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

WhatsNext Vision Motors is upgrading its customer experience through a new Salesforce project focused on improving the vehicle ordering process. The system automatically recommends the nearest dealer based on customer address and prevents orders for out-of-stock vehicles to avoid confusion. It also includes a scheduled process that updates order statuses: marking them as **Pending** if the vehicle is unavailable and **Confirmed** if it is in stock.

This project aims to streamline operations, reduce errors, enhance transparency, and improve customer satisfaction. By automating key processes, the company can lessen staff workload and increase overall efficiency, allowing employees to focus on more strategic tasks.

Requirements :

1. **Salesforce CRM Implementation**

- Store and manage vehicle details, stock availability, and dealer information in Salesforce.
- Track customer orders, test drives, and service requests efficiently.
- Automate workflows to assign orders to the nearest dealer based on customer location.

2. **Process Automation**

- Prevent order placement if the vehicle is out of stock.
- Auto-assign orders to the nearest dealer based on the customer’s location.
- Send automated email reminders for scheduled test drives.

3. Apex and Trigger

- Implement Apex triggers to enforce business rules such as stock validation and automatic dealer assignment.
- Use trigger handlers to follow best practices and ensure modularity and maintainability.

4. Batch Jobs

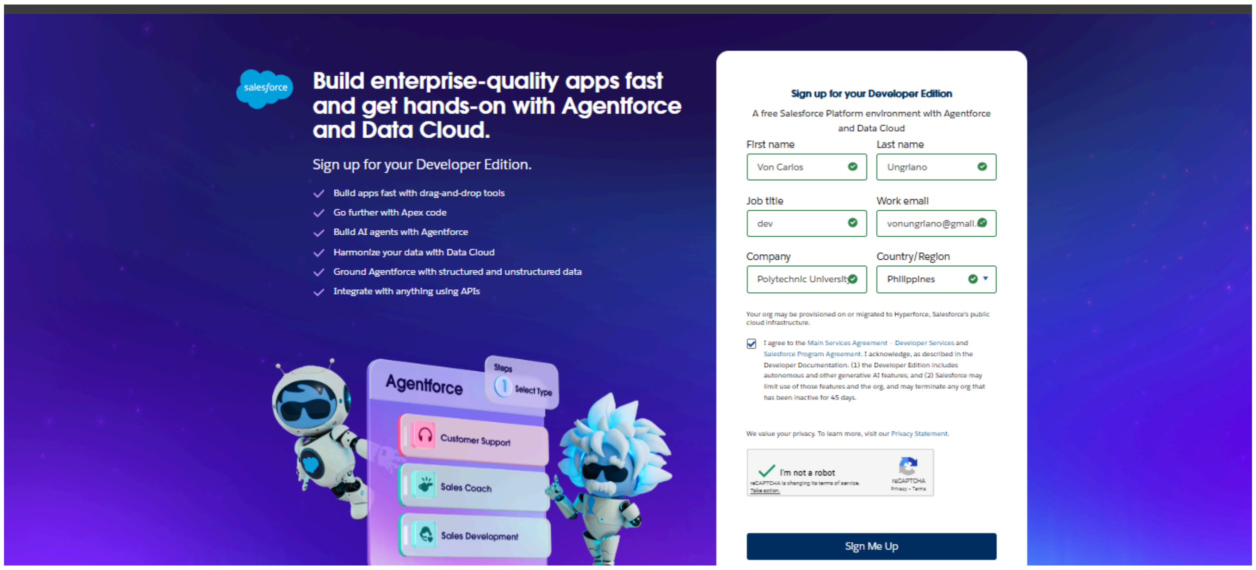
- Develop a batch Apex job to periodically check vehicle stock levels and update availability.
- Send scheduled email notifications for stock replenishment and order processing.

What you'll learn :

1. Data Modelling
2. Fields and Relationships
3. Lightning App Builder
4. Record Triggered Flows
5. Apex and Apex Triggers
6. Batch Apex
7. Scheduled Apex

Creating a developer org in salesforce.

1. Go to https://developer.salesforce.com/signup
2. On the sign up form, enter the following details :



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

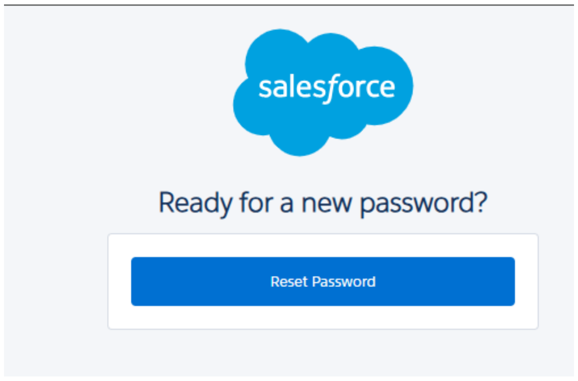
A free Salesforce Platform environment with Agentforce and Data Cloud

First name: Von Carlos	Last name: Ungriano
Job title: dev	Work email: vonungriano@gmail.
Company: Polytechnic Universit	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.



