

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

## **Abstraction**

The Salesforce Virtual Internship program, hosted via SkillUp Wallet and Trailhead, offers students an opportunity to explore how Customer Relationship Management (CRM) systems can be applied to real-life business challenges. In this internship, participants worked on a simulated project for **WhatsNext Vision Motors**, an innovative automotive company seeking to streamline and modernize its order and delivery workflows.

The core aim was to use Salesforce tools to digitize manual business processes—automating dealer assignment, validating inventory availability, updating order statuses in real-time, and ensuring seamless communication between customers, dealers, and the organization. Through a mix of declarative tools (like Flow and Process Builder) and programmatic approaches (like Apex triggers and batch classes), participants learned to build scalable and intelligent cloud-based solutions.

This document outlines the entire lifecycle of the project—from understanding the problem to implementing solutions and exploring the long-term potential of Salesforce in transforming customer service operations in the automotive domain.

## **Objectives**

### **1. Automate Order Management:**

To streamline the customer order process by automatically assigning the nearest available dealer based on the customer's location.

### **2. Prevent Out-of-Stock Orders:**

To implement validation rules and inventory checks to prevent customers from placing orders for vehicles that are not available.

### **3. Enhance Dealer Allocation:**

To build a system that dynamically assigns dealers using Salesforce Flows, improving delivery efficiency and customer satisfaction.

### **4. Enable Real-Time Inventory Updates:**

To utilize Apex Triggers to ensure the inventory is updated immediately after an order is placed.

### **5. Use Batch Processing for Order Updates:**

To implement Batch Apex classes that periodically update pending orders and notify stakeholders of changes or delays.

## 6. Improve Customer Experience:

To create a more transparent and responsive order tracking system for customers using Salesforce Dashboards and Reports.

## 7. Leverage Low-Code Tools:

To apply Salesforce's low-code solutions (like Process Builder and Flow Builder) to solve complex business logic with minimal coding.

## 8. Develop Admin and Developer Skills:

To gain practical experience with both admin-level configuration and developer-level customization using the Salesforce platform.

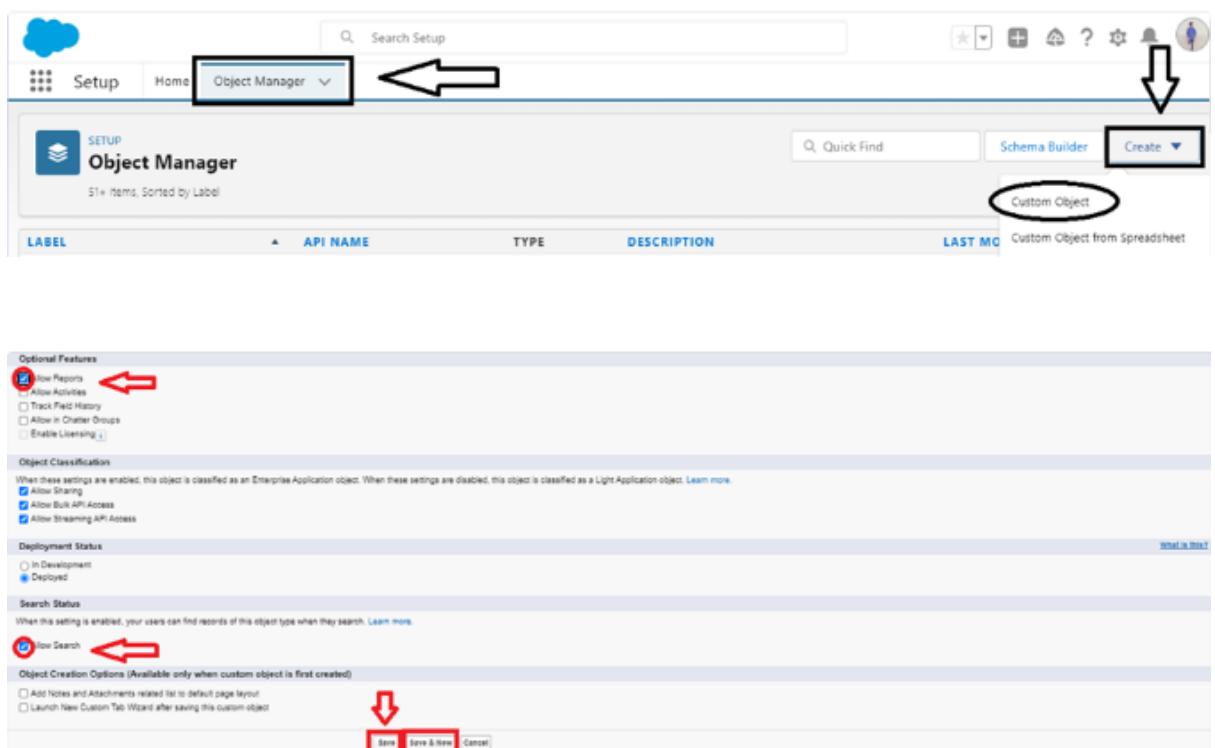
# Data Management-Objects

## 1. Vehicle\_\_c (Stores vehicle details)

Related to: Dealer & Orders

Steps:

- Go to Setup > Object Manager > Create > Custom Object → Name it Vehicle.
- Add fields like Model, Price, Type.
- Create Lookup Relationship fields to Dealer\_\_c and Order\_\_c.



**SETUP**

## New Custom Object

Custom Object Definition Edit

**Custom Object Information**

The singular and plural labels are used in tabs, page layouts, and reports.

Label	<input type="text"/>	Example: Account
Plural Label	<input type="text"/>	Example: Accounts
Starts with vowel sound <input type="checkbox"/>		

The Object Name is used when referencing the object via the API.

Object Name	<input type="text"/>	Example: Account
Description	<input type="text"/>	

Context-Sensitive Help Setting:

- Open the standard Salesforce.com Help & Training window
- Open a window using a Visualforce page

Content Name:

**Enter Record Name Label and Format**

The Record Name appears in page layouts, key lists, related lists, lookups, and search results. For example, the Record Name for Account is "Account Name" and for Case it is "Case Number". Note that the Record Name field is required.

Record Name	<input type="text"/>	Example: Account Name
Data Type	<input type="text" value="Text"/>	Warning: If you plan to insert a high volume of records in this object, via the API for example, use the Text data type.

## 2. Vehicle\_Dealer\_\_c (Authorized dealer info)

Related to: Orders

Steps:

- Create object Vehicle\_Dealer under Object Manager.
- Add fields like Dealer Name, Location.
- Create Lookup Relationship to Order\_\_c.

## 3. Vehicle\_Customer\_\_c (Customer details)

Related to: Orders & Test Drives

Steps:

- Create object Vehicle\_Customer.
- Add fields like Customer Name, Phone, Email.
- Add Lookup Relationships to Order\_\_c and Test\_Drive\_\_c.

## 4. Vehicle\_Order\_\_c (Vehicle purchases)

Related to: Customer & Vehicle

Steps:

- Create object Vehicle\_Order.

- Add fields like Order Date, Status, Total Amount.
- Add Lookup Relationships to Vehicle\_\_c and Customer\_\_c.

