

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.

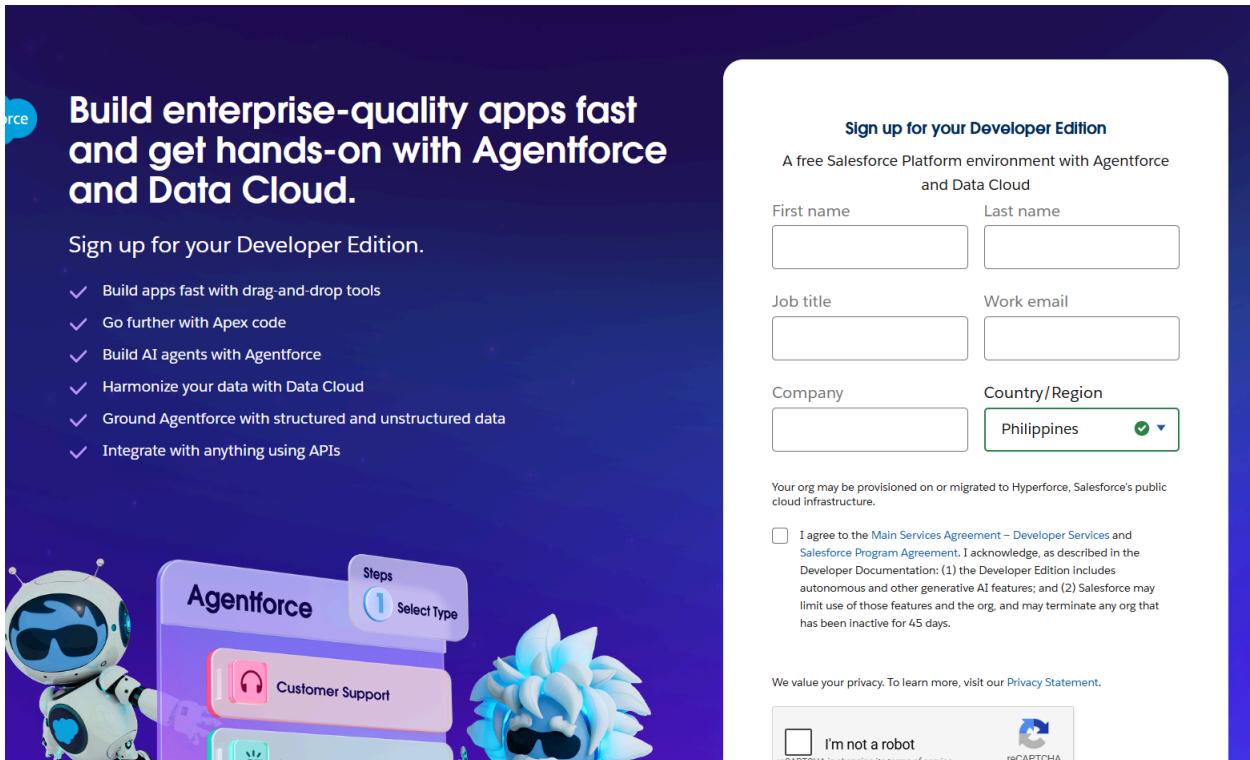


Figure 1. Sign up for Developer

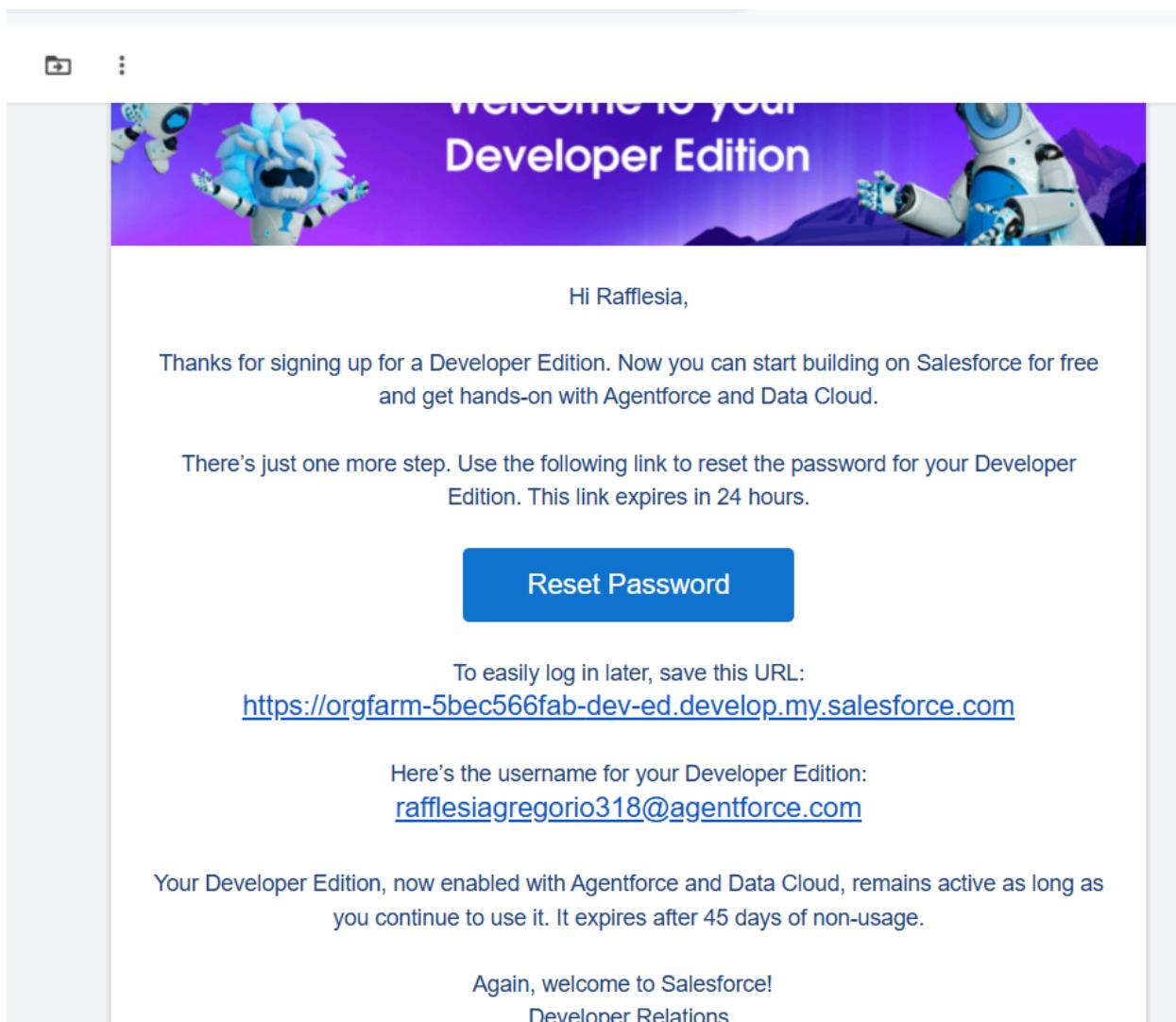


Figure 2. Password Reset

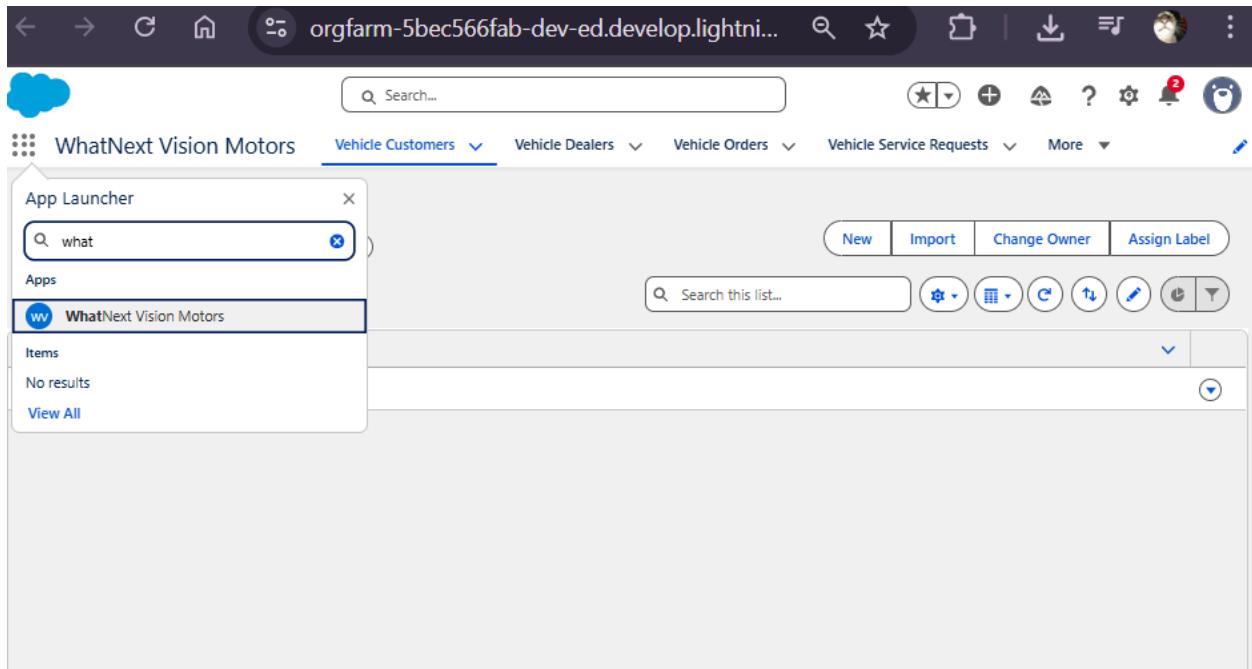


Figure 3. Creating Lightning App

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.

Object Manager					
6 Items. Sorted by Label					
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		New	What Is This?
Action	Label	Tab Style	
Edit Del	Vehicle Customers		People
Edit Del	Vehicle Dealers		Building
Edit Del	Vehicle Orders		Box
Edit Del	Vehicles		Car
Edit Del	Vehicle Service Requests		Form
Edit Del	Vehicle Test Drives		Gears

Figure 5. Custom Object Tabs

Order Number

New Vehicle Order

* = Required Information

Information

Vehicle Order Number	Owner
Vehicle Customer	Rafflesia Gregorio
Vehicle	John
Order Date	Honda
Status	11/29/2025
Assigned Dealer	Pending
	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.

