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.

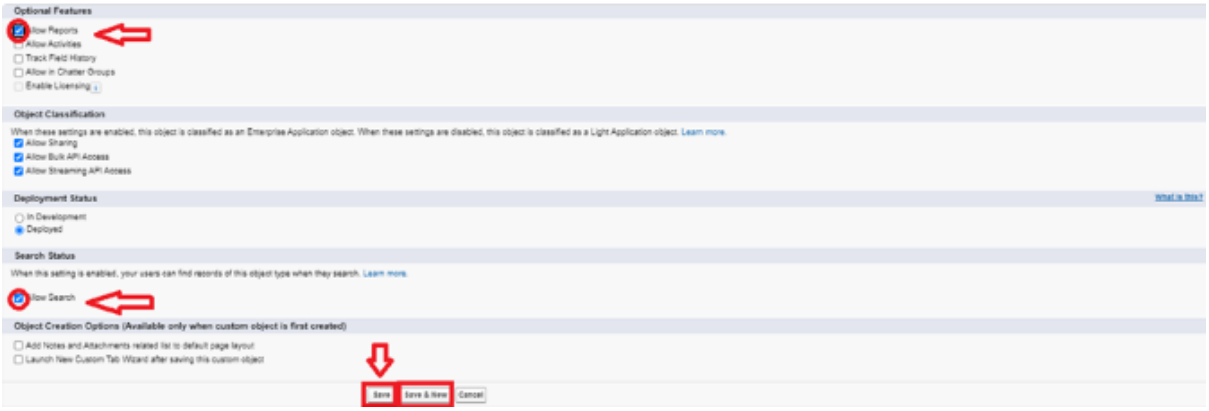
6. **Improve Customer Experience:**
   To create a more transparent and responsive order tracking system for customers using Salesforce Dashboards and Reports.
7. **Leverage Low-Code Tools:**
   To apply Salesforce's low-code solutions (like Process Builder and Flow Builder) to solve complex business 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.

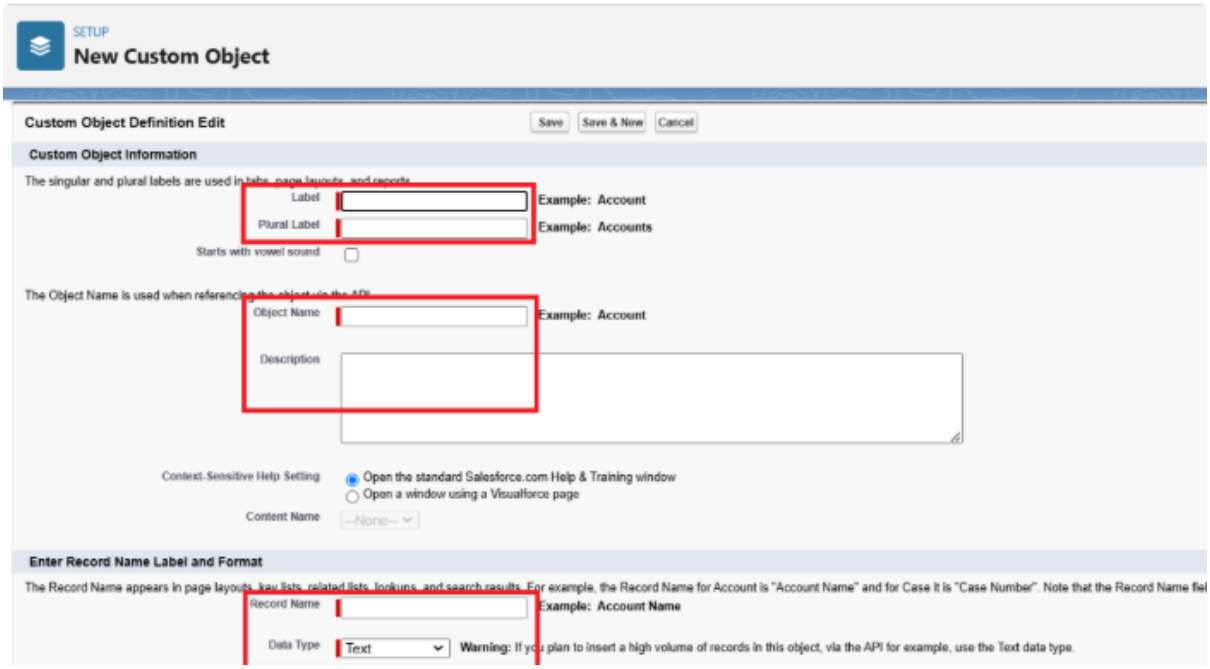
Data Management-Objects

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.





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.

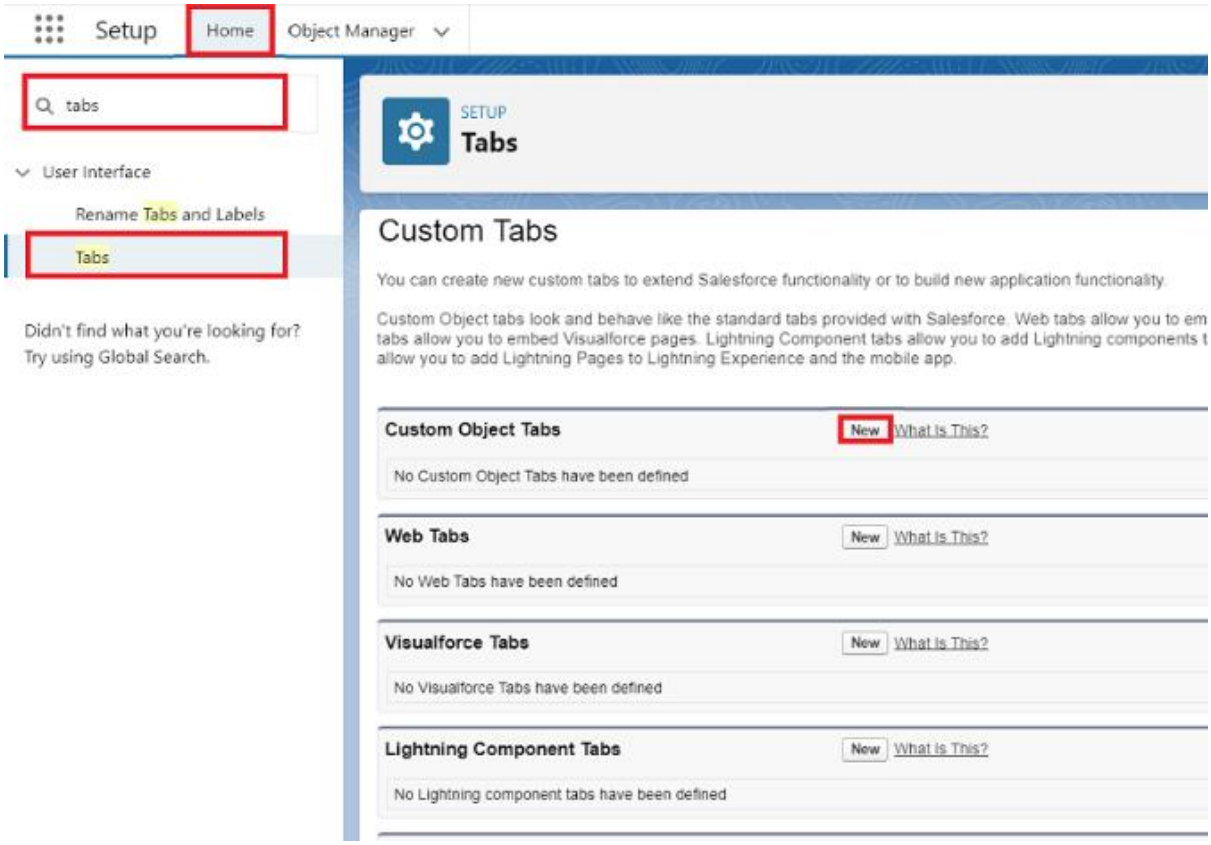
Data Management – Object Tab Creation

Task:

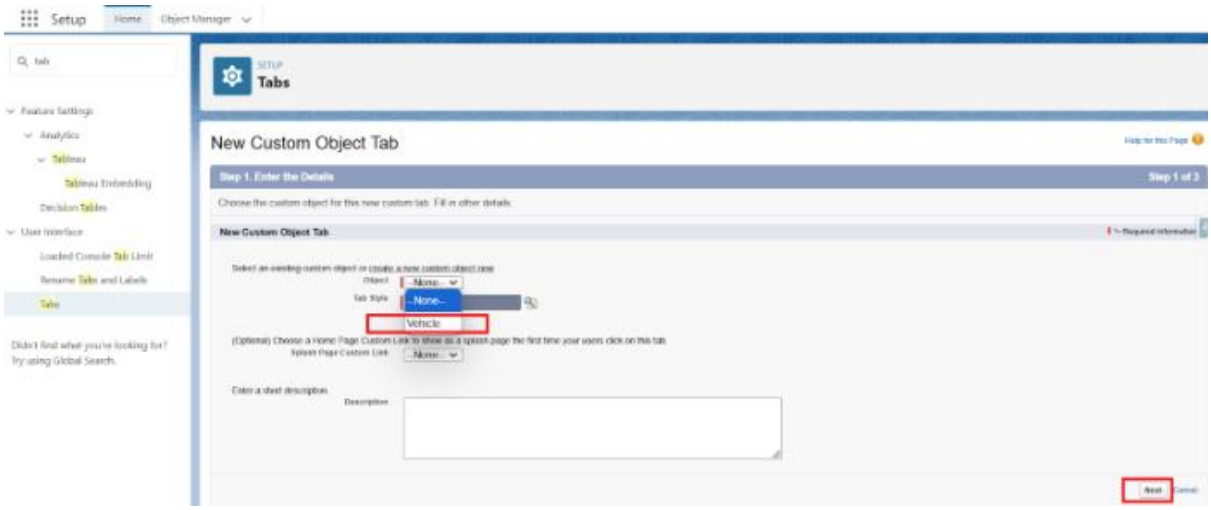
Creating a Custom Tab for the Vehicle__c object to make it visible and accessible in the app.

🛠️ Steps:

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



3. Select Object: Vehicle__c → choose any Tab Style → click Next.



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.

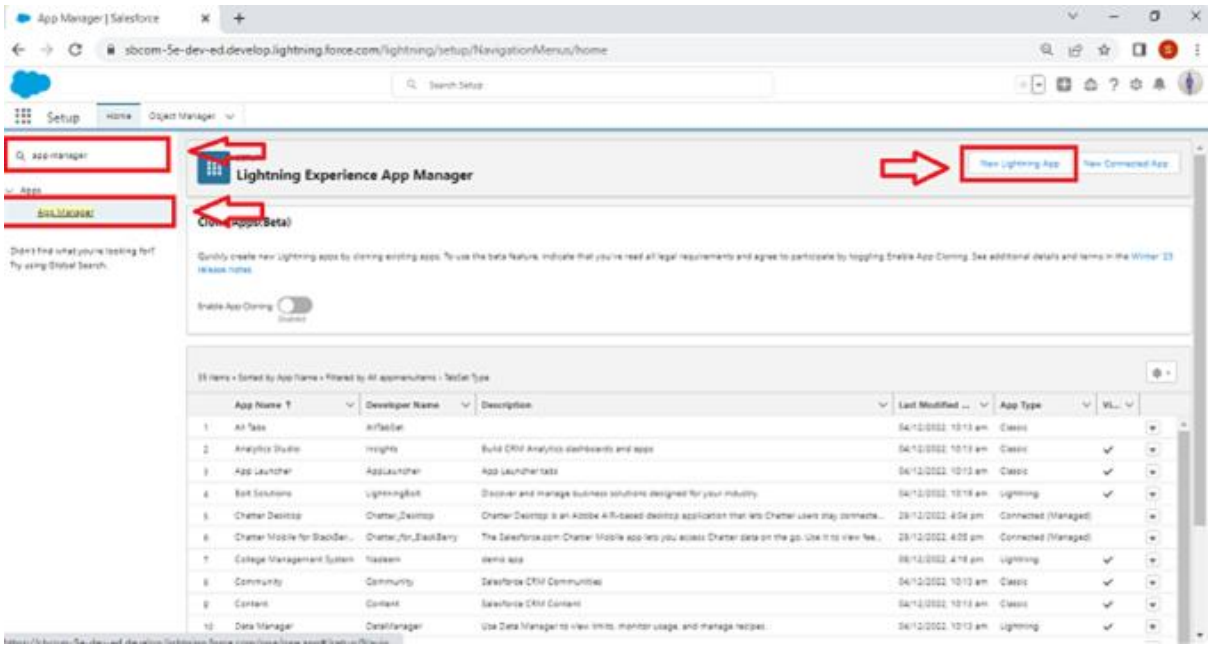
Data Management – App Manager

Task:

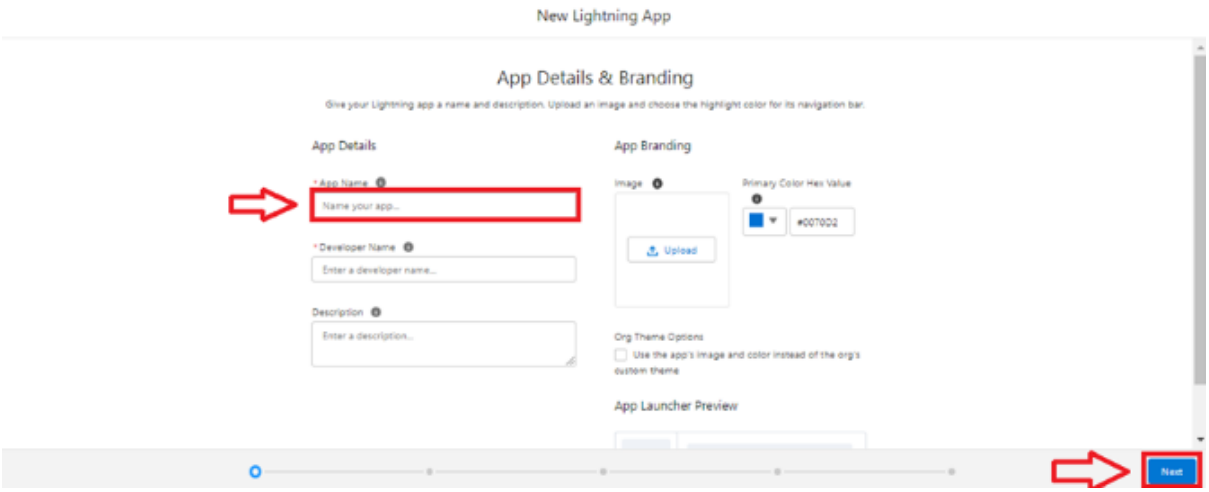
Create a Lightning App named *WhatNext Vision Motors* to organize and navigate between custom objects easily.

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.