## 5. Vehicle\_Test\_Drive\_\_c (Test drive bookings)

Related to: Customer & Vehicle

Steps:

- Create object Vehicle\_Test\_Drive.
- Add fields like Booking Date, Time, Status.
- Add Lookup Relationships to Vehicle\_\_c and Customer\_\_c.

## 6. Vehicle\_Service\_Request\_\_c (Service requests)

Related to: Customer & Vehicle

Steps:

- Create object Vehicle\_Service\_Request.
- Add fields like Request Date, Issue Type, Status.
- Add Lookup Relationships to Vehicle\_\_c and Customer\_\_c.

# Data Management – Object Tab Creation

Task:

Creating a Custom Tab for the Vehicle\_\_c object to make it visible and accessible in the app.



1. Go to Setup → type “Tabs” in the Quick Find bar → click on Tabs.
2. Under Custom Object Tabs, click on New.

The screenshot shows the Salesforce Setup interface. The top navigation bar includes 'Setup', 'Home' (which is highlighted with a red box), and 'Object Manager'. Below the navigation is a search bar with the placeholder 'Q\_ tabs'. A sidebar on the left contains sections like 'User Interface' (with 'Rename Tabs and Labels' and 'Tabs' highlighted with a red box), 'Analytics', 'Tab Embedding', and 'Decision Tables'. A message at the bottom of the sidebar says 'Didn't find what you're looking for? Try using Global Search.' The main content area is titled 'SETUP Tabs' and features a section titled 'Custom Tabs' with the sub-section 'Custom Object Tabs'. It displays the message 'No Custom Object Tabs have been defined' and includes 'New' and 'What Is This?' buttons. Below it are sections for 'Web Tabs', 'Visualforce Tabs', and 'Lightning Component Tabs', each with similar 'No [Type] Tabs have been defined' messages and 'New' and 'What Is This?' buttons.

3. Select Object: Vehicle \_\_c → choose any Tab Style → click Next.

The screenshot shows the 'New Custom Object Tab' setup page, Step 1 of 3. The top navigation bar is identical to the previous screenshot. The sidebar on the left includes 'Feature Settings', 'Analytics', 'Tab Embedding' (highlighted with a red box), 'Decision Tables', 'User Interface', 'Loaded Console Tab Limits', 'Rename Tabs and Labels', and 'Tabs'. A message at the bottom of the sidebar says 'Didn't find what you're looking for? Try using Global Search.' The main content area is titled 'SETUP Tabs' and 'New Custom Object Tab'. It has a sub-section 'Step 1. Enter the Details' with the instruction 'Choose the custom object for this new custom tab. Fill in other details.' It includes fields for 'Object' (set to 'None'), 'Tab style' (set to 'None', with 'Vehicle' highlighted with a red box), and 'Home Page Custom Link' (set to 'None'). There is also a 'Description' field and 'Next Step' buttons.

4. Keep default profile and app settings → click Next → then Save.

This allows users to easily view, create, and manage Vehicle records from the navigation bar in Salesforce.

# Data Management – App Manager

Task:

Create a Lightning App named *WhatNext Vision Motors* to organize and navigate between custom objects easily.

Steps to Create the Lightning App:

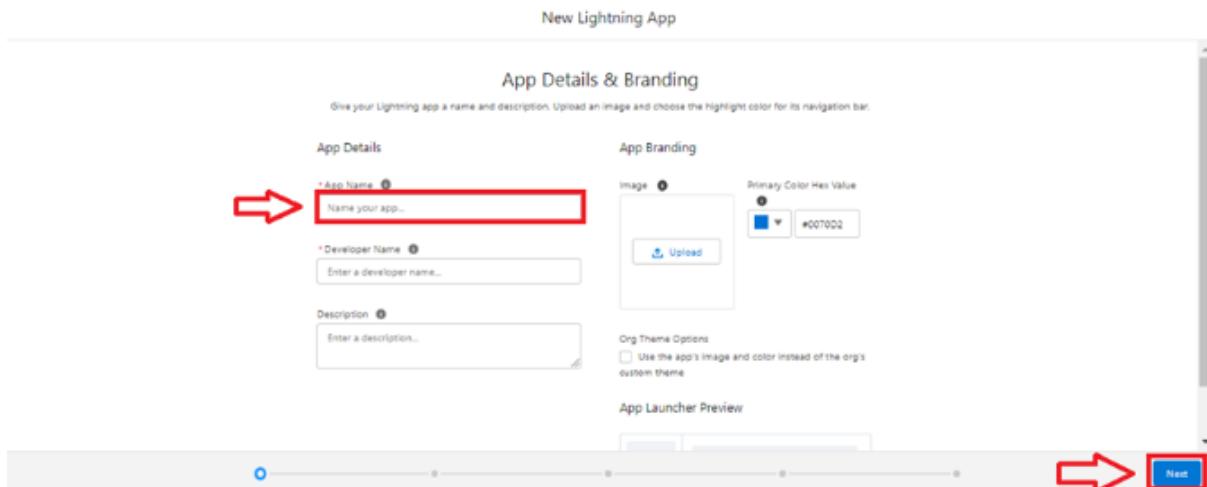
1. Go to Setup → search "App Manager" in the Quick Find bar → click App Manager → click New Lightning App.

The screenshot shows the Salesforce App Manager interface. At the top, there is a search bar with the text "app manager" and a "Search Setup" button. Below the search bar, there are tabs for "Setup", "Home", and "Object Manager". A red arrow points to the "App Manager" tab, which is highlighted in yellow. Another red arrow points to the "New Lightning App" button in the top right corner. The main area displays a list of existing apps with columns for "App Name", "Developer Name", "Description", "Last Modified", "App Type", and "V...". The list includes apps like "All Tab", "Analytics Studio", "App Launcher", "Bolt Solutions", "Chatter Desktop", "Chatter Mobile for BlackBerry", "College Management System", "Community", "Content", and "Data Manager".

2. Fill in the App Details:

- App Name: WhatNext Vision Motors
- Developer Name: Auto-filled
- Description: Provide a brief, meaningful description
- Image: Optional
- Primary Color Hex: Leave default  
→ Click Next

