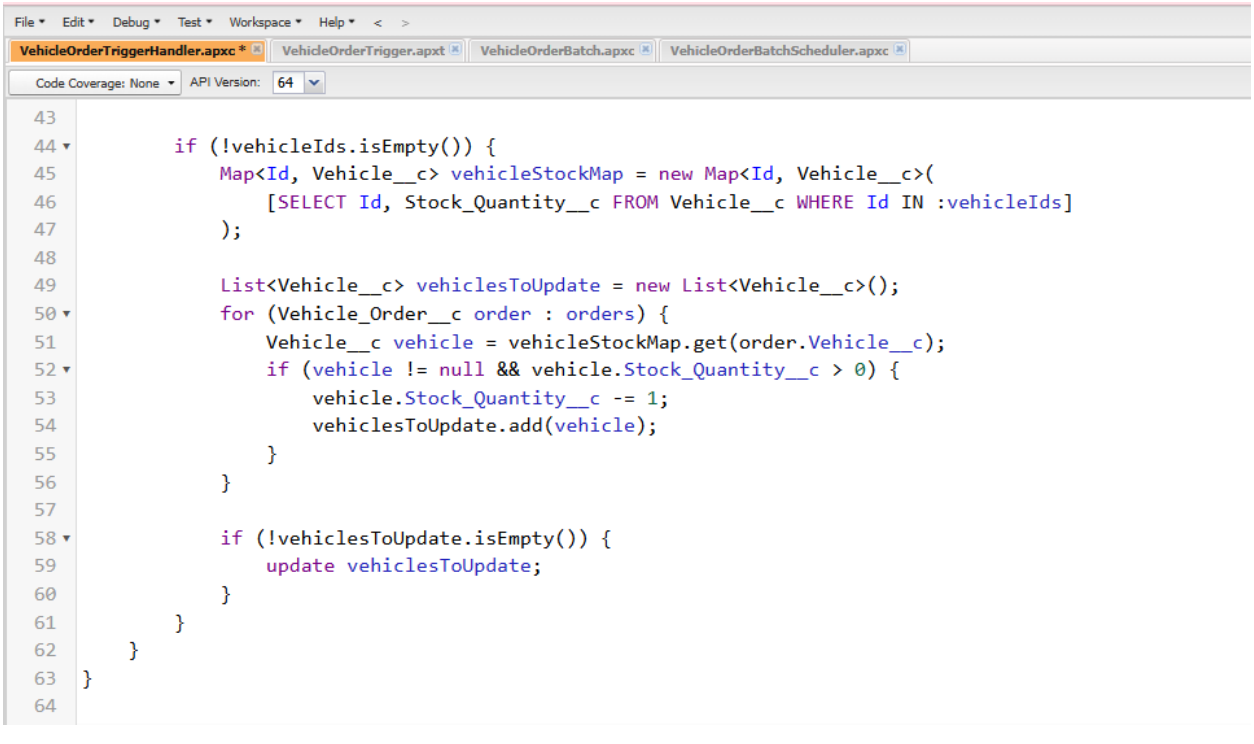




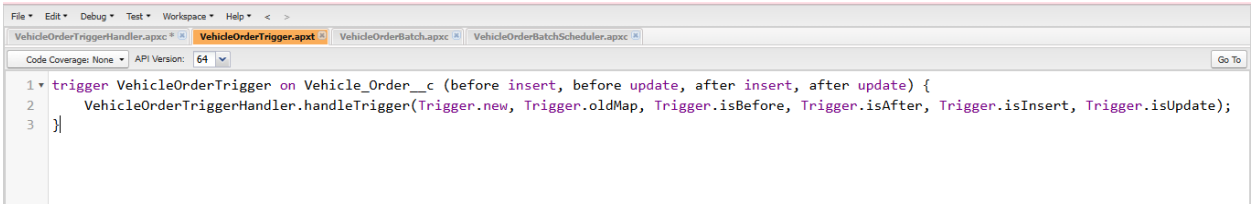
```
public class VehicleOrderTriggerHandler {

    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);
        }
    }

    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()) {
```