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



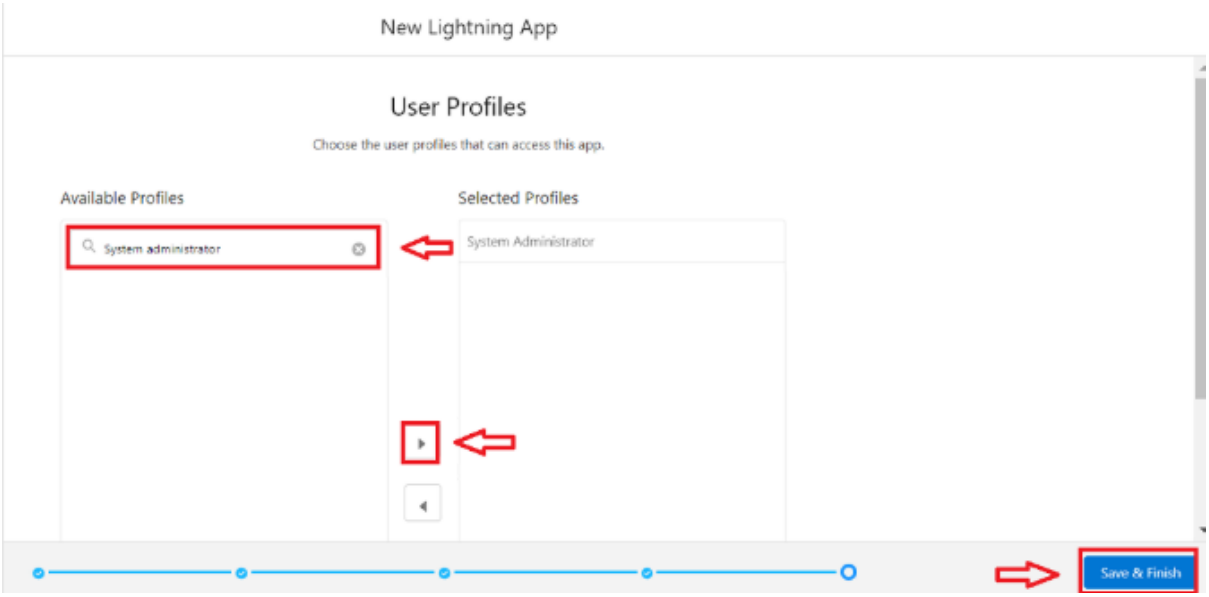
4. Utility Items: Skip or leave default → Click Next

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



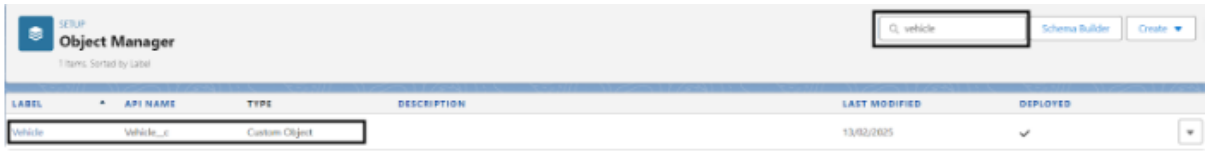
Data Management – Fields & Relationships

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.

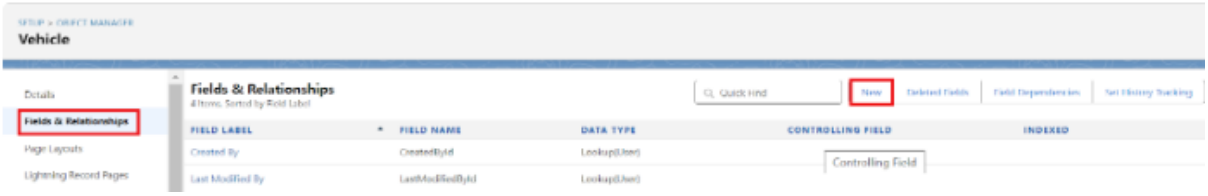
Creating a Field in Vehicle__c Object

Steps:

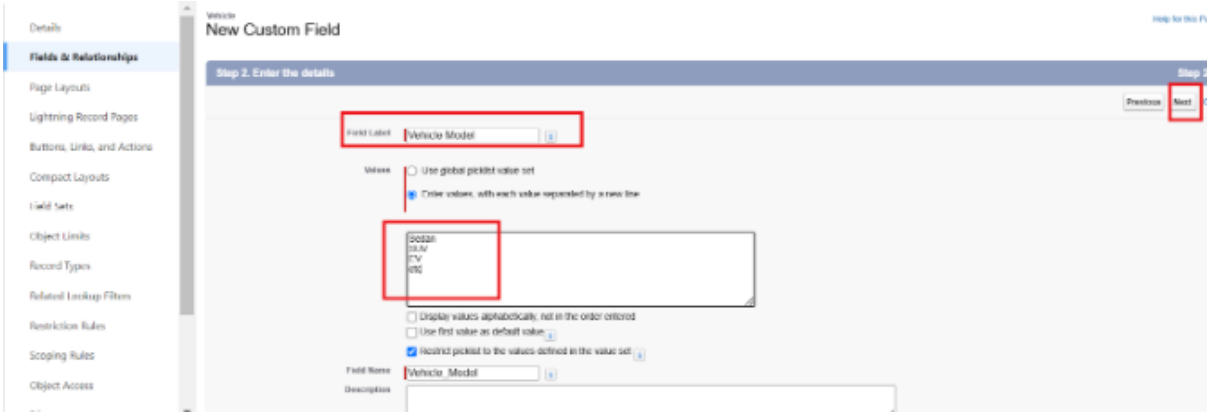
1. **Go to Setup → Object Manager → search Vehicle → click on it.**



2. **Click Fields & Relationships → New.**



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



4. **Fill:**
    - **Field Label: Vehicle Model**
    - **Picklist Values: Sedan, SUV, EV, etc.**
5. **Click Next → Next → Save & New.**

2️⃣ 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.**

3️⃣ 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.**

4️⃣ 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.**

5️⃣ 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.**

6️⃣ Creating a Field in Vehicle_Service_Request__c Object

Steps:

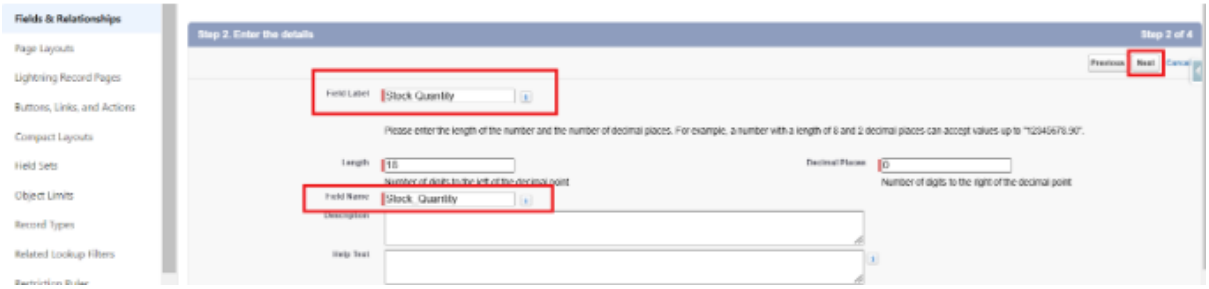
1. **Go to Setup → Object Manager → open Vehicle_Service_Request.**
2. **Click Fields & Relationships → New.**
3. **Select Data Type: *Picklist* → Next.**
4. **Fill:**
    - **Field Label: Status**
    - **Picklist Values: Requested, In Progress, Completed**
5. **Click Next → Next → Save.**