3. App Options Page: Leave everything as default → Click Next



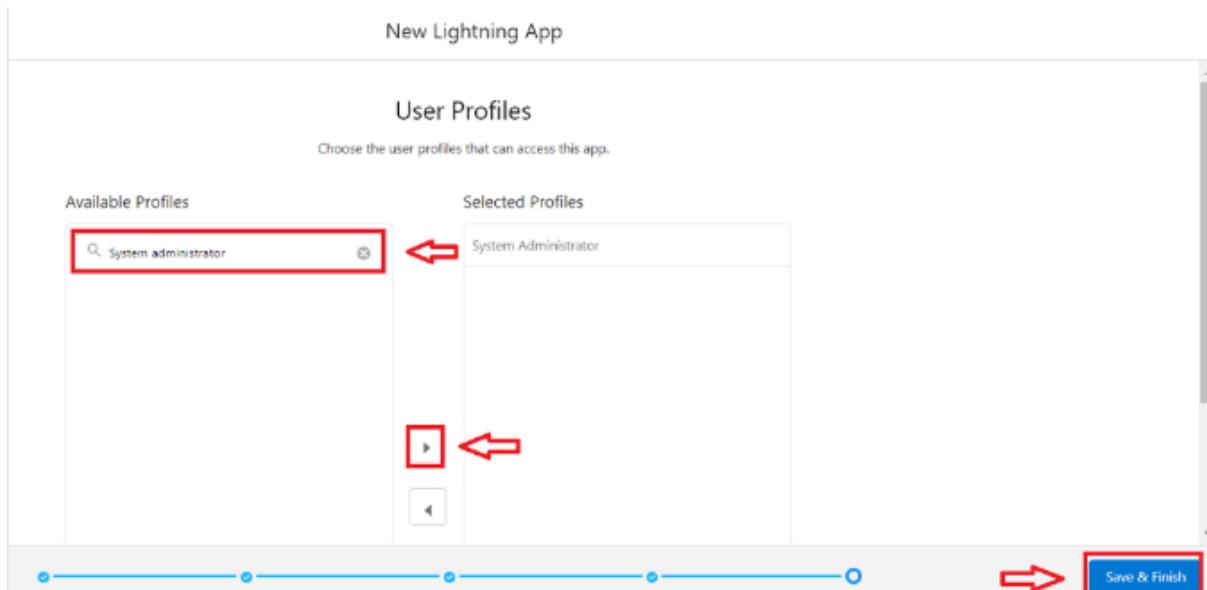
4. Utility Items: Skip or leave default → Click Next

#### Add Navigation Items:

- Search and add the following custom objects:  
Vehicle, Dealer, Customer, Order, Test Drive, Service Request, Reports, Dashboards
- Use the arrow button to move them to the selected list  
→ Click Next

#### Add User Profiles:

- Search for System Administrator
- Move it using the arrow button  
→ Click Save & Finish



# Data Management – Fields & Relationships

This section outlines the key fields created for each custom object in the WhatsNext Vision Motors app. Fields were defined to capture relevant data and build relationships among Vehicle, Dealer, Customer, and Orders.

## Creating a Field in Vehicle\_\_c Object

Steps:

1. Go to Setup → Object Manager → search Vehicle → click on it.

The screenshot shows the Salesforce Object Manager interface. In the top right, there is a search bar with 'Q. vehicle' and a 'Create' button. Below the search bar, a table lists objects: 'Vehicle' (API Name: Vehicle\_\_c, Type: Custom Object) with a last modified date of 13/02/2025 and a deployed status. The 'Vehicle' row is highlighted with a red border.

2. Click Fields & Relationships → New.

The screenshot shows the 'Fields & Relationships' page for the Vehicle object. The 'Fields & Relationships' tab is selected. At the top right, there is a 'New' button, which is highlighted with a red box. The table below lists two fields: 'Created By' (Field Label: Created By, Field Name: CreatedById, Data Type: Lookup(Users)) and 'Last Modified By' (Field Label: Last Modified By, Field Name: LastModifiedById, Data Type: Lookup(Users)). A 'Controlling Field' column contains 'Controlling Field' for both rows.

3. Select Data Type: *Picklist* → Next.

The screenshot shows the 'Step 1. Select the field type' screen of the 'New Custom Field' wizard. The 'Fields & Relationships' tab is selected. On the left, a sidebar lists various field types: Details, Fields & Relationships (selected), Page Layouts, Lightning Record Pages, Patterns, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Restriction Rules, Scoping Rules, Object Access, and Record Labels. On the right, a list of data types is shown with 'Picklist' selected and highlighted with a red box. A detailed description of the Picklist type is provided: 'Allows users to select a value from a list you define'. Below the list, other options like 'Text' and 'Phone' are also listed with their descriptions.

The screenshot shows the 'Step 2. Enter the details' screen of the 'New Custom Field' wizard. The 'Fields & Relationships' tab is selected. The 'Field Label' is set to 'Vehicle Model'. Under the 'Values' section, the 'Type values, with each value separated by a new line' radio button is selected and highlighted with a red box. A text area contains the values 'SUV', 'CUV', 'EV', and 'Sed'. Below the text area, there are three checkboxes: 'Display values alphabetically, not in the order entered', 'Use first value as default value', and 'Require picklist to the values defined in the value set', which is checked and highlighted with a red box. The 'Field Name' is set to 'Vehicle\_Model' and the 'Description' is left blank.

**4. Fill:**

- **Field Label:** Vehicle Model
- **Picklist Values:** Sedan, SUV, EV, etc.

**5. Click Next → Next → Save & New.**

## **2 Creating a Field in Vehicle\_Dealer\_\_c Object**

Steps:

1. Go to Setup → Object Manager → search Vehicle\_Dealer → open the object.
2. Click Fields & Relationships → New.
3. Select Data Type: *Auto Number* → Next.
4. Fill:
  - **Field Label:** Dealer Code
  - **Display Format:** DL-{0000}
  - **Starting Number:** 1
5. Click Next → Next → Save.

## **3 Creating a Field in Vehicle\_Order\_\_c Object**

Steps:

1. Go to Setup → Object Manager → open Vehicle\_Order.
2. Click Fields & Relationships → New.
3. Select Data Type: *Lookup Relationship* → Next.
4. Related Object: Vehicle\_\_c
  - **Field Label:** Vehicle
  - **Field Name:** Vehicle
5. Click Next → Next → Save.

## **4 Creating a Field in Vehicle\_Customer\_\_c Object**

Steps:

1. Go to Setup → Object Manager → open Vehicle\_Customer.

2. Click Fields & Relationships → New.
3. Select Data Type: *Picklist* → Next.
4. Fill:
  - Field Label: Preferred Vehicle Type
  - Picklist Values: Sedan, SUV, EV
5. Click Next → Next → Save.

## **5 Creating a Field in Vehicle\_Test\_Drive\_\_c Object**

Steps:

1. Go to Setup → Object Manager → open Vehicle\_Test\_Drive.
2. Click Fields & Relationships → New.
3. Select Data Type: *Picklist* → Next.
4. Fill:
  - Field Label: Status
  - Picklist Values: Scheduled, Completed, Canceled
5. Click Next → Next → Save.

## **6 Creating a Field in Vehicle\_Service\_Request\_\_c Object**

Steps:

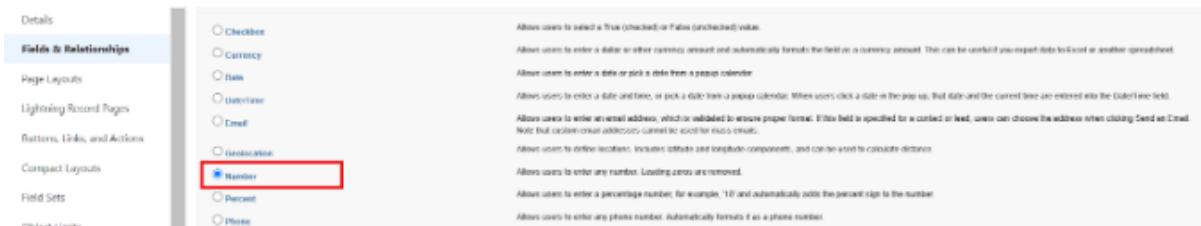
1. Go to Setup → Object Manager → open Vehicle\_Service\_Request.
2. Click Fields & Relationships → New.
3. Select Data Type: *Picklist* → Next.
4. Fill:
  - Field Label: Status
  - Picklist Values: Requested, In Progress, Completed
5. Click Next → Next → Save.

