

Salesforce CRM Project Documentation

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

Project Overview

WhatNext Vision Motors. This project aims to develop an automotive platform that enhances convenience, safety, and efficiency for everyday use. Core functionalities include real-time vehicle monitoring, streamlined user interaction through a mobile interface, and optimized performance supported by data-driven decision-making. By incorporating emerging technologies and forward-looking design principles, Vision Motors aims to address current challenges in mobility—such as congestion, maintenance inefficiencies, and fragmented user experiences. This documentation outlines the project's concept, system architecture, key features, and potential impact, demonstrating how Vision Motors contributes to the future of accessible and smarter transportation.

Objectives

1. **Automate Order and Dealer.** Automatically assign the nearest dealer based on the customer's location at the time of order placement.
2. **Prevention of Out-of-Stock Orders.** Restrict placing orders for vehicles when out of stock
3. **Test Drive Reminders.** Schedule automated emails to remind customers about their upcoming test drives.

Detailed Execution of Project Phases

1. Developer Org Setup

Salesforce Developer Org was created using

<https://developer.salesforce.com/signup>

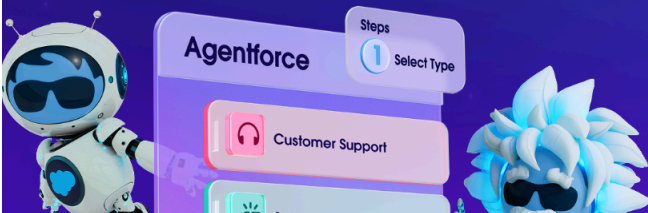
Account was verified, password reset, and granted access to the Salesforce Setup Page.

orce

Build enterprise-quality apps fast and get hands-on with Agentforce and Data Cloud.

Sign up for your Developer Edition.

- ✓ Build apps fast with drag-and-drop tools
- ✓ Go further with Apex code
- ✓ Build AI agents with Agentforce
- ✓ Harmonize your data with Data Cloud
- ✓ Ground Agentforce with structured and unstructured data
- ✓ Integrate with anything using APIs



Sign up for your Developer Edition

A free Salesforce Platform environment with Agentforce and Data Cloud

First name


Last name

Job title

Work email

Company

Country/Region

Philippines 

Your org may be provisioned on or migrated to Hyperforce, Salesforce's public cloud infrastructure.

☐ I agree to the [Main Services Agreement – Developer Services](#) and [Salesforce Program Agreement](#). I acknowledge, as described in the Developer Documentation: (1) the Developer Edition includes autonomous and other generative AI features; and (2) Salesforce may limit use of those features and the org, and may terminate any org that has been inactive for 45 days.

We value your privacy. To learn more, visit our [Privacy Statement](#).