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.**

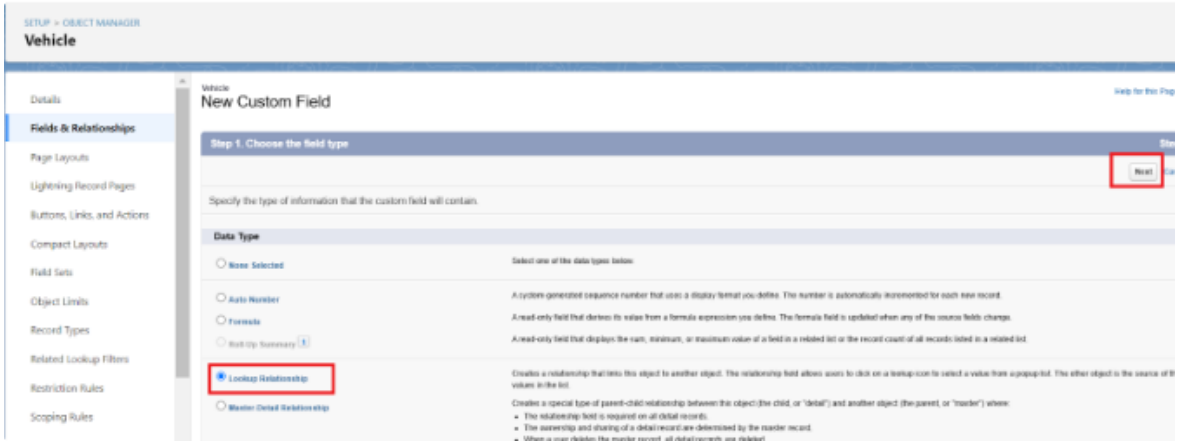


3. **Fill in the details:**
    - **Field Label: Stock Quantity**
4. **Click Next → Next → Save & New (if adding more fields).**

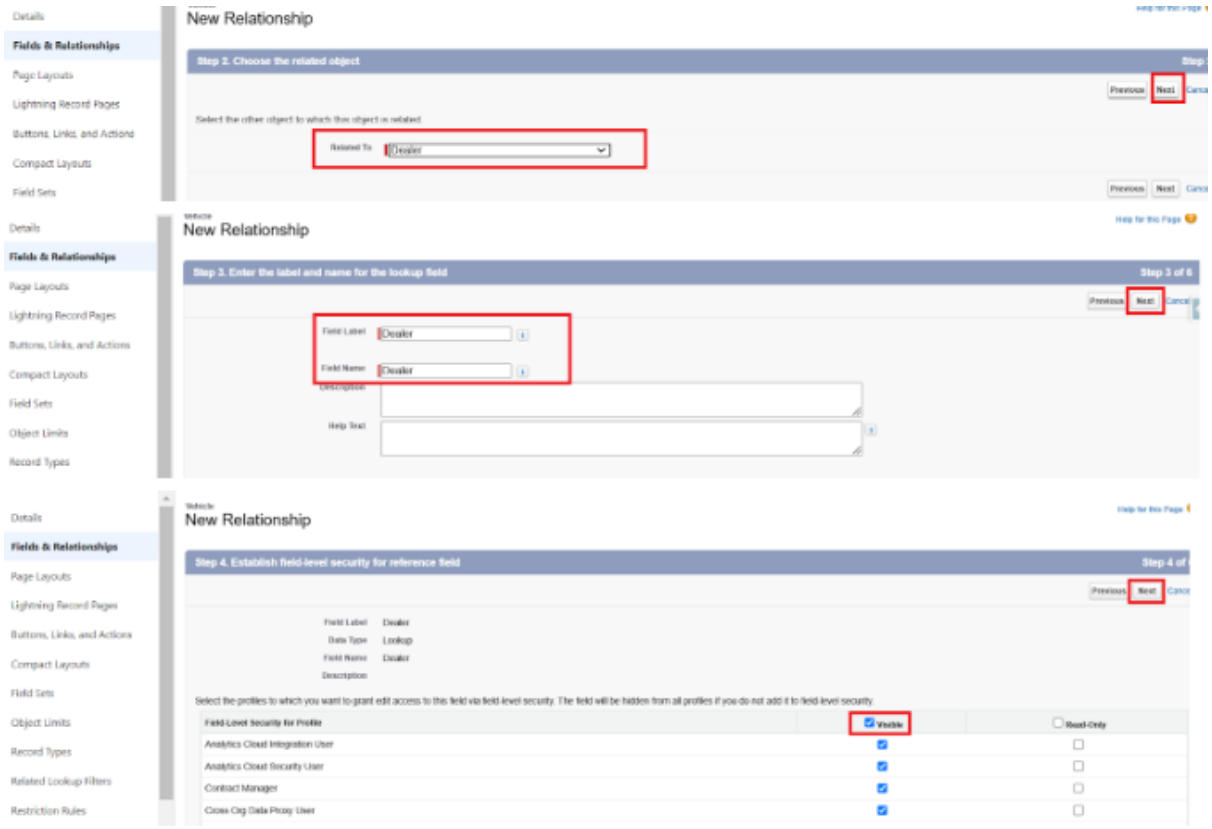
Creating a Lookup Relationship in Vehicle__c to Dealer__c

Steps:

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



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



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

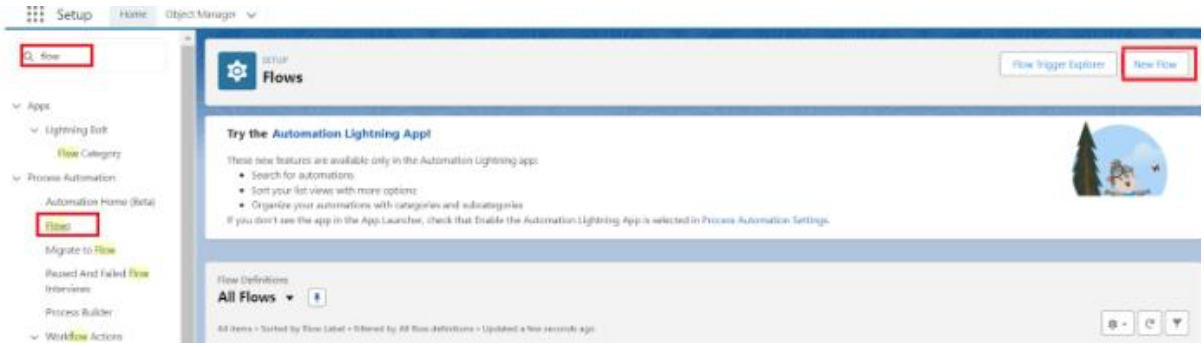
Create all other fields and relationship like same

Automation

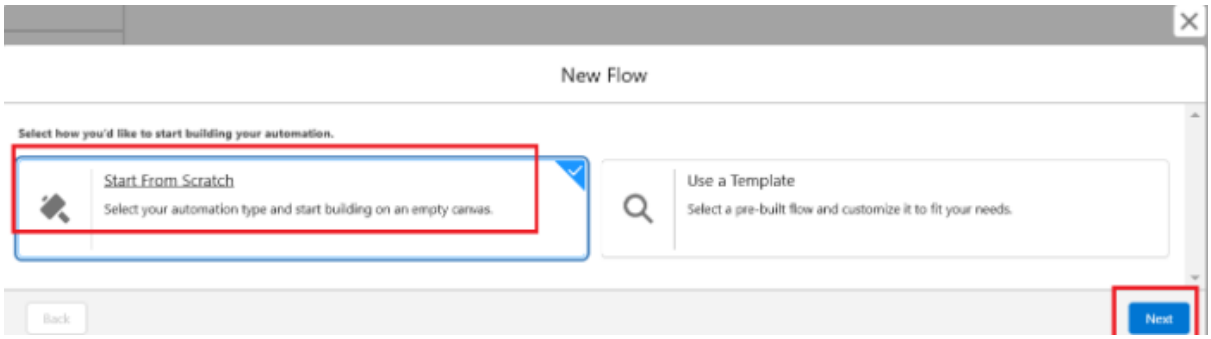
This automation ensures that when a new Vehicle Order is created with status "Pending", the system will auto-assign the nearest Vehicle Dealer based on the customer's location.