Ready for a new password?

Objects & Relationships

Object Name	**Purpose**	**Relationships**
Vehicle__c	Stores vehicle details	Related to Dealer & Orders
Vehicle_Dealer__c	Stores authorized dealer info	Related to Orders
Vehicle_Customer__c	Stores customer details	Related to Orders & Test Drives

Vehicle_Order__c	Tracks vehicle purchases	Related to Customer & Vehicle
Vehicle_Test_Drive__c	Tracks test drive bookings	Related to Customer & Vehicle
Vehicle_Service_Request__c	Tracks vehicle servicing requests	Related to Customer & Vehicle

To create an object:

1. From the setup page → Click on Object Manager → Click on Create → Click on Custom Object.
2. Enter the label name→ Vehicle
3. Plural label name→ Vehicles
4. Enter Record Name Label and Format
    a. Record Name → Vehicle Name
    b. Data Type → Text
5. Click on Allow reports,
6. Allow search → **Save.**

Create Dealer Object

The purpose of creating a Dealer object is to have detailed information about Dealer.

To create an object:

1. From the setup page → Click on Object Manager → Click on Create → Click on Custom Object.
2. Enter the label name→ Vehicle Dealer
3. Plural label name→ Vehicle Dealers
4. Enter Record Name Label and Format
    - Record Name → Dealer Name
    - Data Type → Text
5. Click on Allow reports,
6. Allow search → **Save**

Creating a Custom Tab(Vehicle)

1. Go to setup page → type Tabs in Quick Find bar → click on tabs → New (under custom object tab)
2. Select Object(Vehicle) → Select any tab style → Next (Add to profiles page) keep it as default → Next (Add to Custom App) keep it as default → Save.

Create a Lightning App

To create a lightning app page:

1. Go to setup page → search “app manager” in quick find → select “app manager” → click on New lightning App.
2. Fill the app name in app details and branding as follow
   App Name : WhatNext Vision Motors
   Developer Name : this will auto populated
   Description : Give a meaningful description
   Image : optional (if you want to give any image you can otherwise not mandatory)
   Primary color hex value : keep this default
3. Then click Next → (App option page) keep it as default → Next → (Utility Items) keep it as default → Next.
4. To Add Navigation Items:

earch the items in the search bar(Vehicle, Dealer, Customer, Order, Test Drive, Service Request, Reports, Dashboard) from the search bar and move it using the arrow button → Next.
Note: select the custom object which we have created in the previous activity.

1. To Add User Profiles:

Search profiles (System administrator) in the search bar → click on the arrow button → save & finish.

Fields & Relationships

Key Fields for Each Object

1. Vehicle__c (Custom Object)

- **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)

2.Vehicle_ Dealer__c (Custom Object)

- **Dealer_Name__c** (Text)
- **Dealer_Location__c** (Text)
- **Dealer_Code__c** (Auto Number)
- **Phone__c** (Phone)
- **Email__c** (Email)

3. Vehicle_Order__c (Custom Object)

- **Customer__c** (Lookup to Customer__c)
- **Vehicle__c** (Lookup to Vehicle__c)

- **Order_Date__c** (Date)
- **Status__c** (Picklist: Pending, Confirmed, Delivered, Canceled)

4. Vehicle_Customer__c (Custom Object)

- **Customer_Name__c** (Text)
- **Email__c** (Email)
- **Phone__c** (Phone)
- **Address__c** (Text)
- **Preferred_Vehicle_Type__c** (Picklist: Sedan, SUV, EV, etc.)

5. Vehicle_Test_Drive__c (Custom Object)

- **Customer__c** (Lookup to Customer__c)
- **Vehicle__c** (Lookup to Vehicle__c)
- **Test_Drive_Date__c** (Date)
- **Status__c** (Picklist: Scheduled, Completed, Canceled)

6. Vehicle_Service_Request__c (Custom Object)

- **Customer__c** (Lookup to Customer__c)
- **Vehicle__c** (Lookup to Vehicle__c)
- **Service_Date__c** (Date)
- **Issue_Description__c** (Text)

Status__c (Picklist: Requested, In Progress, Completed)

Creating a Field in Vehicle Object

1. Go to setup → click on Object Manager → type object name(Vehicle) in quick find bar→ click on the object.
2. Now click on “Fields & Relationships” → New
3. Select Data type as “PickList”.
4. Click on Next
5. Fill the above as following:
   a. Field Label: Vehicle Model
   b. Fill PickList Values
   c. Click on Next → Next → Save and new.