☐ I'm not a robot




Figure 1. Sign up for Developer

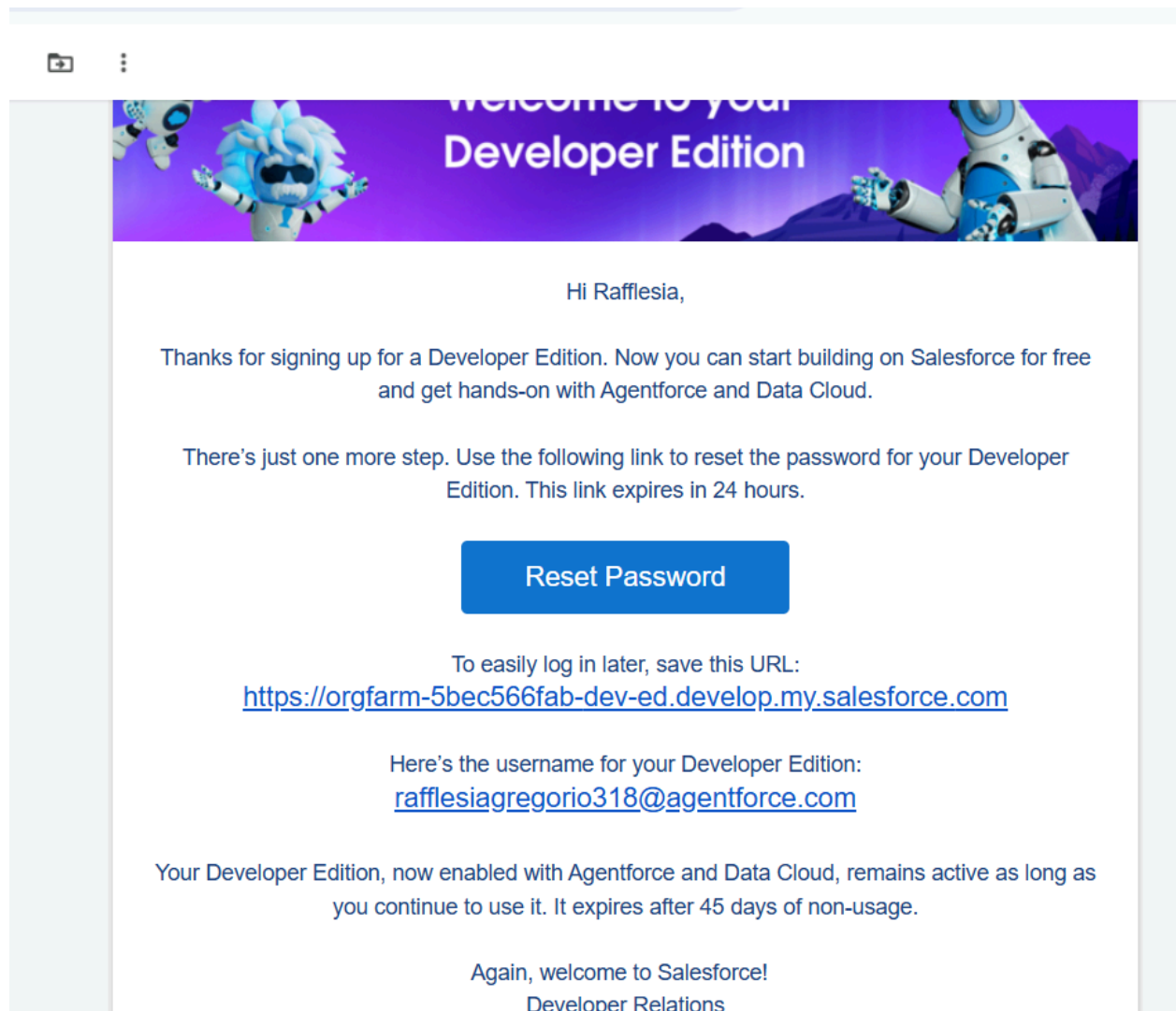


Figure 2. Password Reset

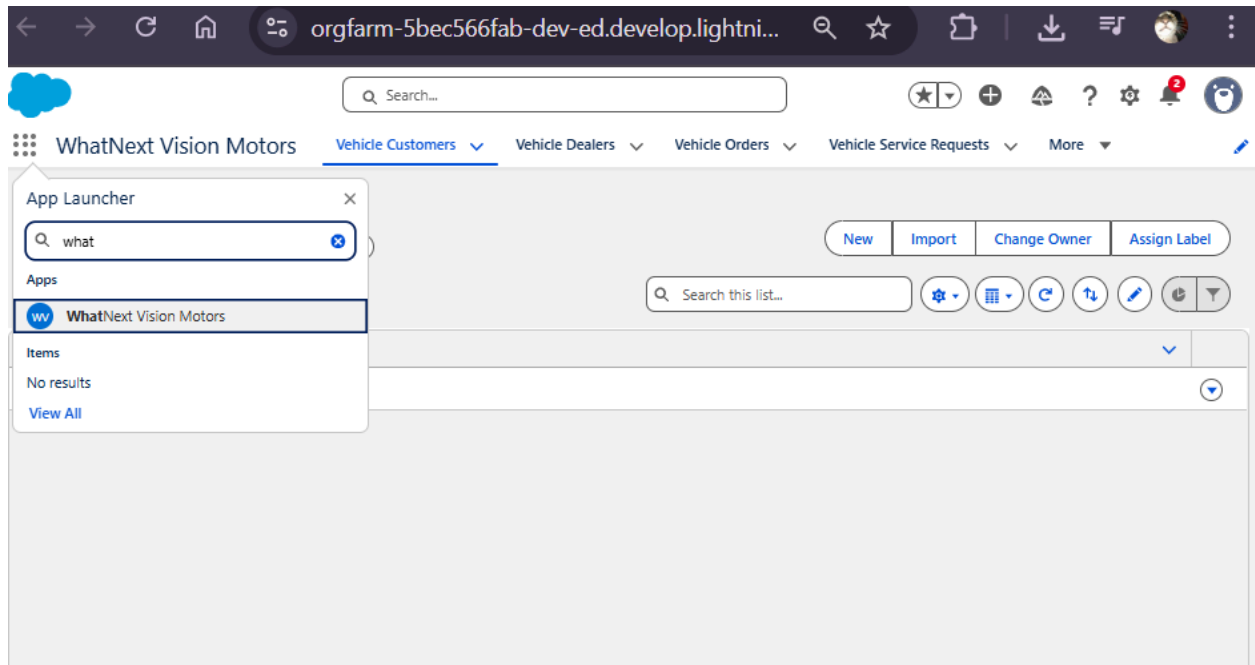



Figure 3. Creating Lightning App

A custom Lightning app named WhatNext Vision Motors
Included Tabs: Vehicle Customers, Vehicle Dealers, Vehicle Orders, Vehicles, Vehicle
Service Request, Vehicle Test Drivers, Reports, and Dashboard

2. Custom Object Creation

- Vehicle: Contains details about each vehicle and its availability.
- Vehicle Dealer: Holds information about dealerships.
- Vehicle Customer: Records customer profiles and contact information.
- Vehicle Order: Tracks and manages vehicle purchase orders.
- Vehicle Test Drive: Schedules and monitors test drive appointments.
- Vehicle Service Request: Logs and oversees vehicle maintenance and service history.

SETUP

Object Manager

6 Items, Sorted by Label

🔍 vehicle

[Schema Builder](#)

[Create](#) ▼

LABEL	▲ API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED	
Vehicle	Vehicle__c	Custom Object		11/21/2025	✓	▼
Vehicle Customer	Vehicle_Customer__c	Custom Object		11/21/2025	✓	▼
Vehicle Dealer	Vehicle_Dealer__c	Custom Object		11/21/2025	✓	▼
Vehicle Order	Vehicle_Order__c	Custom Object		11/21/2025	✓	▼
Vehicle Service Request	Vehicle_Service_Request__c	Custom Object		11/21/2025	✓	▼
Vehicle Test Drive	Vehicle_Test_Drive__c	Custom Object		11/21/2025	✓	▼

Figure 4. Custom Object Manager






Custom Object Tabs			<div> New What Is This? </div>
Action	Label	Tab Style	
Edit Del	Vehicle Customers	 <div>People</div>	
Edit Del	Vehicle Dealers	 <div>Building</div>	
Edit Del	Vehicle Orders	 <div>Box</div>	
Edit Del	Vehicles	 <div>Car</div>	
Edit Del	Vehicle Service Requests	 <div>Form</div>	
Edit Del	Vehicle Test Drives	 <div>Gears</div>	