# Creating Stock Quantity Field in Vehicle\_\_c Object

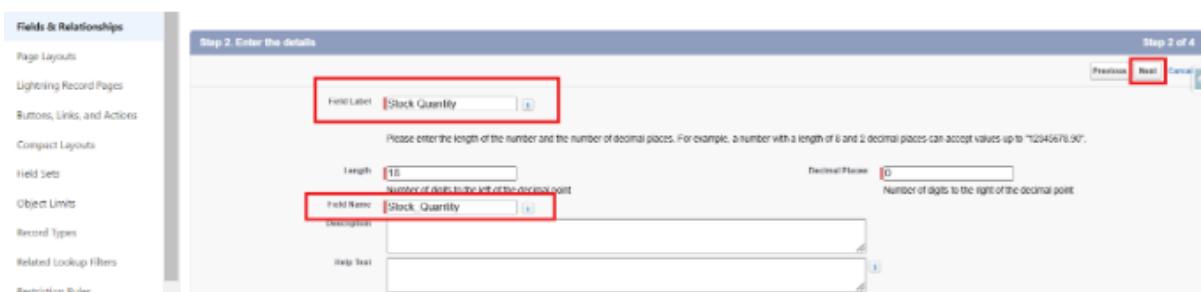
Steps:

## 1. Repeat Step 1 & 2 from the previous field activity:

- Go to Setup → Object Manager → search for Vehicle → click on it.
- Click Fields & Relationships → New.



## 2. Select Data Type: Number → click Next.



## 3. Fill in the details:

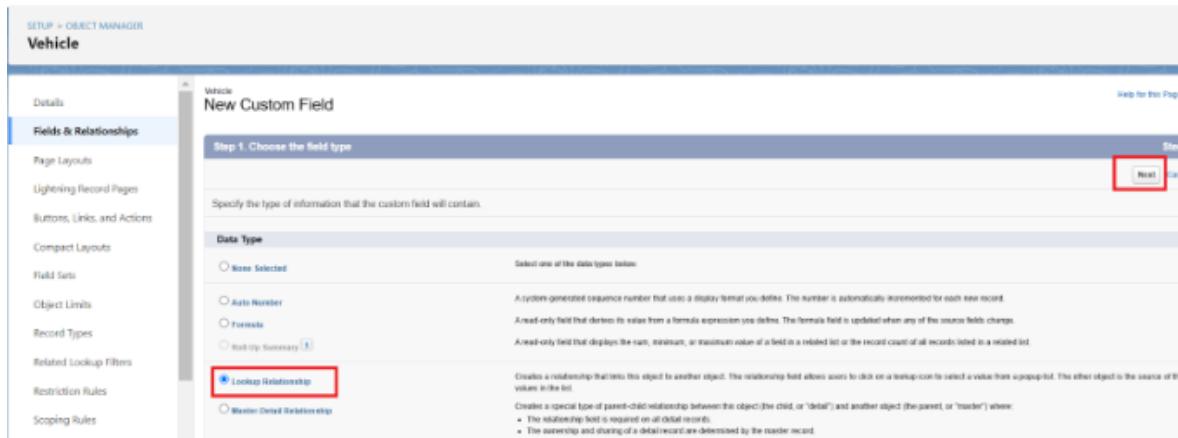
- Field Label: Stock Quantity

## 4. Click Next → Next → Save & New (if adding more fields).

# Creating a Lookup Relationship in Vehicle\_\_c to Dealer\_\_c

Steps:

## 1. Go to Setup → Object Manager → search and select Vehicle.



2. Click **Fields & Relationships** → **New**.
3. Select **Data Type: Lookup Relationship** → click **Next**.

**Step 1: New Relationship**

Related To: Dealer

**Step 2: Step 3. Enter the label and name for the lookup field**

Field Label: Dealer  
Field Name: Dealer

**Step 3: Step 4. Establish field-level security for reference field**

Visible (checkbox checked)

4. Choose **Related Object: Dealer\_c** → click **Next**.
5. Configure field label (e.g., **Dealer**) → click **Next** → **Next** → **Save**.

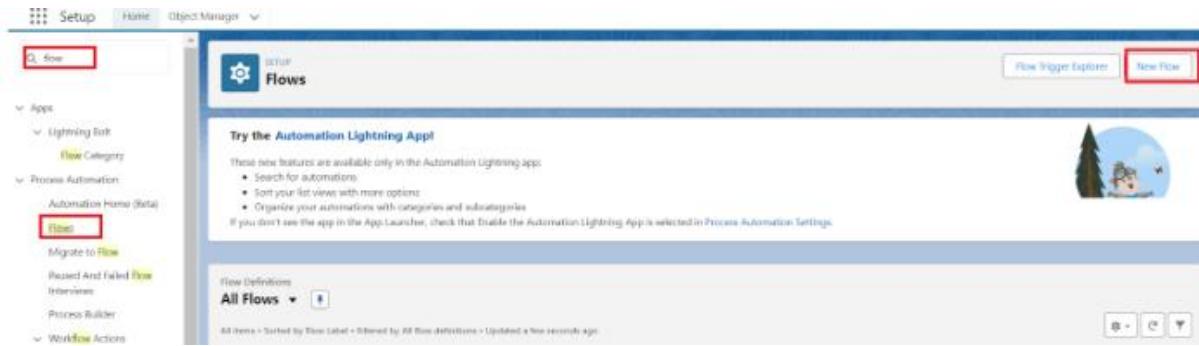
Create all other fields and relationship like same

## Automation

This automation ensures that when a new Vehicle Order is created with status "Pending", the system will auto-assign the nearest Vehicle Dealer based on the customer's location.

