

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

ABSTRACT

WhatNext Vision Motors is enhancing both its customer service capabilities and internal operational efficiency through the implementation of a robust Salesforce CRM ecosystem. The system optimizes the vehicle purchasing workflow by automatically assigning customer orders to the nearest dealership based on geographic proximity and preventing transactions for units that are no longer in stock. Automated processes maintain accurate, real-time order status updates and generate reminder emails prior to scheduled test-drive appointments. The platform's functionality is supported by Apex triggers to enforce inventory rules, Batch Apex for routine stock validation, and Scheduled Apex for autonomous processing of pending orders. Collectively, these advancements improve customer satisfaction, minimize ordering inaccuracies, and streamline day-to-day dealership operations.

OBJECTIVES

This capstone project aims to fully digitize the entire vehicle order and service management workflow by integrating automation across all operational stages. It seeks to assign customers to the nearest eligible dealer based on address using Salesforce Flows, while preventing orders for vehicles that are no longer available through Apex triggers. To ensure real-time accuracy in inventory records, the system continuously monitors and updates stock availability using batch and scheduled Apex jobs. The platform also streamlines service scheduling and sends automated email reminders for test drives and maintenance appointments. Additionally, the use of validation rules reduces manual data entry and promotes data accuracy, while automated status notifications enhance customer engagement throughout the ordering and servicing lifecycle. Overall, the system is designed with scalability in mind to support future business expansion and potential third-party integrations.

TECHNOLOGY STACK

- **Lightning App Builder:** A drag-and-drop tool for creating custom apps and user interfaces. It allows the addition of pages, components, and tabs to deliver a tailored experience for different users.
- **Salesforce CRM Platform:** The cloud-based foundation for the application, supporting object creation, automation, and interface configuration.
- **Flow Builder:** Enables no-code automation for various business processes, including:
 - **Scheduled Flows:** Run at specific times to automate tasks like stock monitoring and status updates.
 - **Record-Triggered Flows:** Automatically triggered when a record is created or updated, used for dealer assignment, status updates, and sending emails.
- **Apex Classes and Triggers:** Implements advanced logic and enforces business rules during record operations, such as vehicle stock validation.
- **Scheduled Apex:** Executes automated jobs at defined intervals, for example, sending order updates or summaries daily.
- **Batch Apex:** Handles large data volumes in batches to maintain accurate stock levels and vehicle availability.
- **Email Alerts and Notifications:** Configured through Flow Builder to automatically send reminders, confirmations, and updates to customers and dealers, ensuring consistent communication.

SYSTEM REQUIREMENTS & FUNCTIONAL HIGHLIGHTS

Salesforce CRM Core Setup

- Maintain comprehensive and up-to-date records of all vehicles, inventory levels, and dealership branch information.
- Efficiently document and monitor customer transactions, scheduled test drives, and service-related inquiries.
- Automatically assign customer orders to the dealership geographically closest to the customer's location.

Business Process Automation

- Prevent order submissions when the selected vehicle is out of stock.
- Initiate automated dealership assignment based on the customer's address or location data.
- Dispatch scheduled email notifications to provide timely reminders for upcoming test-drive appointments.

Apex Development & Trigger Logic

- Utilize Apex triggers to enforce critical business rules, including inventory validation and automated order routing.
- Adopt a structured trigger handler architecture to promote clean, scalable, and easily maintainable code.

Batch & Scheduled Processes

- Employ Batch Apex to regularly monitor inventory levels and update vehicle availability records.
- Utilize Scheduled Apex to automate routine stock updates and facilitate order-related notifications.

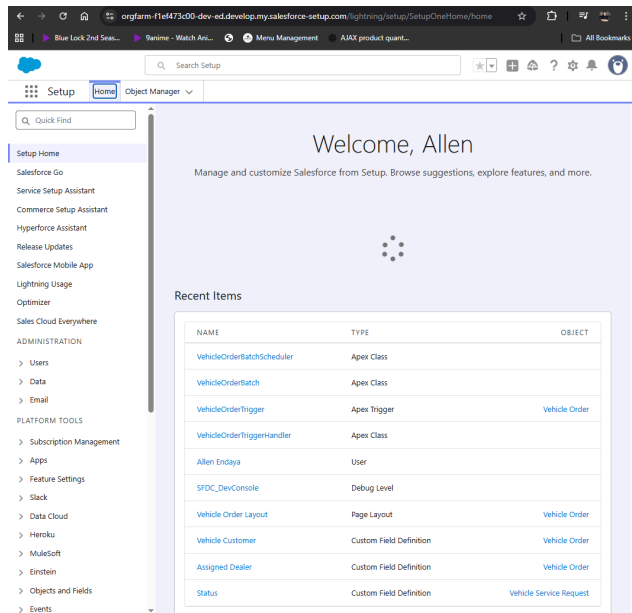
What you'll learn:

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

DETAILED EXECUTION OF PROJECT PHASES

Phase 1: Developer Org Setup

- Created a free Salesforce Developer Org from <https://developer.salesforce.com/signup>
- Verified the account via email and logged in.
- Enabled Developer Console and activated Lightning Experience.
- Personalized the org by enabling necessary features for custom object and app creation.