Figure 5. Custom Object Tabs

The screenshot shows a web form titled "New Vehicle Order". At the top right, a legend indicates that an asterisk (*) denotes "Required Information". The form is organized into sections. The "Information" section is highlighted with a grey header. Below it, the "Vehicle Order Number" field is empty. The "Owner" field is populated with "Rafflesia Gregorio" and includes a user profile icon. The "Vehicle Customer" field contains "John" with a red person icon and a blue refresh button. The "Vehicle" field contains "Honda" with a red car icon and a blue refresh button. The "Order Date" field shows "11/29/2025" with a calendar icon. The "Status" field is set to "Pending" with a blue refresh button. The "Assigned Dealer" field has a placeholder "Search Vehicle Dealers...". A pink error message box is overlaid on the form, stating "We hit a snag." and "Review the errors on this page." with a bullet point: "This vehicle is out of stock. Order cannot be placed." At the bottom right, there are three buttons: a red "X" icon, a "Cancel" button, a "Save & New" button, and a blue "Save" button.

Order Number

New Vehicle Order

* = Required Information

Information

Vehicle Order Number

Owner
Rafflesia Gregorio

Vehicle Customer
John

Vehicle
Honda

Order Date
11/29/2025

Status
Pending

Assigned Dealer
Search Vehicle Dealers...

We hit a snag.

Review the errors on this page.

- This vehicle is out of stock. Order cannot be placed.

Cancel Save & New Save

Figure 6. Validation Rules

Prevents creating an order if the user wants to, and an error will appear, and won't proceed.

Flow 1: Auto-Assigned Dealer

The flow works on Vehicle_Order__c

Customer information will be fetched

Finds the nearest dealer to the customer and assigns it to the customer's order.