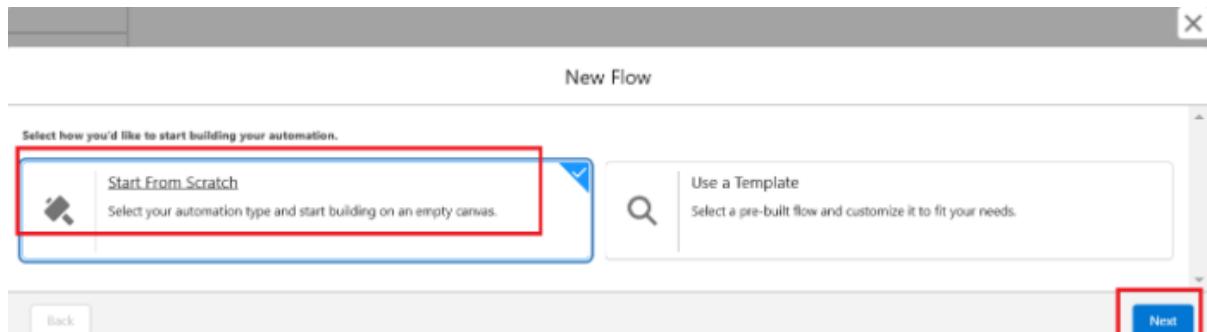
### Steps to Create Record-Triggered Flow

1. Go to Flows
  - o In Quick Find, type Flows → Click New Flow



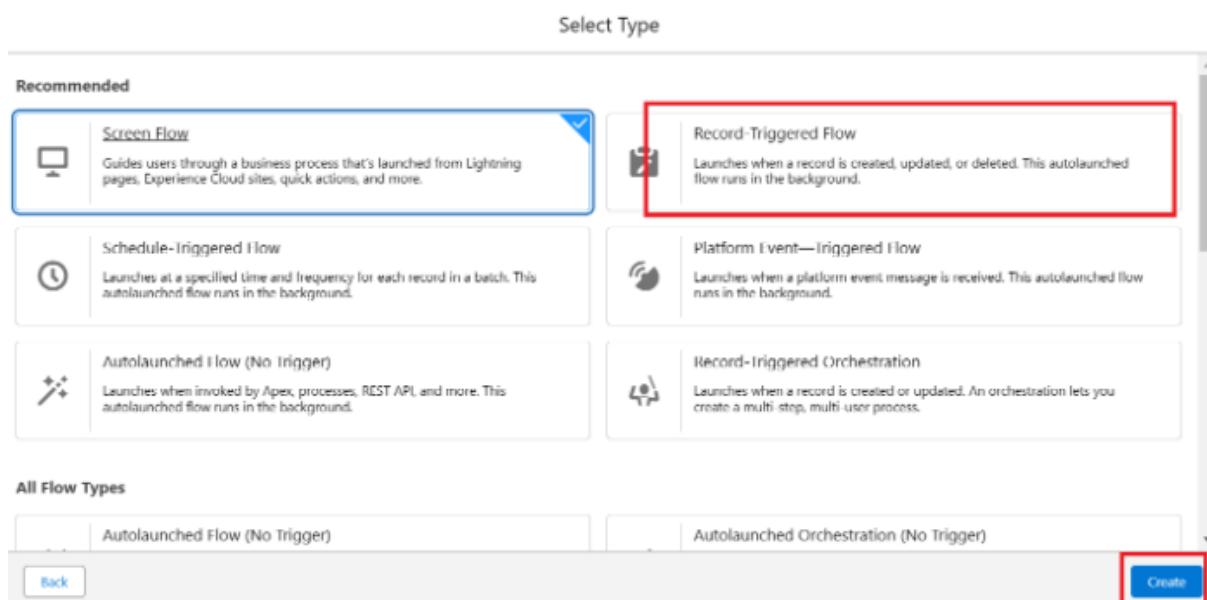
## 2. Start From Scratch

- Select "Start from Scratch" → Click Next



## 3. Choose Flow Type

- Select Record-Triggered Flow → Click Create



## Trigger Configuration

- Object: Vehicle\_Order\_\_c
- Trigger the Flow When: Record is Created

- Condition Requirements:
  - All Conditions Are Met (AND)
    - Status\_\_c Equals Pending

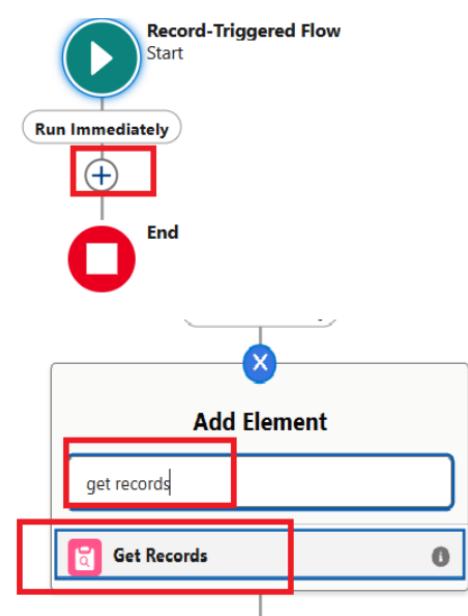
## Step 1: Get Customer Information

- Click + → Get Records

Value : Pending

```
app = Flask(__name__)
```

Step 5 : Click + → Select Get Records



- Label: Get Customer Information

Label	API Name						
Get Customer Information.	Get_Customer_Information						
Description							
Get Records of This Object							
Object							
Vehicle Customer							
Filter Vehicle Customer Records							
Condition Requirements							
All Conditions Are Met (AND)							
<table border="1"> <tr> <td>Field</td> <td>Operator</td> <td>Value</td> </tr> <tr> <td>Id</td> <td>Equals</td> <td>\$Record &gt; Vehicle Customer</td> </tr> </table>		Field	Operator	Value	Id	Equals	\$Record > Vehicle Customer
Field	Operator	Value					
Id	Equals	\$Record > Vehicle Customer					
<a href="#">+ Add Condition</a>							
Sort Vehicle Customer Records							

- Object: Vehicle\_Customer\_\_c
- Condition:
  - Id Equals {\$Record.Vehicle\_Customer\_\_c}
- Store Only the first record
- Automatically store all fields

## Step 2: Get Nearest Dealer

Get Records

\* Label: Get Nearest Dealer \* API Name: Get\_Nearest\_Dealer

Description:

Get Records of This Object

\* Object: Vehicle Dealer

Filter Vehicle Dealer Records

Condition Requirements

All Conditions Are Met (AND)

Field	Operator	Value
Dealer_Location__c	Equals	A_a Vehicle Customer from Get_Cu...

+ Add Condition

- Click + → Get Records
- Label: Get Nearest Dealer
- Object: Vehicle\_Dealer\_\_c
- Condition:
  - Dealer\_Location\_\_c Equals {!Get\_Customer\_Information.Address\_\_c}
- Store Only the first record
- Automatically store all fields

## Step 3: Update the Order Record

- Click + → Update Records
- Label: Assign Dealer to Order

**Update Records**

\*Label: Assign Dealer to Order      \*API Name: Assign\_Dealer\_to\_Order

Description:

\* How to Find Records to Update and Set Their Values

- Use the vehicle order record that triggered the flow
- Update records related to the vehicle order record that triggered the flow
- Use the IDs and all field values from a record or record collection
- Specify conditions to identify records, and set fields individually

Select Record(s) to Update

Record or Record Collection: Vehicle Dealer from Get Nearest Dealer X

Make sure that each record has an ID. Otherwise the flow can't find the records to update, and it fails. ⓘ

- How to Find Records:

Save as

Save As: A New Version

\*Flow Label: Auto Assign Dealer      \*Flow API Name: Auto\_Assign\_Dealer

Description:

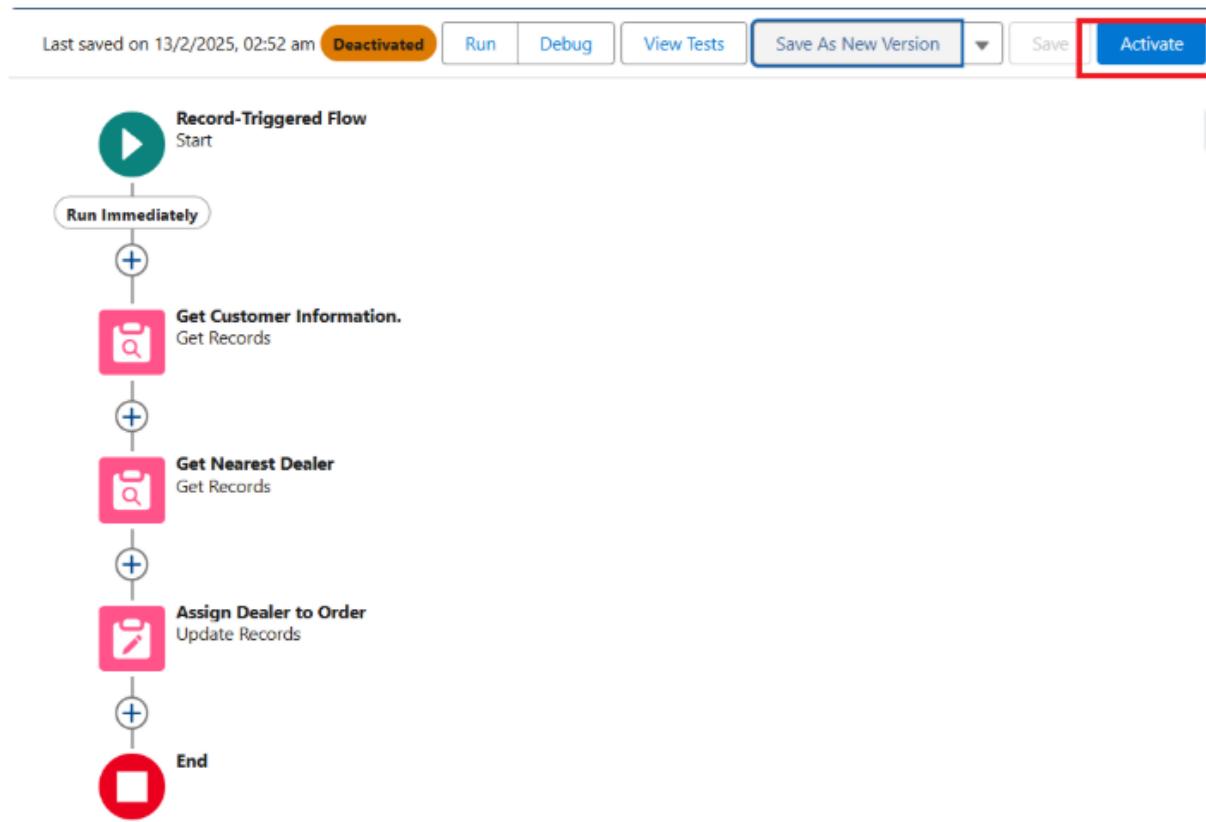
Show Advanced

Cancel      Save

- Use the IDs and all field values from a record or record collection
- Record to Update: {\$Record} (The triggering Vehicle\_Order\_\_c record)
- Field to Update:
  - Dealer\_\_c = {!Get\_Nearest\_Dealer.Id}

#### Step 4: Save & Activate Flow

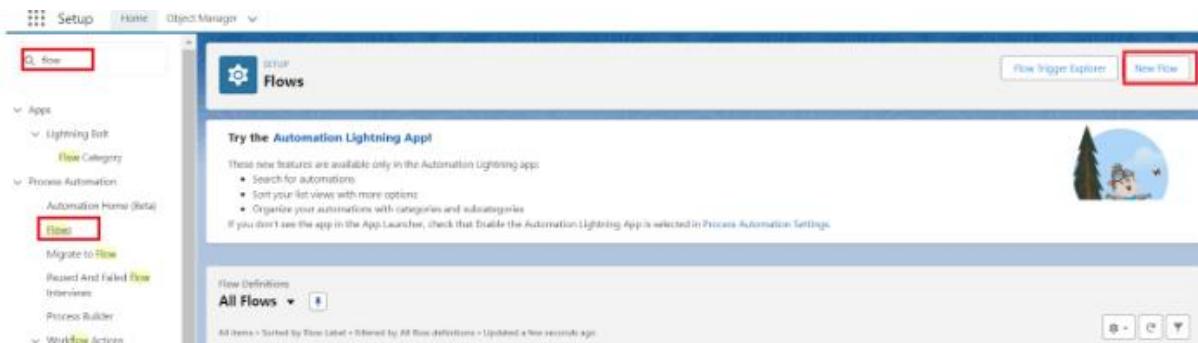
- Label: Auto Assign Dealer
- Click Save → Activate



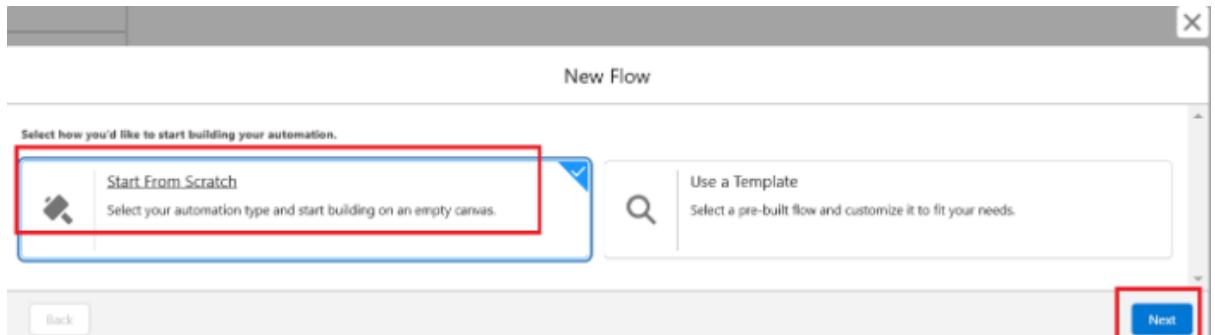
## Creating Record-Triggered Flow to Send an Email to the Customer Reminding About the Test Drive

### Step 1: Start Flow Setup

- Go to Setup → Quick Find → Type Flows → Click on Flows.



- Click New Flow.



- Choose Start from Scratch → Click Next.
- Select Record-Triggered Flow → Click Create.