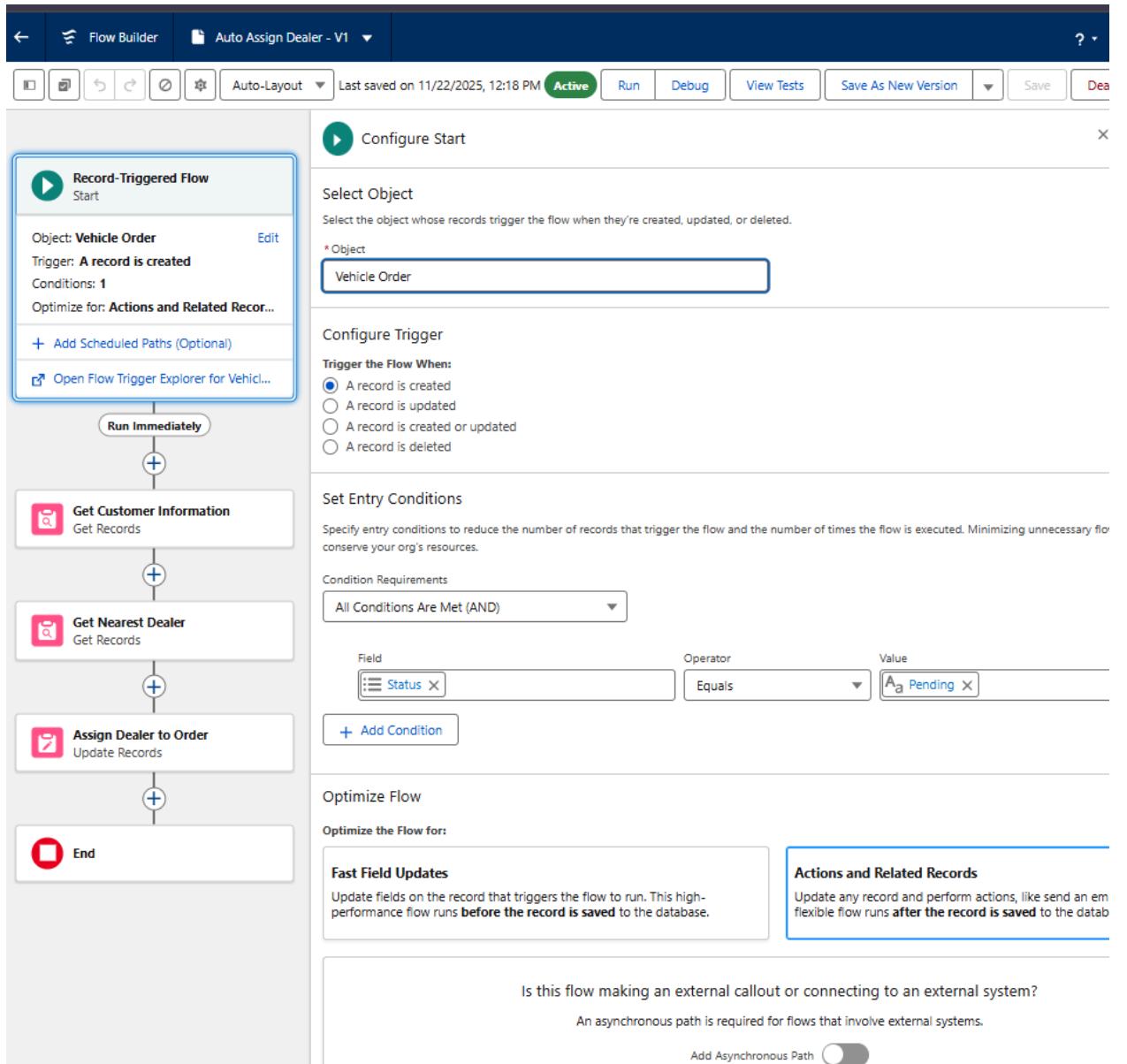


Figure 6. Auto Assign Dealer Flow

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.