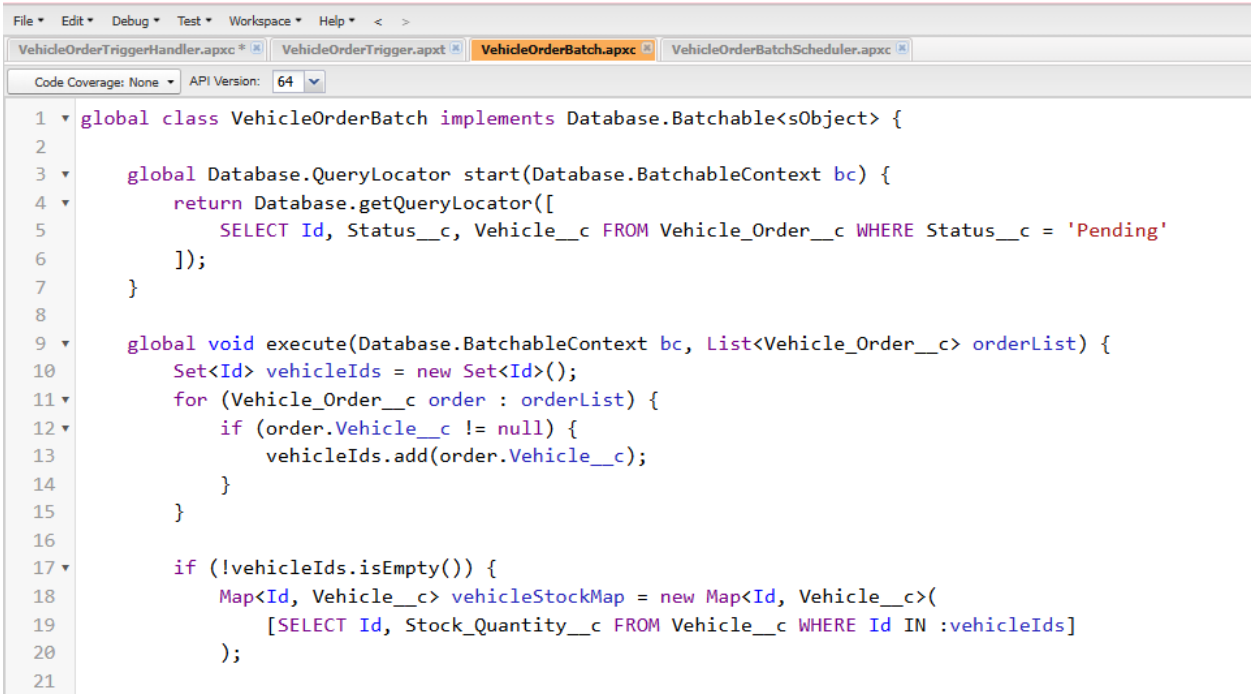
```
        if (!vehicleIds.isEmpty()) {
            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
            );

            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.');
                }
            }
        }
    }

    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);
            }
        }
```



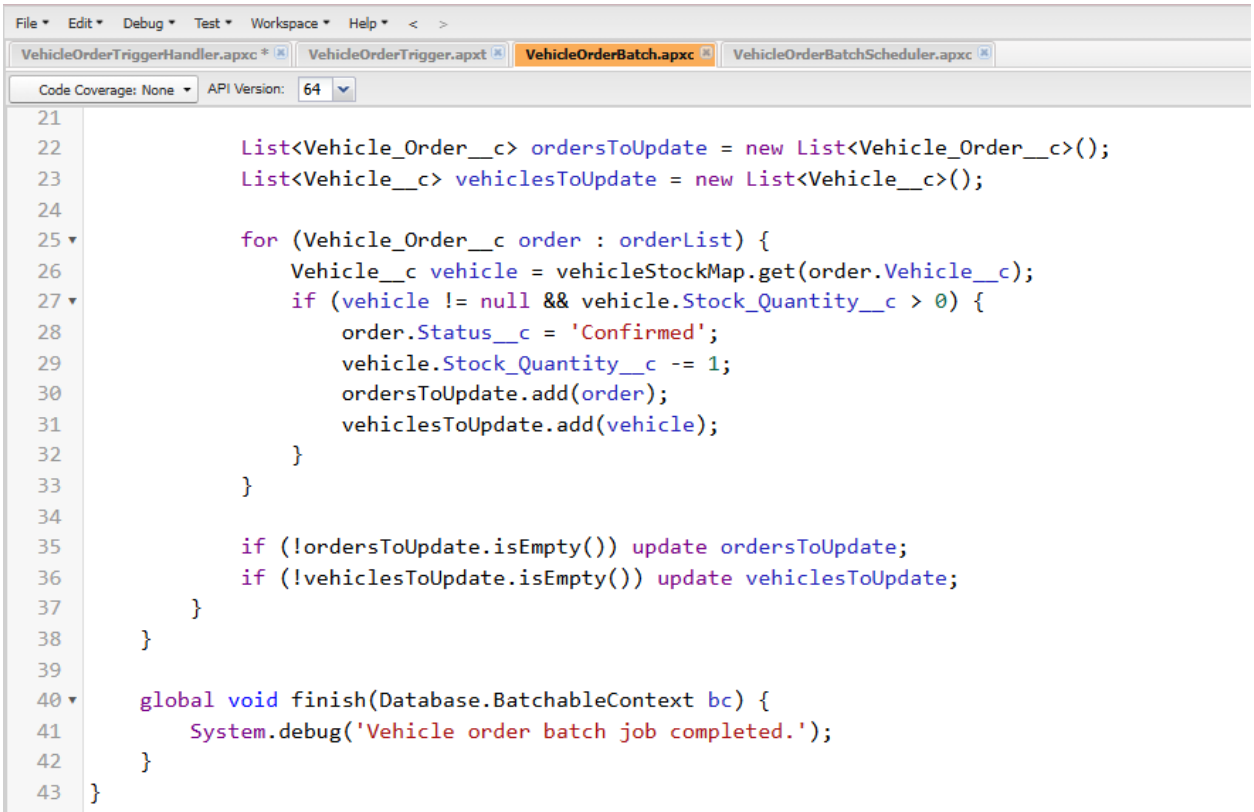
```
        if (!vehicleIds.isEmpty()) {
            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
            );

            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;
            }
        }
    }
}
```



```
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);
}
```




```apex
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]
            );