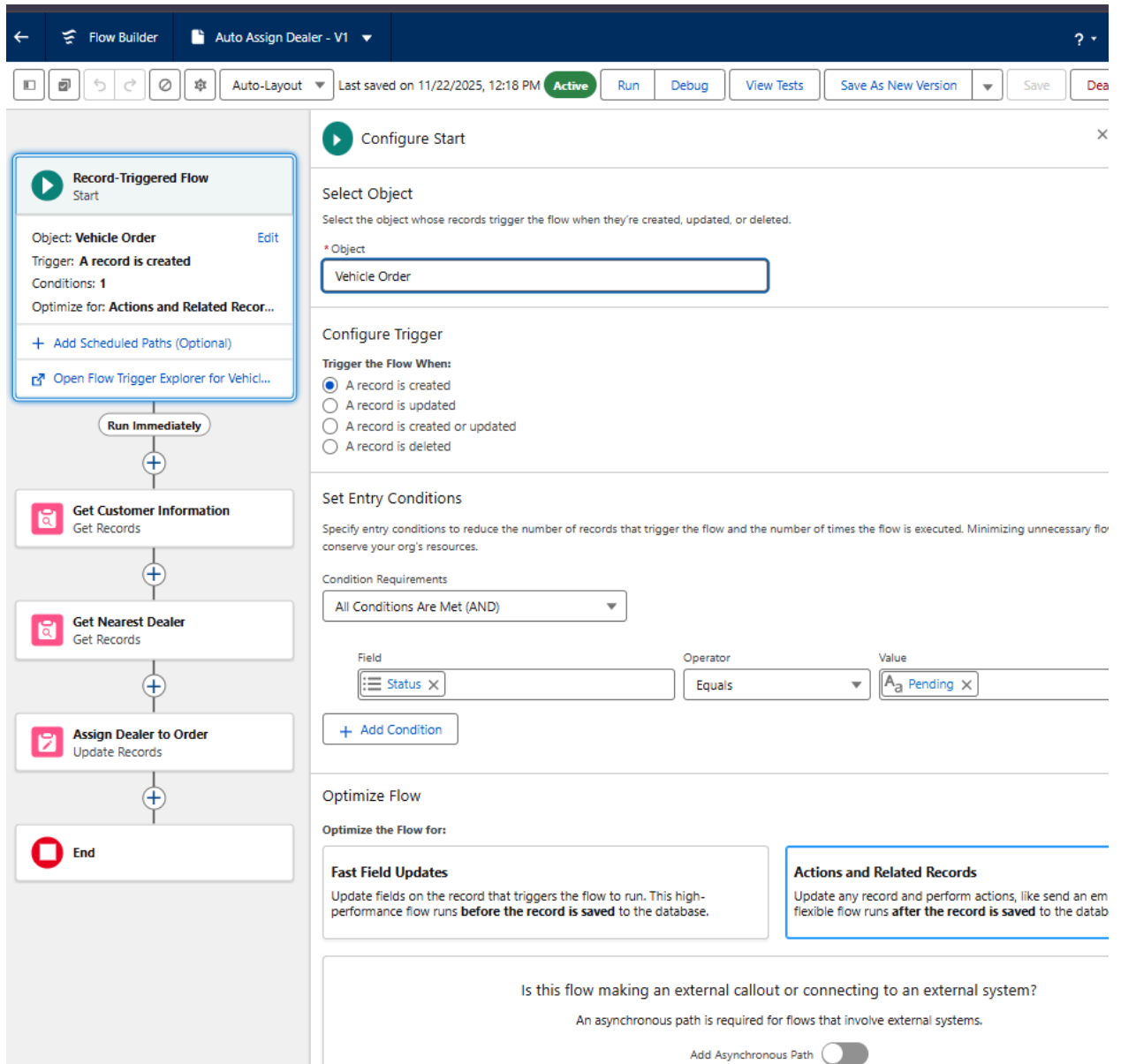


Figure 6. Auto Assign Dealer Flow

Flow 2: Test Drive Reminder

This flow works on Vehicle_Test_Drive__c

A reminder will be sent to the user for the upcoming test drive, and the email will show how successful it's implemented.