## Step 2: Configure Trigger

- Object: Vehicle Test Drive
- Trigger the Flow When: A record is created or updated
- Set Entry Condition:
  - Condition Requirements: **All Conditions Are Met (AND)**
  - Field: Status\_\_c
  - Operator: Equals
  - Value: Scheduled

## Step 3: Add Scheduled Path

- Click + Add Scheduled Paths (below the trigger).
  - **Label:** Reminder Before Test Drive
  - **Time Source:** Test\_Drive\_Date\_\_c
  - **Offset Number:** 1
  - **Offset Option:** Days Before
- Click Done

## Step 4: Get Customer Information

- Click + → Select Get Records
  - **Label:** Get Customer Information
  - **Object:** Vehicle\_Customer\_\_c
  - **Filter Conditions:** Id = {!\$Record.Customer\_\_c}
  - **How Many Records to Store:** Only the first record

- **How to Store Record Data:** Automatically store all fields

### **Step 5: Send Email Reminder**

- Click + → Select Action
  - **Action Type:** Send Email
  - **Label:** Send Test Drive Reminder
  - **Subject:** "Reminder: Your Test Drive is Tomorrow!"
  - **Recipient Address List:** {!Get\_Customer\_Information.Email\_\_c}
  - **Rich-Text-Formatted Body:** True
  - **Body:** Create a Variable
    - **API Name:** EmailSent

### **Step 6: Save and Activate**

- **Label Name:** Test Drive Reminder
- Click Save
- Click Activate

## **Apex and Batch Class**

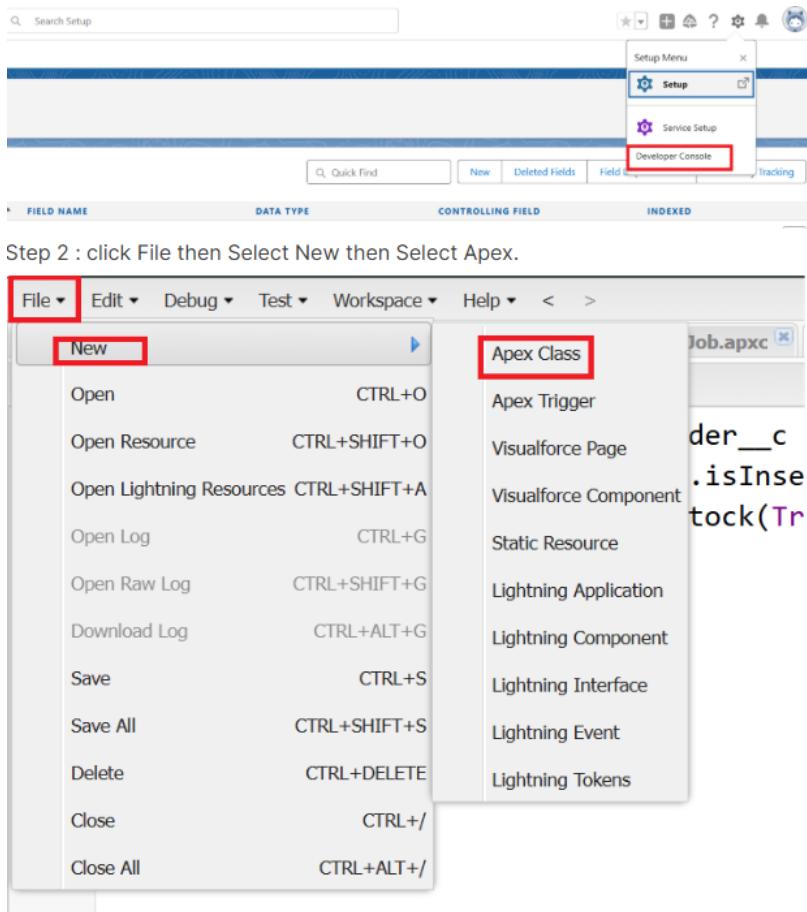
This part of the project uses Apex to ensure business logic such as stock checking and order confirmation is handled with code-level control and automation.

### **Step-by-Step: Apex Trigger and Batch Job**

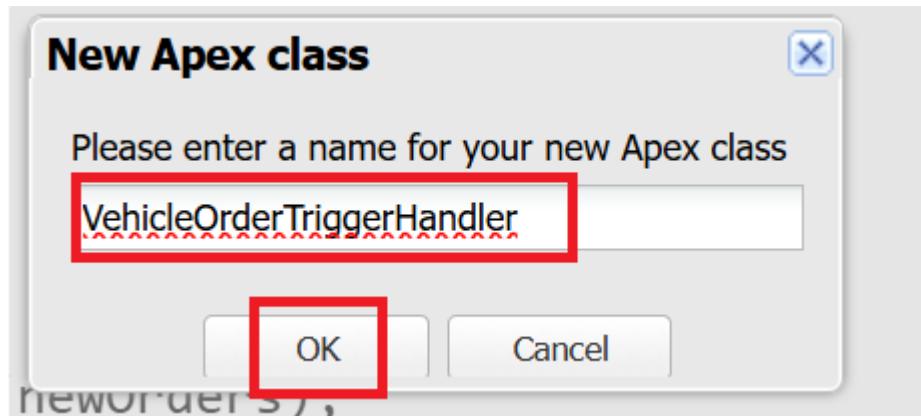
#### **1. Create Apex Class**

##### **Navigation:**

Go to Gear Icon → Developer Console → File → New → Apex Class



**Class Name:** VehicleOrderTriggerHandler



**Code:**

apex

CopyEdit

```
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) {
        if (isInsert || isUpdate) {
            preventOrderIfOutOfStock(newOrders);
        }
    }
    if (isAfter) {
        if (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()) {
        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) {
            if (vehicleStockMap.containsKey(order.Vehicle__c)) {

```

```

        Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
        if (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) {
    if (vehicleStockMap.containsKey(order.Vehicle__c)) {
        Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
        if (vehicle.Stock_Quantity__c > 0) {
            vehicle.Stock_Quantity__c -= 1;
            vehiclesToUpdate.add(vehicle);
        }
    }
}

```

```
        }
```

```
    }
```

```
}
```

```
}
```

```
if (!vehiclesToUpdate.isEmpty()) {
```

```
    update vehiclesToUpdate;
```

```
}
```

```
}
```

```
}
```

```
}
```

Vehicle Order  
VC-0003

\* = Required Information

Vehicle Order Name	Owner
VC-0003	Abdul Trainer
Vehicle Customer	
<input type="text" value="test KHIL 123456"/> <span>X</span>	
Vehicle	
<input type="text" value="Terwe"/> <span>X</span>	
Order Date	
<input type="text" value="21/02/2025"/> <span>▼</span>	
Status	
Pending	

We hit a snag.

Review the errors on this page.