Creating Stock Quantity field in vehicle object

1. Repeat the steps 1 & 2 from the activity 1
2. Select Data type as “Number”.
3. Click on Next
4. Fill the above as following:
    a. Field Label: Stock Quality
    b. Click on Next → Next → Save and new.

Creating a Lookup RelationShip in Vehicle Object to Dealer Object.

1. Select DataType as Lookup Relationship.
2. Click Next
3. Select Dealer Object Form Related Filed.
4. Click Next and Save.

Creating a record triggered flow to assign nearest dealer to the customer’s location

Step 1 : In Quick Find, type Flows and click on Flows. Click New Flow.

Step 2 : Select Start From Scratch and click Next.

Step 3 : Select Record-Triggered Flow and click CreateStep 4 : Select Vehicle Order Object

Trigger the Flow When : Select Record is Created.

Set Entry Condition :
All Conditions Are Met (AND)
Filed : Status__c

Operator : Equals
Value : Pending

Step 5 : Click **+** → Select **Get RecordsStep**

6 : Label : Get Customer Information

- Object : Vehicle Customer
- Condition :
- Field : Id
- Operator : Equals
- Value : {!$Record.Vehicle_Customer__c}

step 7 : Click + → Select Get Records

- Label : Get Nearest Dealer
- Object : Vehicle Dealer
- Condition :
- Field : Dealer_Location__c
- Operator : Equals
- Value : {!Get_Customer_Information.Address__c}

Step 8 : Click + → Select Update Records

- Label : Assign Dealer to Order
- *How to Find Records to Update and Set Their Values : Use the IDs and all field values from a record or record collection
- Select Record(s) to Update : {!Get_Nearest_Dealer}

Step 9 : Click Save and Give label Name and Activate Flow.

- Label Name : Auto Assign Dealer

Step 10 : Activate Flow

Creating record triggered flow to send an email to the customer reminding about the test drive.

Step 1 : Select Record-Triggered Flow and click Create

Step 2 : Select Vehicle Test Drive Object

Trigger the Flow When : A record is created or updated

Set Entry Condition :

- All Conditions Are Met (AND)
- Filed : Status__c
- Operator : Equals
- Value : Scheduled

Step 3 : Click + Add Scheduled Paths (below the trigger).

- Label: Reminder Before Test Drive.
- Time Source: Test_Drive_Date__c
- Offset Number: 1.
- Offset Options: Days Before.
- Click Done.

Step 4 :

1. Click + Add Element → Get Records.
2. Label: Get Customer Information.
3. Object: Vehicle_Customer__c.
4. Filter Conditions: Id = {!$Record.Customer__c}.
5. How Many Records to Store: Select Only the first record.
6. How to Store Record Data: Choose Automatically store all fields.

Step 5: Send Reminder Email

1. Click + Add Element → Action.
2. Action Type: Send Email
3. Label: Send Test Drive Reminder.
4. Subject: "Reminder: Your Test Drive is Tomorrow!".
5. Recipient Address List : {!Get_Customer_Information.Email__c}

6. Rich-Text-Formatted Body : True
7. Body : Create Variable

8 . Api Name : EmailSent

Step 6 : Click save

- **Label Name :** Test Drive Reminder

Step 7 : Activate Flow

Create Apex and Trigger Batch Jobs :

Step 1 : Click Developer Console From Gear icon

Step 3 : Give Name For Apex Class

- Class Name : VehicleOrderTriggerhandler

Step 4 : Write Apex Code

Source Code :

VehicleOrderTriggerHandler:-

```
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);

            }

        }



        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;

            }

        }

    }

}
```

Step 5 : Write Trigger Handler

Step 6 : Writer Trigger Class Name and Select Vehicle Order Object

Step 7 : Call Apex Class in Trigger Class

Source Code:

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);
}
```

Step 8 : Create Batch Job
A customer places an order, but the vehicle is out of stock.

- The order remains pending.
- After new stock is added, the batch job updates the order to confirmed.

Source Code:

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.');
    }
}
```

Step 9 : Create Schedule Class then revoked Batch Class in Schedule Class

Source Code :

VehicleOrderBatchScheduler:-

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

Conclusion:

The Salesforce implementation for WhatsNext Vision Motors marks a major step toward modernizing its operations and elevating the customer experience. By automating dealer assignment, validating stock availability, and streamlining order and test-drive processes, the company ensures smoother, faster, and more transparent customer interactions. The use of Apex triggers, batch jobs, and record-triggered flows strengthens the system's reliability and accuracy, reducing manual work while improving overall efficiency. With these innovations, WhatsNext Vision Motors is now better equipped to meet customer needs, optimize internal processes, and continue leading the mobility sector with excellence and innovation.

Thank you!