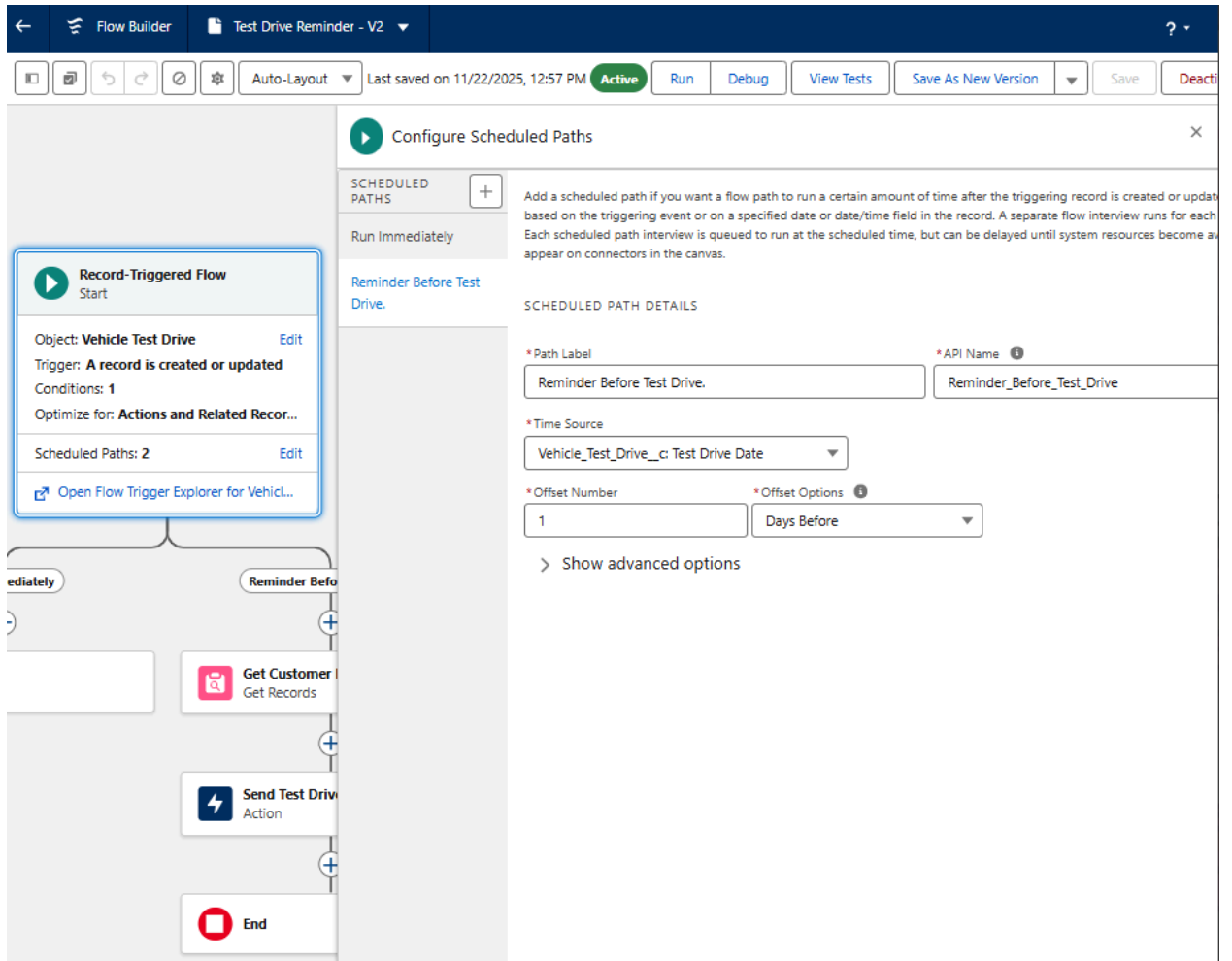


Figure 7. Test Drive Reminder Flow

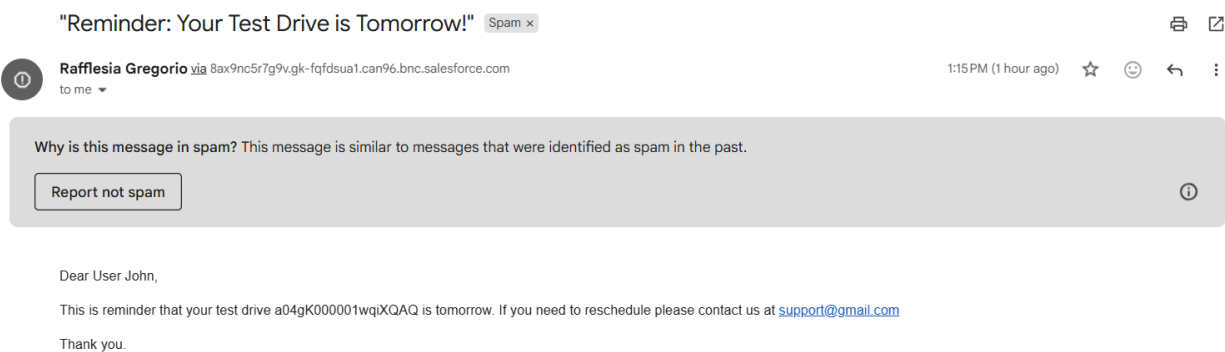
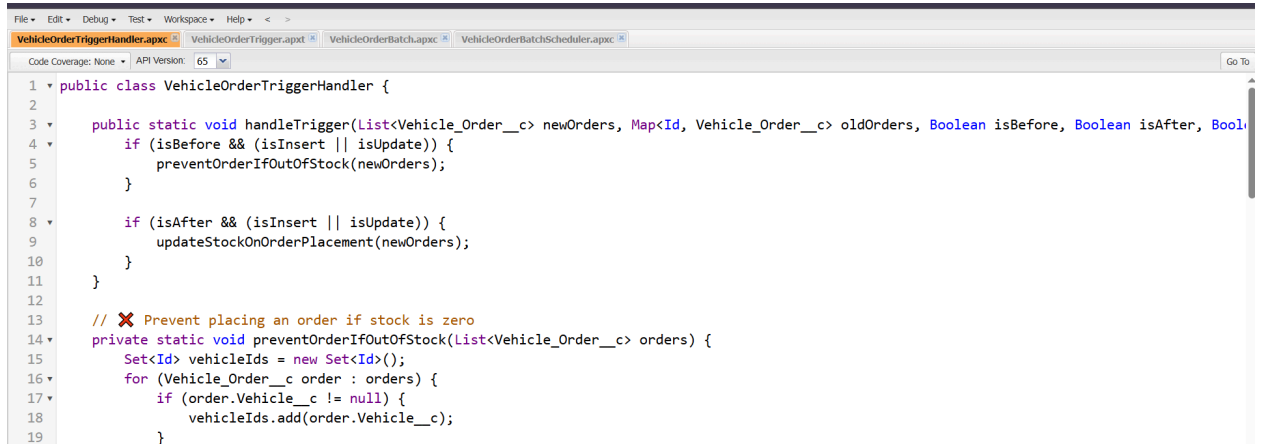