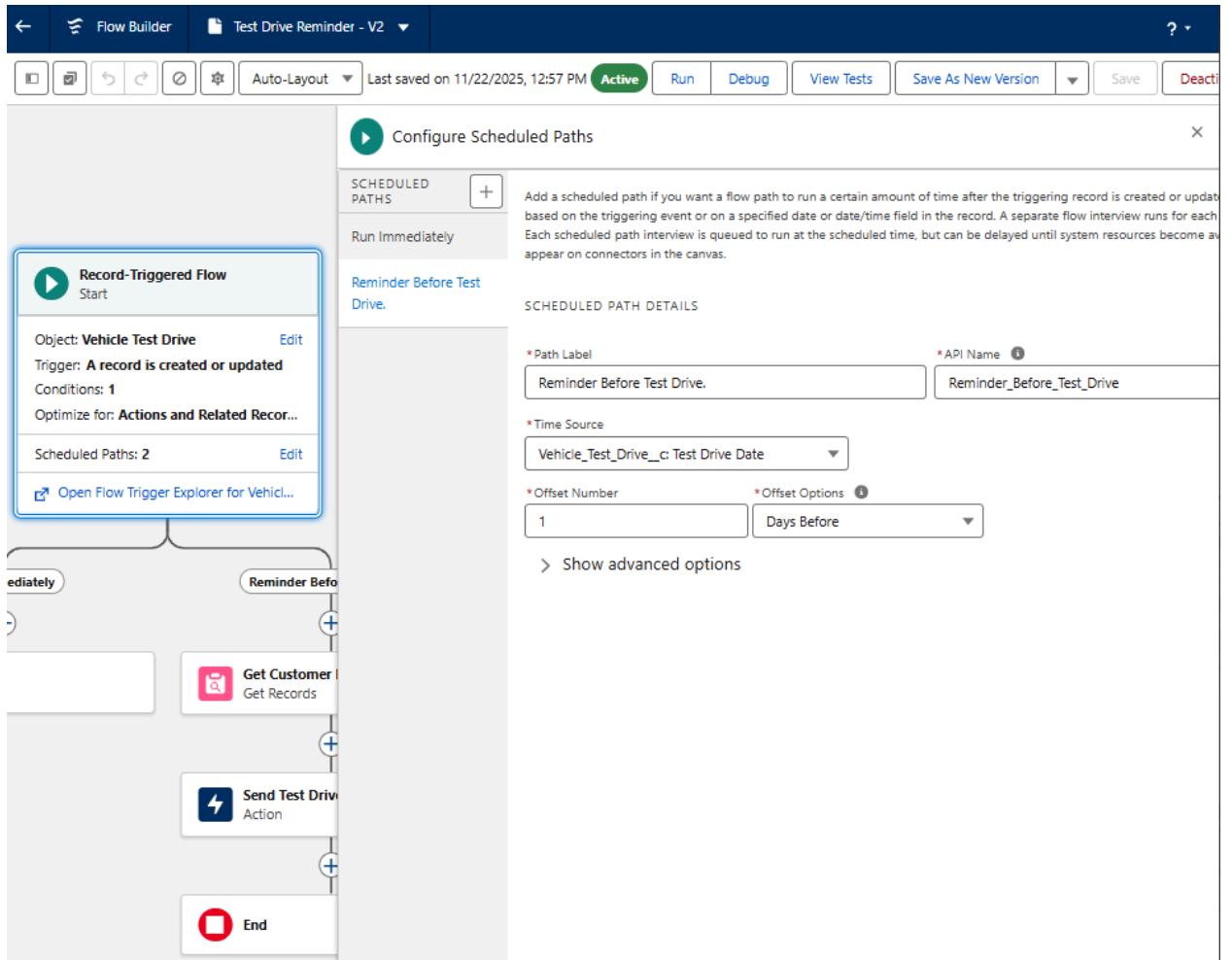


Figure 7. Test Drive Reminder Flow

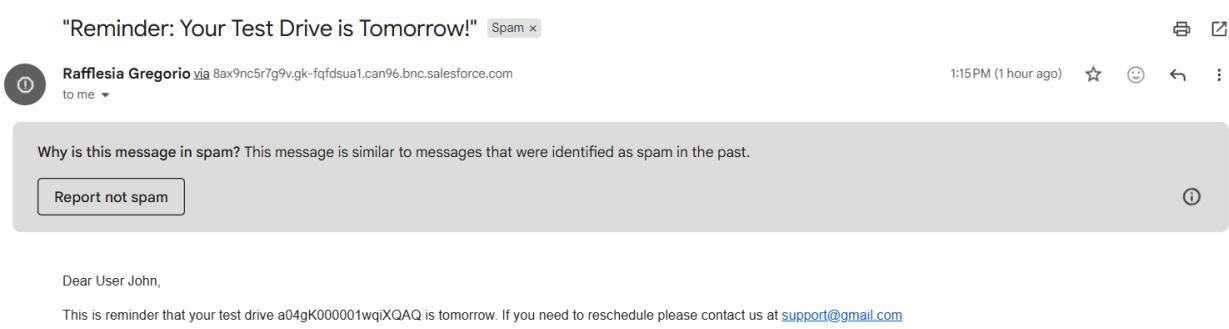


Figure 8. Email Sent for Reminder

Apex Trigger and Handler

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

```
File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
VehicleOrderTriggerHandler.apxc [ ] VehicleOrderTrigger.apxt [ ] VehicleOrderBatch.apxc [ ] VehicleOrderBatchScheduler.apxc [ ]
Code Coverage: None ▾ API Version: 65 ▾ Go To
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
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 }
```

Figure 9. Vehicle Order Trigger Handler

```
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);
}
```

Figure 10. Vehicle Order Trigger

The screenshot shows a Salesforce code editor window with the following details:

- File Menu:** File, Edit, Debug, Test, Workspace, Help.
- Tab Bar:** VehicleOrderTriggerHandler.apxc, VehicleOrderTrigger.apxt, **VehicleOrderBatch.apxc**, VehicleOrderBatchScheduler.apxc.
- Code Coverage:** None.
- API Version:** 65.

```
1 global class VehicleOrderBatch implements Database.Batchable<sObject> {
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.');
42    }
43 }
44 |
```

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.

Edit Raff

* = Required Information

*Vehicle Dealer Name <input type="text" value="Raff"/>	Owner Rafflesia Gregorio
Dealer Location <input type="text" value="Calumpit"/>	
Dealer Code DC-0001	
Phone <input type="text" value="1234567890"/>	
Email <input type="text" value="abc@gmail.com"/>	
Created By Rafflesia Gregorio, 11/21/2025, 8:08 PM	Last Modified By Rafflesia Gregorio, 11/21/2025, 8:08 PM

Figure 14. Vehicle Dealers
Holds information about dealerships

WhatNext Vision Motors

Vehicle Customers ▾ Vehicle Dealers ▾ **Vehicle Orders** ▾ Vehicle Service Requests ▾

Vehicle Order O-0003

Vehicle Order "O-0003" was created.