Phase 2: Requirement Gathering & Planning

- Defined project scope based on real-world vehicle service processes.
- Identified major entities: Customer, Vehicle, Dealer, Order, and Test Drive.
- Outlined automation goals: automatic dealer assignment, stock validation, and email reminders.

Phase 3: Custom Object Creation

- Created six main custom objects:
 - Vehicle__c – Name, Type, Price, Stock Quantity
 - Vehicle_Customer__c – Name, Email, Phone, Address
 - Vehicle_Dealer__c – Name, Location, Assigned Vehicles
 - Vehicle_Order__c – Vehicle (lookup), Customer (lookup), Dealer (lookup), Status, Date
 - Vehicle_Service_Request__c – Customer (lookup), Vehicle (lookup), Request Date, Description, Status
 - Vehicle_Test_Drive__c – Customer (lookup), Vehicle (lookup), Preferred Date
- Configured relationships using Lookup fields.

The screenshot shows the Salesforce Setup interface. The browser address bar displays the URL: `orgfarm-f1ef473c00-dev-ed.develop.my.salesforce-setup.com/lightning/setup/ObjectManager/home`. The page title is "Object Manager" with a sub-header "6 Items, Sorted by Label". A search bar contains the text "vehicle". Below the search bar is a table listing six custom objects.

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Vehicle	Vehicle__c	Custom Object		11/20/2025	✓
Vehicle Customer	Vehicle_Customer__c	Custom Object		11/20/2025	✓
Vehicle Dealer	Vehicle_Dealer__c	Custom Object		11/20/2025	✓
Vehicle Order	Vehicle_Order__c	Custom Object		11/20/2025	✓
Vehicle Service Request	Vehicle_Service_Request__c	Custom Object		11/20/2025	✓
Vehicle Test Drive	Vehicle_Test_Drive__c	Custom Object		11/20/2025	✓

Phase 4: Data Management-Table

← → ↺ 🏠

orgfarm-f1ef473c00-dev-ed.develop.my.salesforce-setup.com/lightning/setup/CustomTabs/page?a...

☆

📄


⬇

☰

👤

⋮

📖 | ▶ Blue Lock 2nd Seas... ▶ 9anime - Watch Ani... ⌂ 🗂 Menu Management ⚙️ AJAX product quant... | 📁 All Bookmarks



🔍 Search Setup

★ + 🗂 ? ⚙️ 🔔 🧑

⋮ Setup

Home

Object Manager ▾


🔍 tabs

▼ User Interface

Rename **Tabs** and Labels

Tabs

Didn't find what you're looking for? Try using Global Search.

 **SETUP**

Tabs







Custom Tabs

Help for this Page ?

You can create new custom tabs to extend Salesforce functionality or to build new application functionality.

Custom Object tabs look and behave like the standard tabs provided with Salesforce. Web tabs allow you to embed external web applications and content within the Salesforce window. Visualforce tabs allow you to embed Visualforce pages. Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app. Lightning Page tabs allow you to add Lightning Pages to Lightning Experience and the mobile app.

Custom Object Tabs New What Is This?

Action	Label	Tab Style	Description
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	









Web Tabs New What Is This?

No Web Tabs have been defined

Visualforce Tabs New What Is This?