Figure 8. Email Sent for Reminder

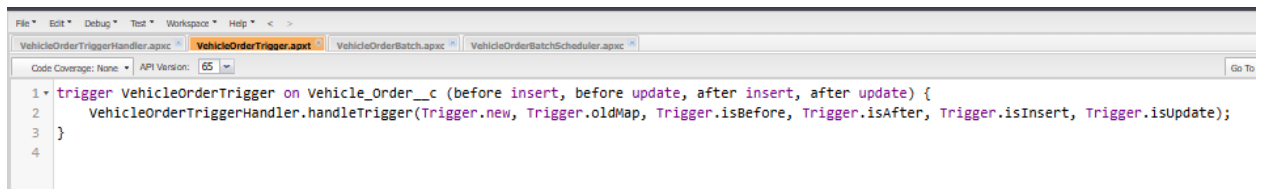
Apex Trigger and Handler

The figure shown is the handler or codes to prevent ordering and update the stock when the order is confirmed.



```
1 public class VehicleOrderTriggerHandler {
2
3     public static void handleTrigger(List<Vehicle_Order__c> newOrders, Map<Id, Vehicle_Order__c> oldOrders, Boolean isBefore, Boolean isAfter, Boolean isUpdate) {
4         if (isBefore && (isInsert || isUpdate)) {
5             preventOrderIfOutOfStock(newOrders);
6         }
7
8         if (isAfter && (isInsert || isUpdate)) {
9             updateStockOnOrderPlacement(newOrders);
10        }
11    }
12
13    // ✖ Prevent placing an order if stock is zero
14    private static void preventOrderIfOutOfStock(List<Vehicle_Order__c> orders) {
15        Set<Id> vehicleIds = new Set<Id>();
16        for (Vehicle_Order__c order : orders) {
17            if (order.Vehicle__c != null) {
18                vehicleIds.add(order.Vehicle__c);
19            }
20        }
21    }
22 }
```

Figure 9. Vehicle Order Trigger Handler



```
1 trigger VehicleOrderTrigger on Vehicle_Order__c (before insert, before update, after insert, after update) {
2     VehicleOrderTriggerHandler.handleTrigger(trigger.new, trigger.oldMap, trigger.isBefore, trigger.isAfter, trigger.isInsert, trigger.isUpdate);
3 }
4
```