Details

Vehicle Order Number O-0003	Owner Rafflesia Gregorio
Vehicle Customer John	
Vehicle Honda	
Order Date 11/25/2025	
Status Confirmed	
Assigned Dealer	
Created By Rafflesia Gregorio, 11/21/2025, 11:12 PM	Last Modified By Rafflesia Gregorio, 11/21/2025, 11:12 PM

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

The screenshot shows a software application window titled "Vehicle" with a sub-section for "Honda". At the top right is a "New" button. Below the title, there are two tabs: "Related" and "Details", with "Details" being the active tab. The "Details" section contains the following fields:

Vehicle Name	Honda	Owner	Rafflesia Gregorio
Vehicle Model	EV		
Stock Quantity	99		
Price	\$80,000		
Vehicle Dealer	WW		
Status	Available		
Created By	Rafflesia Gregorio, 11/21/2025, 8:10 PM	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.

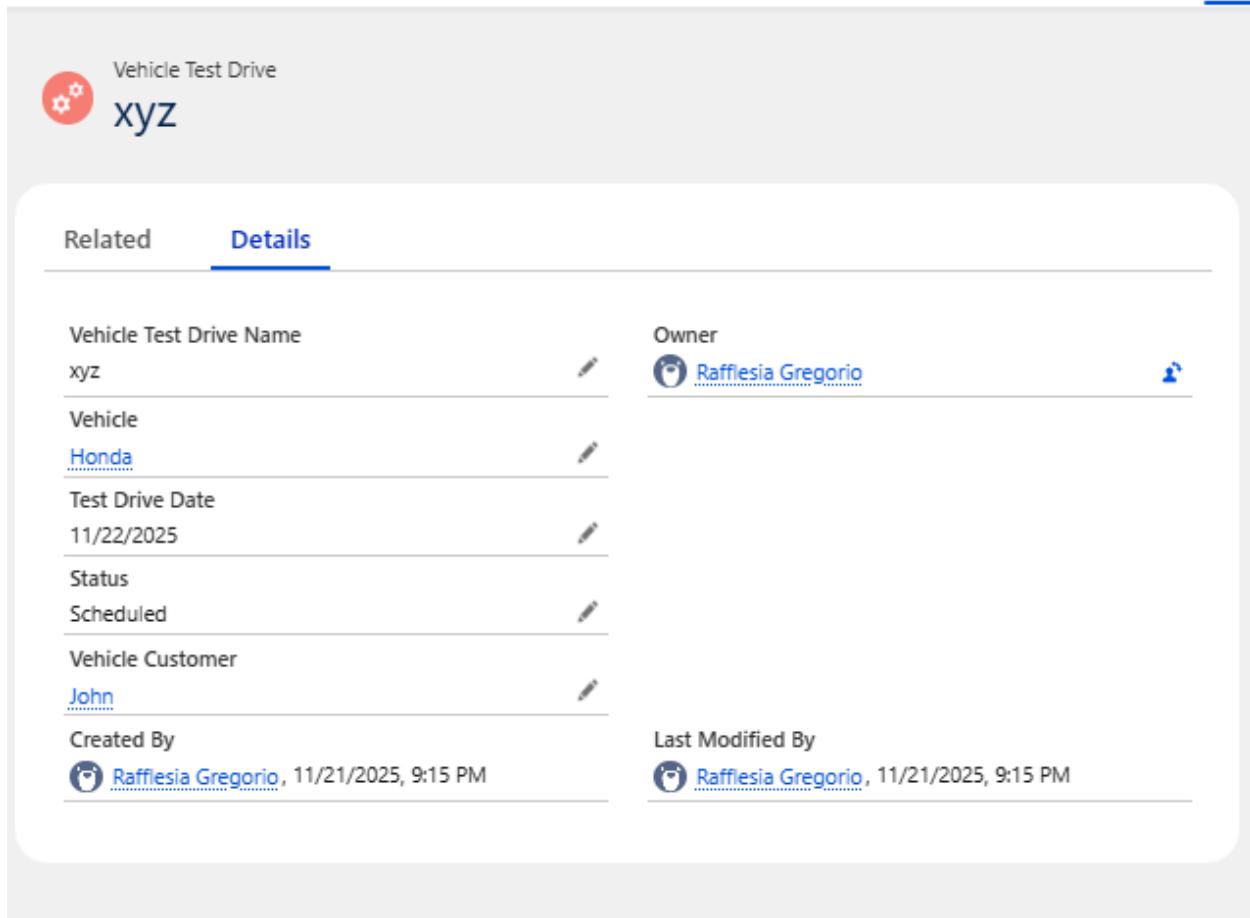


Figure 17. Vehicle Test Drive

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.