Steps to Create Record-Triggered Flow

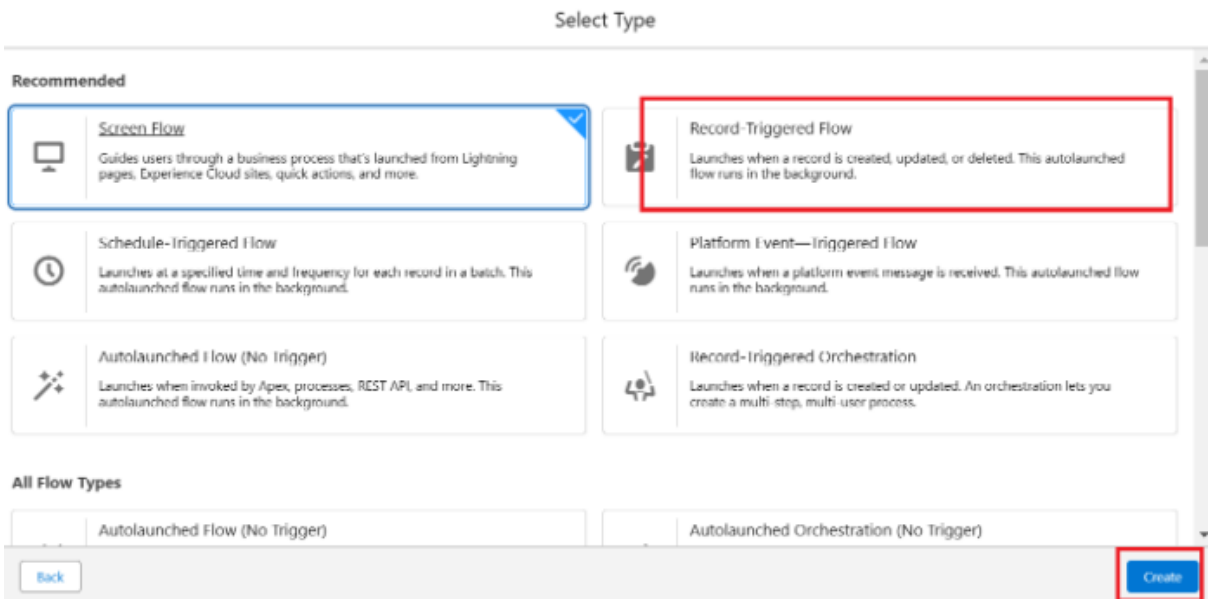
1. Go to Flows
    - In Quick Find, type Flows → Click New Flow



2. Start From Scratch
    - Select "Start from Scratch" → Click Next



3. Choose Flow Type
    - Select Record-Triggered Flow → Click Create



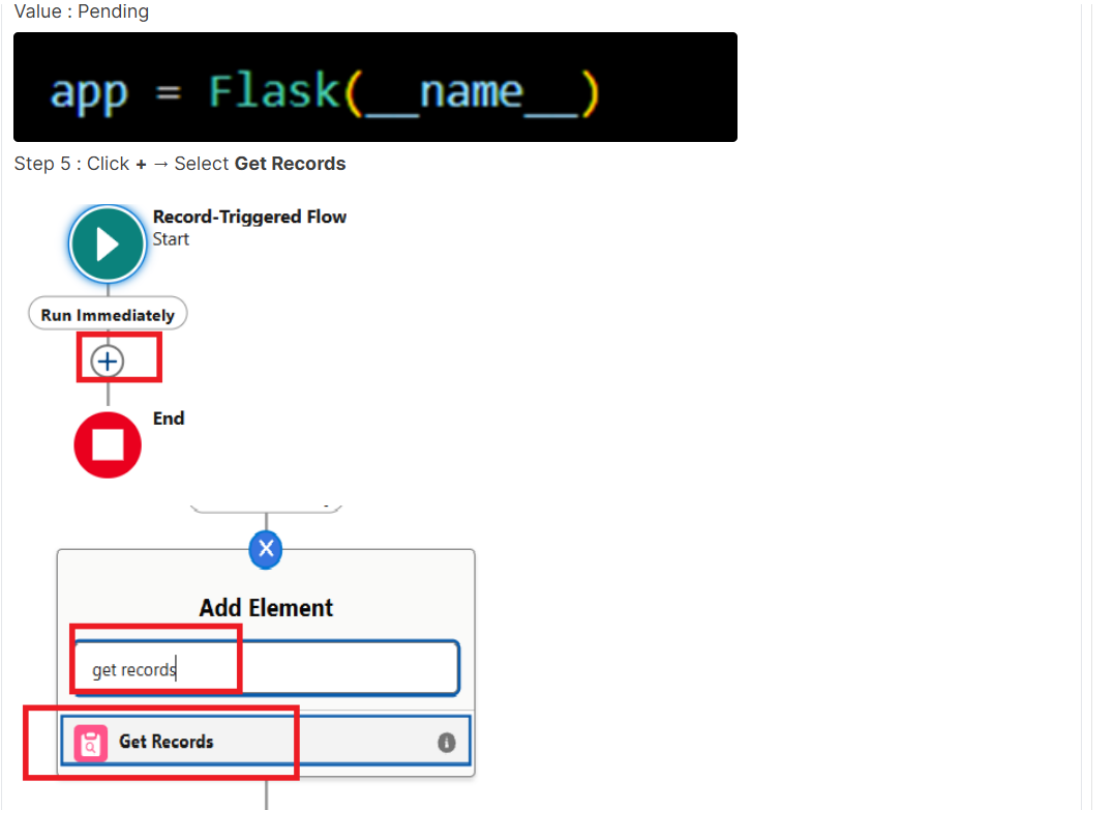
Trigger Configuration

- Object: Vehicle_Order__c
- Trigger the Flow When: Record is Created

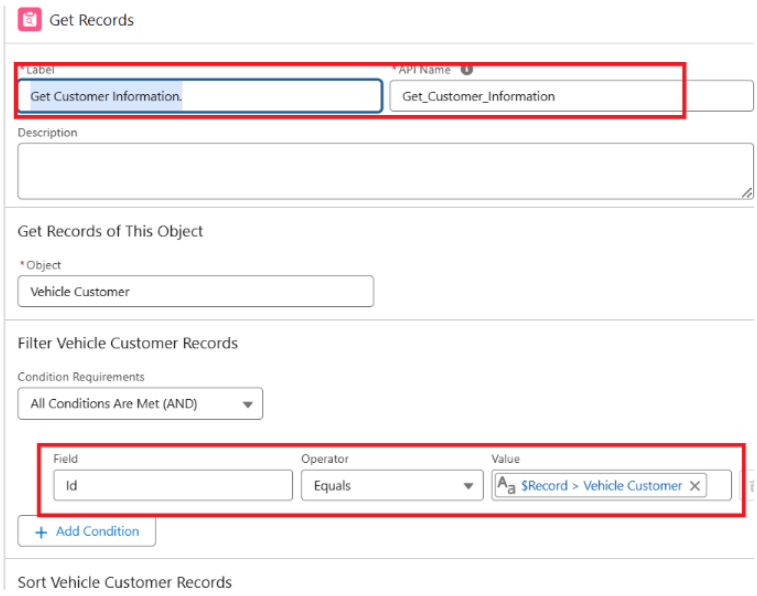
- Condition Requirements:
    - All Conditions Are Met (AND)
        - Status__c Equals Pending