```



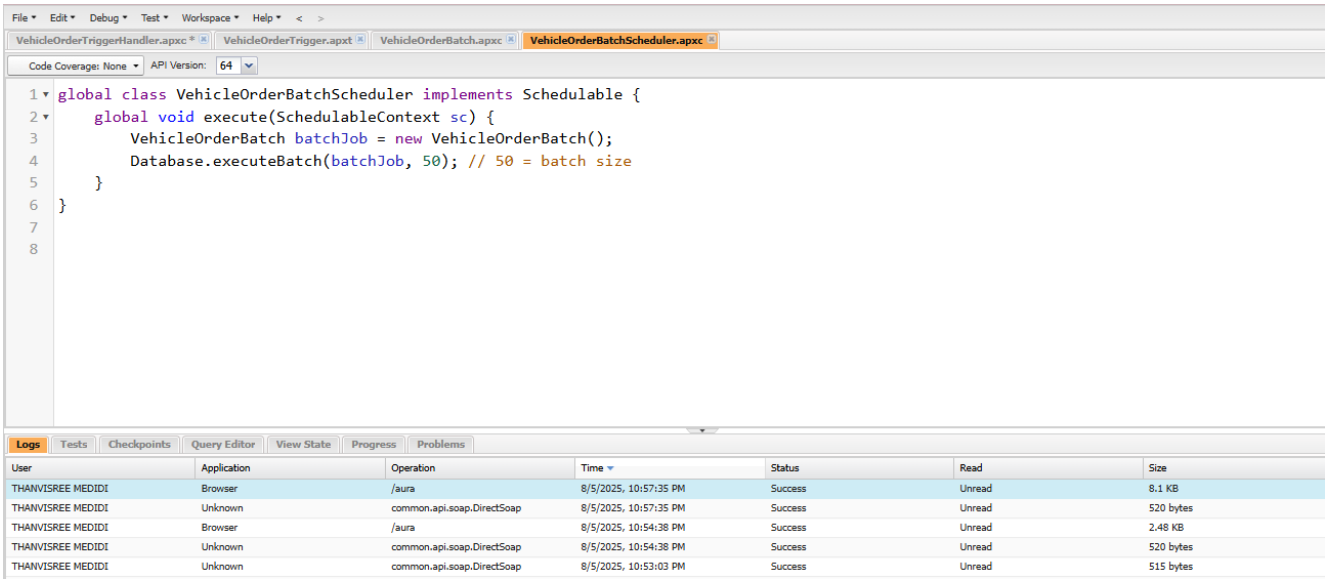

```apex

            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.');
    }
}
```

File Edit Debug Test Workspace Help < >

VehicleOrderTriggerHandler.apxc * | VehicleOrderTrigger.apxt | VehicleOrderBatch.apxc | VehicleOrderBatchScheduler.apxc

Code Coverage: None | API Version: 64

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



Project Explanation Using a Real-Life Example

A customer named Alia wants to purchase an SUV from WhatNext Vision Motors.

1. Alia visits the portal and enters her address.
2. The system automatically suggests the nearest authorised dealer.
3. She selects a model and places an order.
4. The system checks stock:
    - If available: the order status is immediately marked as **Confirmed**.

- If not: status is set to **Pending**.

5. Alia can also book a test drive for another model.
6. Post-purchase, if the vehicle needs servicing, she can raise a **Service Request** directly from the portal.
7. All data is tracked within Salesforce for better customer service and follow-ups.

Conclusion

The Salesforce project "WhatNext Vision Motors" successfully addresses key operational challenges in vehicle sales and service management. By leveraging Salesforce's powerful data modeling and automation capabilities, the company has:

- Improved the accuracy and reliability of the ordering process.
- Reduced the workload of sales and service teams.
- Enhanced the overall customer experience.

NAME	**THANVISREE MEDIDI**
STUDY	**B.TECH-CSE[CORE]-4TH YEAR**
COLLEGE	**GITAM DEEMED TO BE UNIVERSITY**