Figure 10. Vehicle Order Trigger

```

1  global class VehicleOrderBatch implements Database.Batchable<Object> {
2
3      global Database.QueryLocator start(Database.BatchableContext bc) {
4          return Database.getQueryLocator([
5              SELECT Id, Status__c, Vehicle__c FROM Vehicle_Order__c WHERE Status__c = 'Pending'
6          ]);
7      }
8
9      global void execute(Database.BatchableContext bc, List<Vehicle_Order__c> orderList) {
10         Set<Id> vehicleIds = new Set<Id>();
11         for (Vehicle_Order__c order : orderList) {
12             if (order.Vehicle__c != null) {
13                 vehicleIds.add(order.Vehicle__c);
14             }
15         }
16
17         if (!vehicleIds.isEmpty()) {
18             Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>([
19                 SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds
20             ]);
21
22             List<Vehicle_Order__c> ordersToUpdate = new List<Vehicle_Order__c>();
23             List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();
24
25             for (Vehicle_Order__c order : orderList) {
26                 Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
27                 if (vehicle != null && vehicle.Stock_Quantity__c > 0) {
28                     order.Status__c = 'Confirmed';
29                     vehicle.Stock_Quantity__c -= 1;
30                     ordersToUpdate.add(order);
31                     vehiclesToUpdate.add(vehicle);
32                 }
33             }
34
35             if (!ordersToUpdate.isEmpty()) update ordersToUpdate;
36             if (!vehiclesToUpdate.isEmpty()) update vehiclesToUpdate;
37         }
38     }
39
40     global void finish(Database.BatchableContext bc) {
41         System.debug('Vehicle order batch job completed.');

```

Figure 11. Vehicle Order Batch

```
File Edit Debug Test Workspace Help < >
VehicleOrderTriggerHandler.apxc VehicleOrderTrigger.apxt VehicleOrderBatch.apxc VehicleOrderBatchScheduler.apxc
Code Coverage: None API Version: 65
1 global class VehicleOrderBatchScheduler implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         VehicleOrderBatch batchJob = new VehicleOrderBatch();
4         Database.executeBatch(batchJob, 50); // 50 = batch size
5     }
6 }
```

Figure 12. Vehicle Order Batch Scheduler

Real World Example

The following figure is a walkthrough on how to make it organized.

Edit John

* = Required Information

*Vehicle Customer Name

John

Email

rafflesiagregorio@gmail.com

Phone

1234567890

Address

Calumpit

Preferred Vehicle Type

Sedan

Owner

Rafflesia Gregorio

Created By

Rafflesia Gregorio, 11/21/2025, 8:08 PM

Last Modified By

Rafflesia Gregorio, 11/21/2025, 8:08 PM

Cancel

Save & New

Save

Figure 13. Vehicle Customer

The figure shows the details about the user as well as the vehicle type.

Figure 15. Vehicle Orders
Tracks and manages vehicle purchase orders.

Vehicle
Honda

New

Related

Details

Vehicle Name

Honda

Vehicle Model

EV

Stock Quantity

99

Price

\$80,000

Vehicle Dealer

ww

Status

Available

Created By

Rafflesia Gregorio, 11/21/2025, 8:10 PM


Owner

Rafflesia Gregorio

Last Modified By

Rafflesia Gregorio, 11/21/2025, 11:12 PM

Figure 16. Vehicles
The vehicle order was placed above, and the stock quantity was reduced from 100 to 1 automatically.



Vehicle Test Drive

xyz

Related

Details

Vehicle Test Drive Name

xyz

Vehicle

Honda

Test Drive Date

11/22/2025


Status

Scheduled


Vehicle Customer

John

Created By

 Rafflesia Gregorio, 11/21/2025, 9:15 PM

Owner

 Rafflesia Gregorio

Last Modified By


 Rafflesia Gregorio, 11/21/2025, 9:15 PM

Figure 17. Vehicle Test Drive

The figure shows schedules and monitors test drive appointments, and the vehicle customers are also on the details.

Conclusion

The WhatNext Vision Motors system is built for business processes like vehicle handling, ordering, and application of assigned dealers near the customers, tracking of quantity, as it has been successfully implemented, and the automation of. Through automation, users can structure their orders with minimal errors and provide better service.