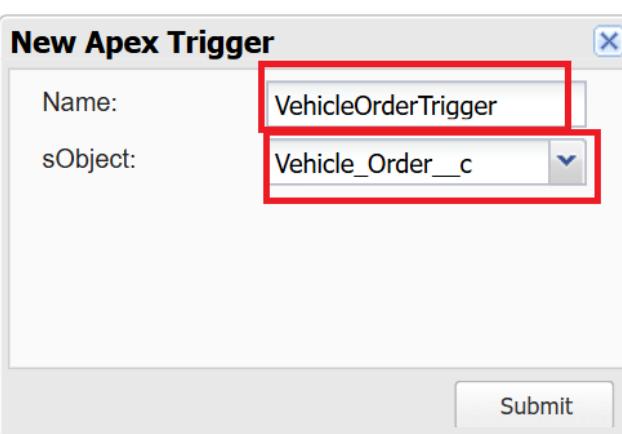
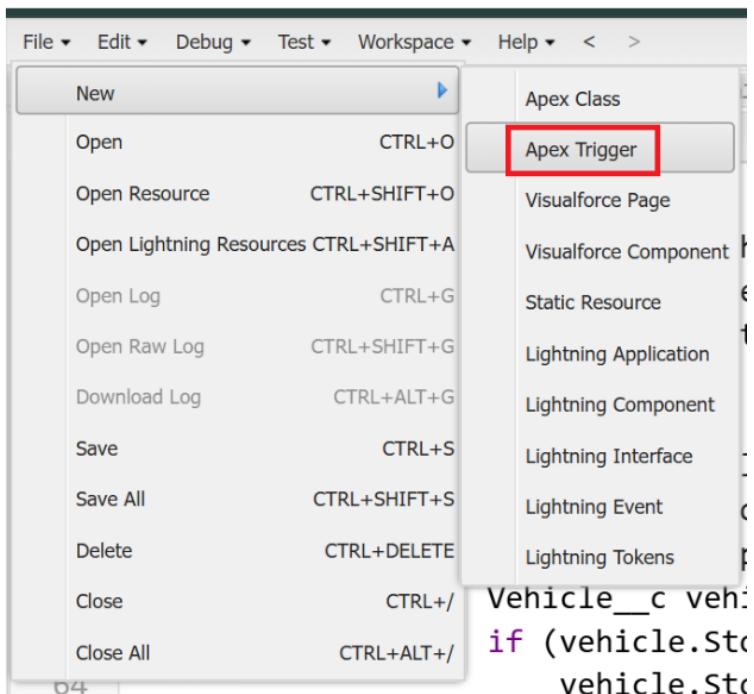
- This vehicle is out of stock. Order cannot be placed.

Cancel Save

## 2. Create Trigger on Vehicle Order

## **Trigger Name:** VehicleOrderTrigger

**Object:** Vehicle\_Order\_c



### Code:

apex

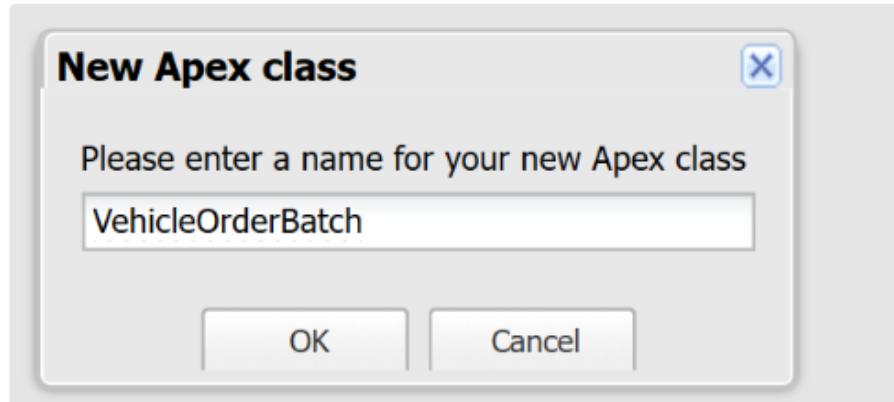
CopyEdit

```
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  
    );  
}
```

### 3. Create Batch Class

**Class Name:** VehicleOrderBatch



**Purpose:** Automatically updates pending orders when new stock is added.

**Code:**

apex

CopyEdit

```
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) {
    if (vehicleStockMap.containsKey(order.Vehicle__c)) {
        Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
        if (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.');
}

}

```

#### **4. Schedule the Batch Job**

**Scheduler Class Name:** VehicleOrderBatchScheduler

**Code:**

apex

CopyEdit

```

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

```

#### **5. Set a Schedule to Run the Job**

**Apex Code to Schedule the Batch Job:**

apex

CopyEdit

```

String cronExp = '0 0 0 * * ?'; // Every night at midnight
System.schedule('Daily Vehicle Order Processing', cronExp, new
VehicleOrderBatchScheduler());

```

To check where the schedule job is running:

Go to **Setup → Quick Find** → Type Scheduled Jobs.

# Future Scope of the Project

The current implementation of the WhatsNext Vision Motors Salesforce system has laid a strong foundation for managing vehicle sales, dealers, customers, and service requests. However, there is great potential to expand its functionality and deliver even more value to the business and customers in the future. Below are key areas for enhancement:

## 1. AI-Powered Dealer Recommendations

- Use Einstein AI to enhance the dealer assignment process by considering additional factors like stock availability, customer preference, and traffic data to assign the most optimal dealer.
- Predict customer buying behavior and recommend the best vehicle models based on interaction history.

## 2. Mobile App Integration

- Integrate Salesforce with a customer-facing mobile app where users can:
  - Browse available vehicles
  - Book test drives
  - Track service requests
  - Receive push notifications and offers

## 3. Chatbot for Customer Support

- Implement Salesforce Einstein Bots to handle customer queries related to vehicle availability, service schedules, and dealer locations in real-time, reducing load on human agents.

## 4. Advanced Reporting & Analytics

- Use Salesforce Reports and Dashboards to provide real-time KPIs for sales teams, dealership performance, service delivery times, etc.
- Introduce predictive dashboards to forecast sales trends and inventory needs.

## 5. Customer Loyalty Program

- Create an integrated loyalty program where customers earn points for:

- Booking services
- Referring friends
- Writing reviews
- Points can be redeemed for discounts or accessories.

## 6. Automated Feedback Collection

- After each test drive or service, automatically send feedback forms and store responses in Salesforce for service quality analysis and improvement.

## 7. IoT Integration

- Integrate vehicle data using IoT to automatically log:
  - Mileage
  - Maintenance needs
  - Performance statistics
- Allow service requests to be generated automatically based on usage data.

## 8. Marketing Cloud Integration

- Use Salesforce Marketing Cloud to run targeted marketing campaigns via:
  - Email
  - SMS
  - Social media
- Personalize promotions based on customer interests and past purchases.

## 9. Enhanced Service Request Management

- Implement case escalation rules and SLA monitoring to ensure all service requests are handled within defined timelines with priority levels.

## 10. Global Expansion Readiness

- Adapt the app for multi-language and multi-currency support for future expansion to international markets.

## Conclusion

The Salesforce-based solution implemented for WhatsNext Vision Motors marks a significant step forward in modernizing the automotive sales and customer service experience. By integrating objects such as Vehicle, Dealer, Customer, and Service Requests, the project successfully streamlines the vehicle purchase journey from test drives to service management.

The system not only automates essential business processes—like dealer assignment, test drive reminders, and stock tracking—but also enhances customer satisfaction through timely communication and structured data handling.

Through the use of record-triggered flows, lookup relationships, and custom objects and fields, this solution showcases the power of the Salesforce platform in solving real-world business challenges efficiently.

Furthermore, the project lays a strong foundation for future innovations such as AI integration, mobile apps, and IoT-connected vehicles. With continuous improvements and expansions, the platform can evolve into a comprehensive digital ecosystem for the automotive industry.

In summary, this project demonstrates the effectiveness of Salesforce as a CRM tool and prepares the business for scalable, customer-centric growth.

**THANK YOU**