Step 1: Get Customer Information

- Click + → Get Records

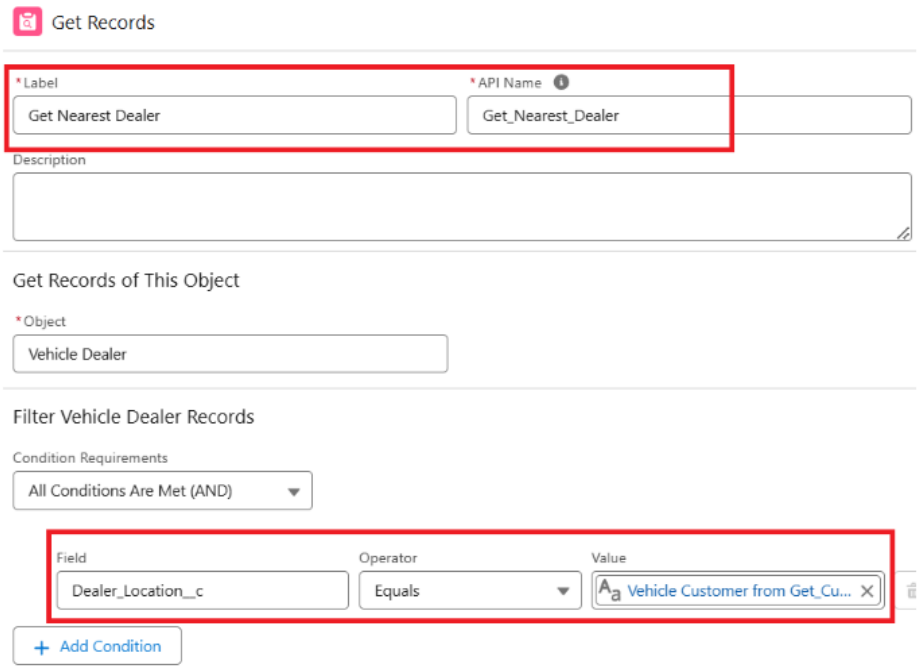


- Label: Get Customer Information



- Object: Vehicle_Customer__c
- Condition:
    - Id Equals {$Record.Vehicle_Customer__c}
- Store Only the first record
- Automatically store all fields

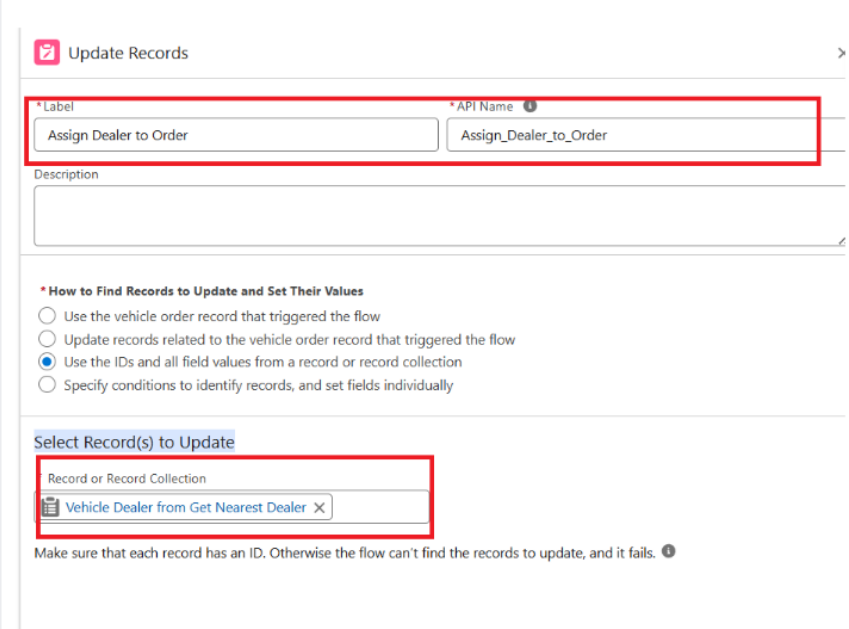
Step 2: Get Nearest Dealer



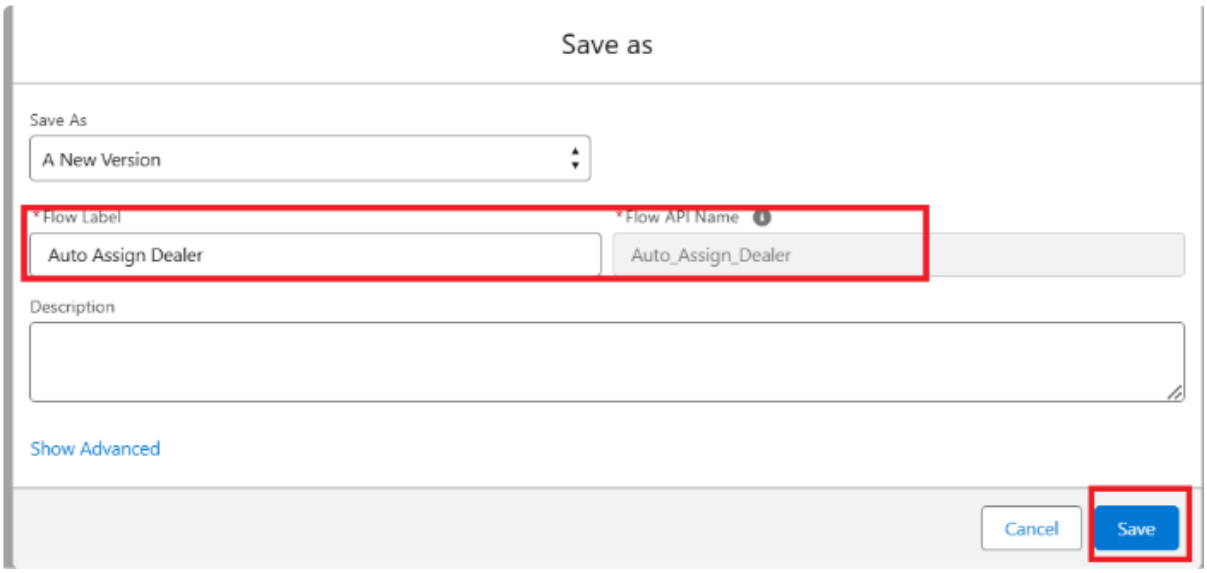
- Click + → Get Records
- Label: Get Nearest Dealer
- Object: Vehicle_Dealer__c
- Condition:
    - Dealer_Location__c Equals {!Get_Customer_Information.Address__c}
- Store Only the first record
- Automatically store all fields

Step 3: Update the Order Record

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



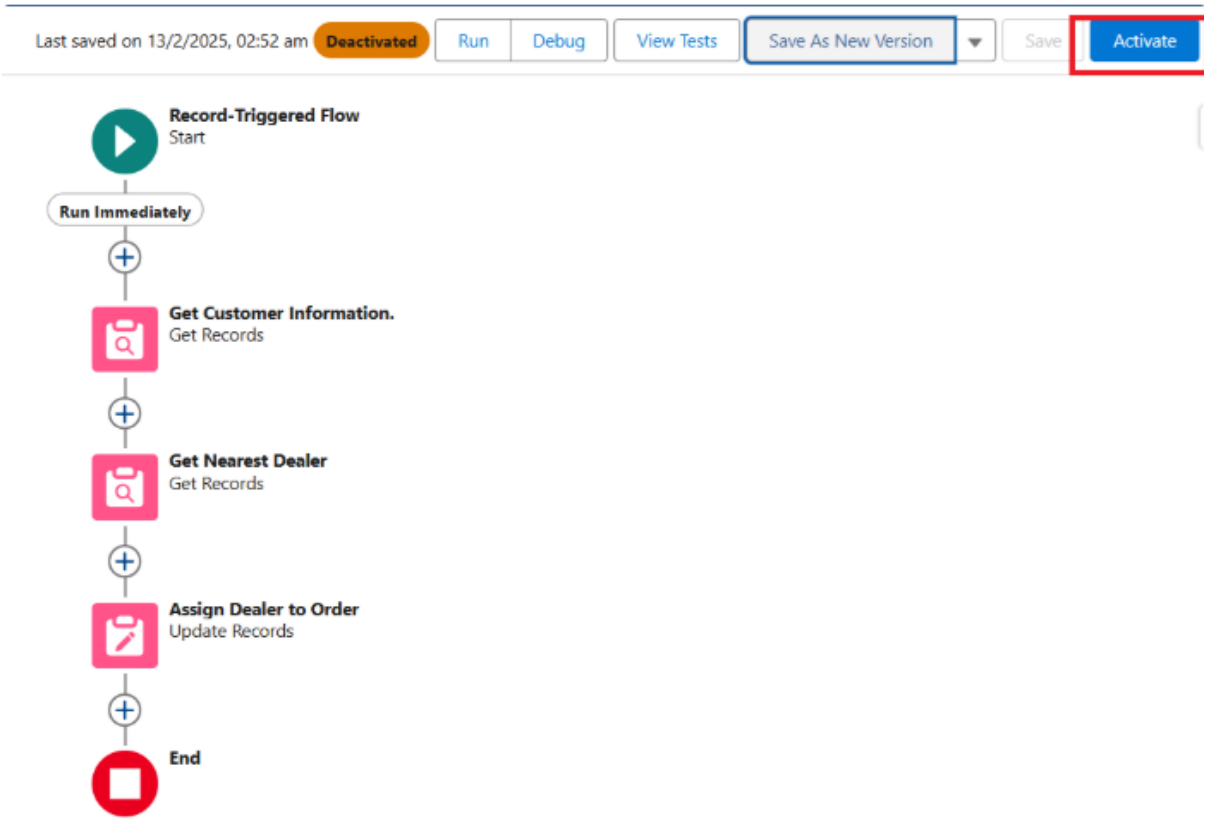
- How to Find Records:



    - 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

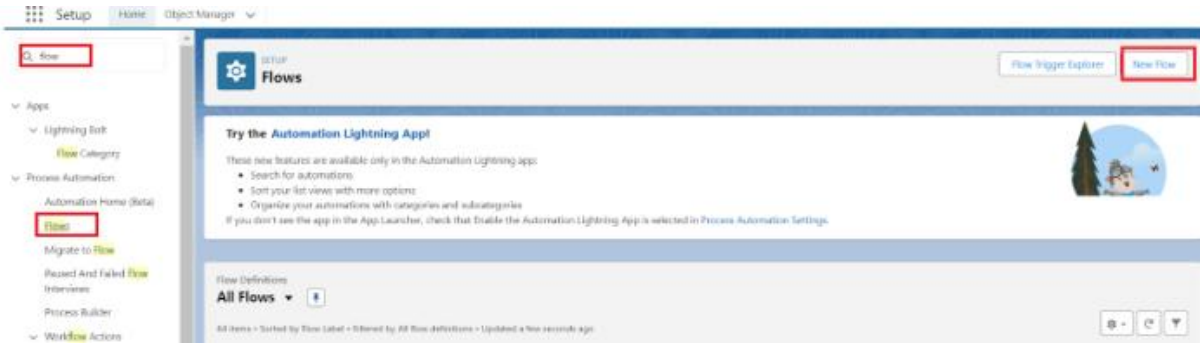
- Label: Auto Assign Dealer
- Click Save → Activate



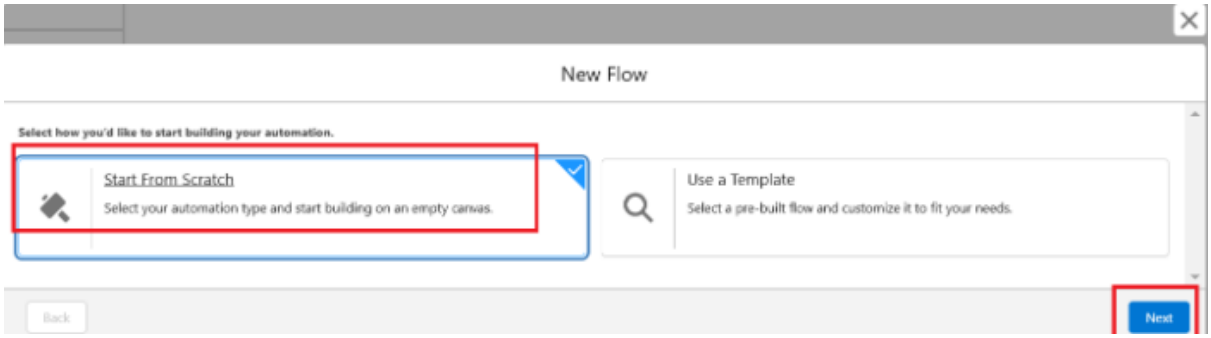
Creating Record-Triggered Flow to Send an Email to the Customer Reminding About the Test Drive

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

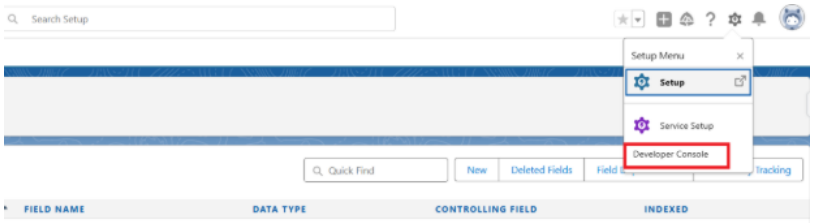
Apex and Batch Class

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.

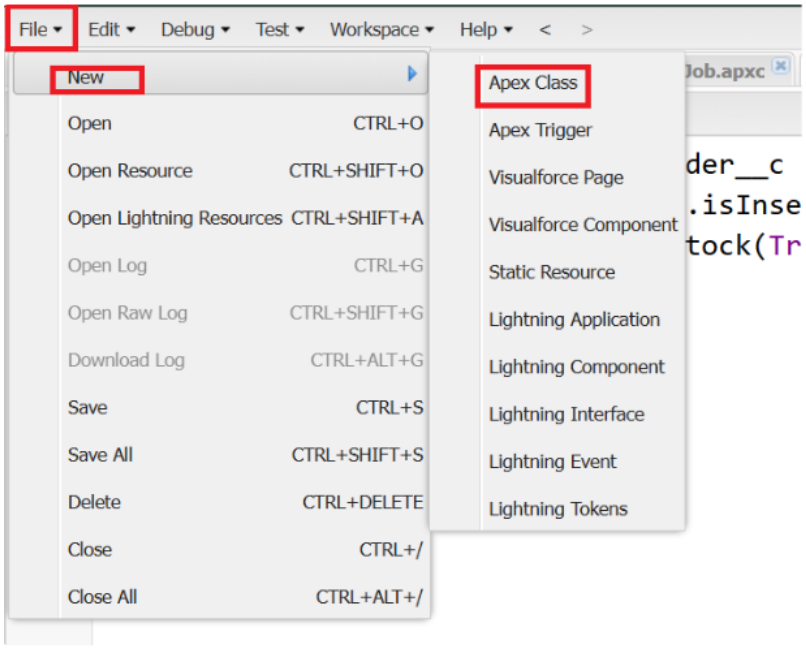
Step-by-Step: Apex Trigger and Batch Job

1. Create Apex Class

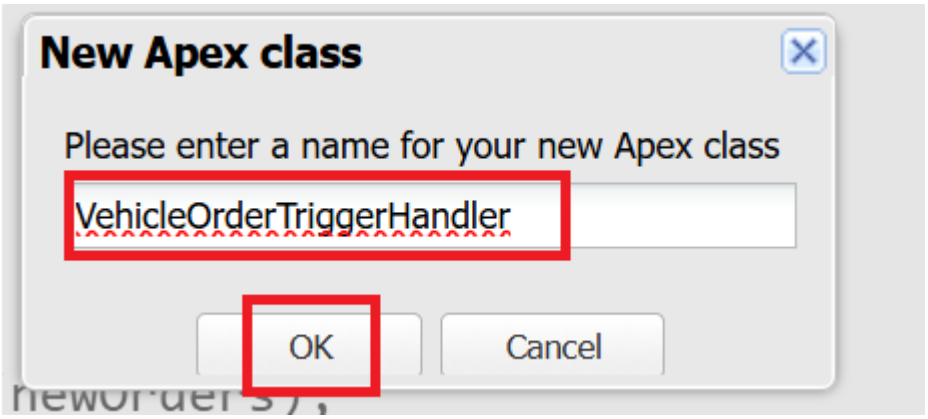
Navigation:
Go to Gear Icon → Developer Console → File → New → Apex Class



Step 2 : click File then Select New then Select Apex.



Class Name: VehicleOrderTriggerHandler



Code:

apex

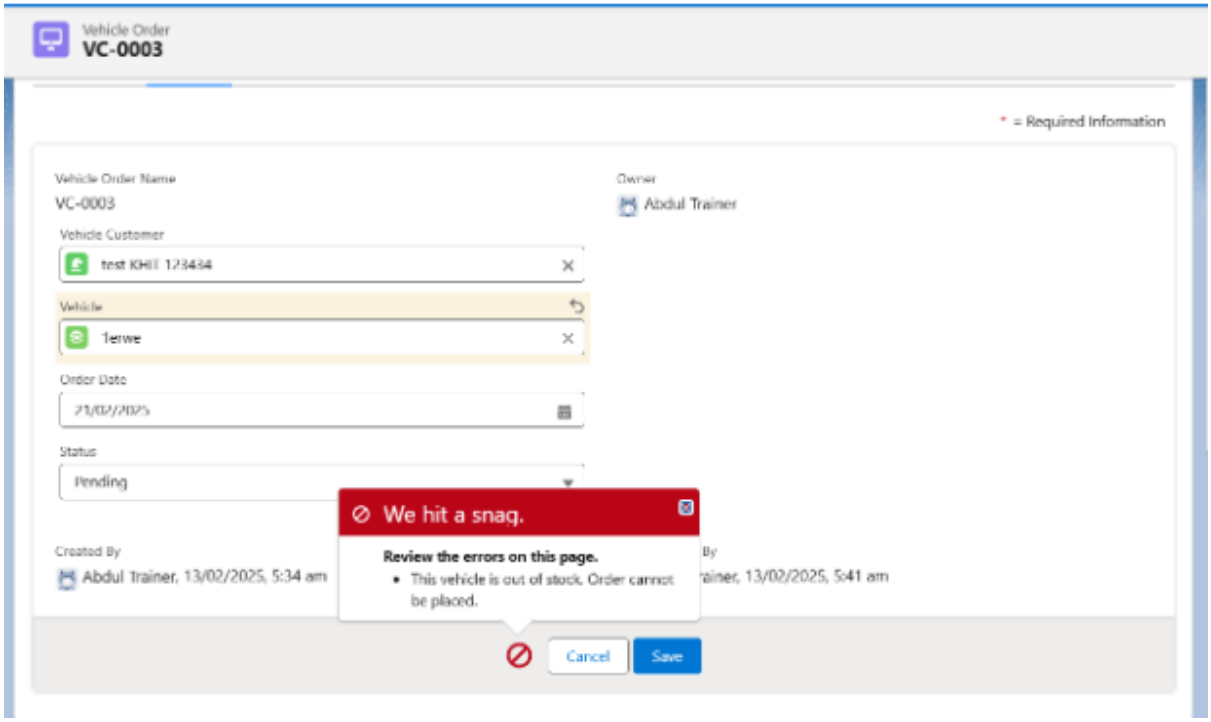
CopyEdit

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)) {
```

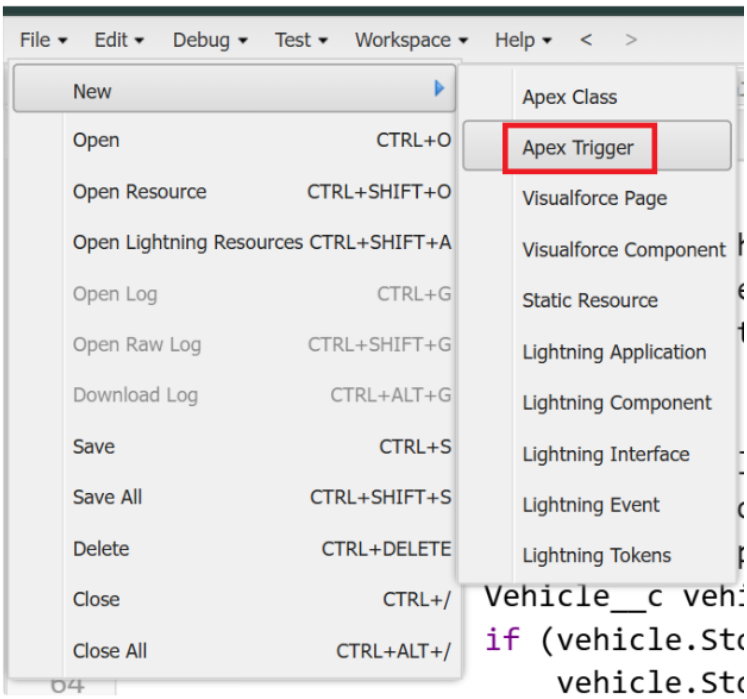
```
            Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
            if (vehicle.Stock_Quantity__c <= 0) {
                order.addError('This vehicle is out of stock. Order cannot be placed.');
            }
        }
    }
}

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);
```

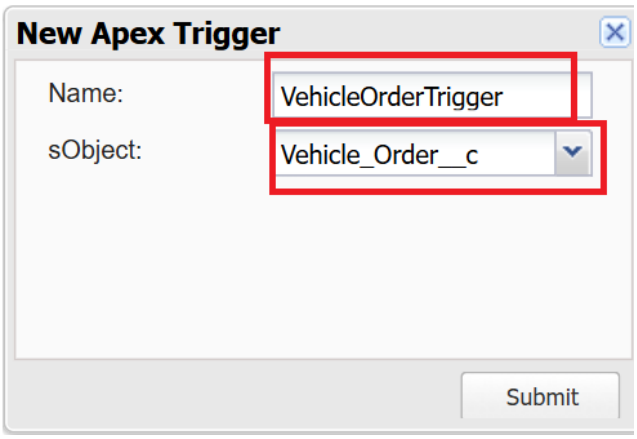
```
                }
            }
        }

        if (!vehiclesToUpdate.isEmpty()) {
            update vehiclesToUpdate;
        }
    }
}
}
```



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(

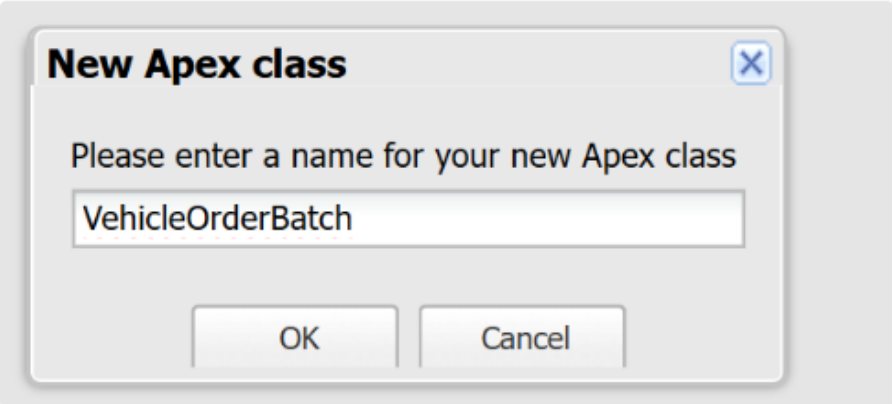
Trigger.new, Trigger.oldMap, Trigger.isBefore, Trigger.isAfter, Trigger.isInsert, Trigger.isUpdate

);

}

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