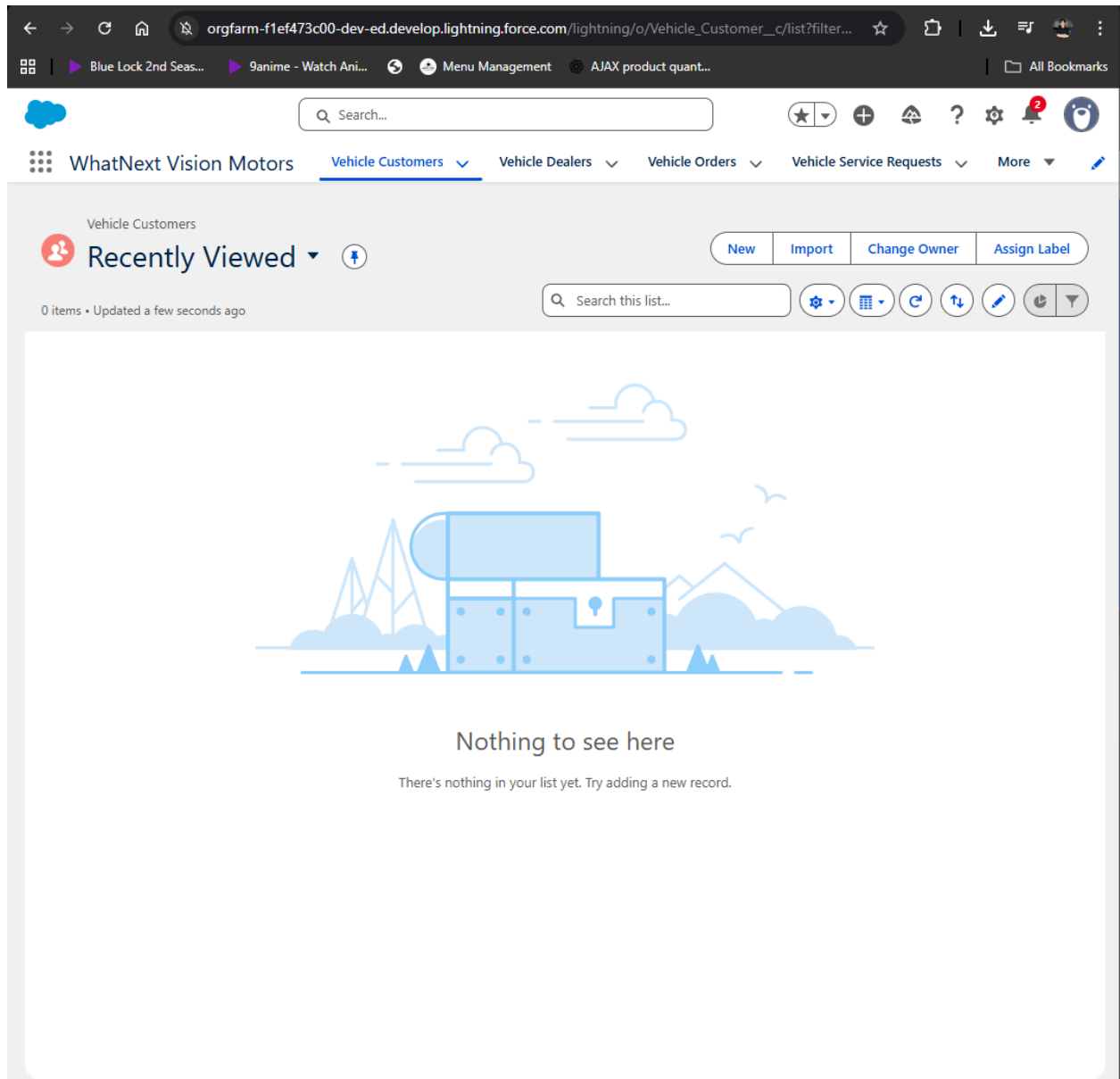
No Visualforce Tabs have been defined

Lightning Component Tabs New What Is This?

Action	Label	Tab Style	Description
Edit 	Get Started with Agentforce	 Heart	
Edit 	Get Started with Data Cloud	 Map	
Edit 	Get Started with MuleSoft	 Heart	
Edit 	Get Started with Salesforce DX	 Building Block	

Phase 5: Lightning App Development

- Used Lightning App Builder to create a custom app named WhatNext Vision Motors
- Added custom object tabs for seamless navigation.
- Customized record pages with components: Related Lists, Charts, and Action Buttons.



Phase 6: Field Creation and Validation Rules

- Created all required fields with appropriate data types like Email, Phone, Picklist, and Lookup.
- Implemented validation rules to prevent incomplete or incorrect submissions.

Vehicle Data Management-Fields

The screenshot shows the Salesforce Setup interface for the 'Vehicle' object. The left sidebar contains a navigation menu with options like Details, Fields & Relationships (selected), Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, Restriction Rules, Scoping Rules, Object Access, Triggers, Flow Triggers, Validation Rules, and Conditional Field Formatting.

The main content area is titled 'Vehicle' and shows the 'Fields & Relationships' section. It displays a table of fields with the following columns: FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The table lists 9 items, sorted by Field Label.

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Price	Price__c	Currency(18, 0)		▼
Status	Status__c	Picklist		▼
Stock Quantity	Stock_Quantity__c	Number(18, 0)		▼
Vehicle Dealer	Vehicle_Dealer__c	Lookup(Vehicle Dealer)		✓ ▼
Vehicle Model	Vehicle_Model__c	Picklist		▼
Vehicle Name	Name	Text(80)		✓ ▼

Vehicle Dealer Data Management-Fields

Vehicle Dealer

SETUP > OBJECT MANAGER

Fields & Relationships

8 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Dealer Code	Dealer_Code__c	Auto Number		
Dealer Location	Dealer_Location__c	Text(60)		
Email	Email__c	Email		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Phone	Phone__c	Phone		
Vehicle Dealer Name	Name	Text(80)		✓

Vehicle Order Data Management-Fields

←

→

↺

🏠

🌐

orgfarm-f1ef473c00-dev-ed.develop.my.salesforce-setup.com/lightning/setup/ObjectManager/01lgK0000...

☆


🔖

📄

👤

⋮

📖 | 🎬 Blue Lock 2nd Seas... | 🎬 9anime - Watch Ani... | 🔄 ⚙️ Menu Management | 🌐 AJAX product quant... | 📁 All Bookmarks



🔍 Search Setup

★

+

🔔

?

⚙️

🔔

👤

Setup

Home

Object Manager

SETUP > OBJECT MANAGER

Vehicle Order

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Search Layouts

List View Button Layout

Restriction Rules

Scoping Rules

Object Access

Triggers

Flow Triggers

Validation Rules

Conditional Field Formatting

Fields & Relationships

8 Items, Sorted by Field Label

Quick Find

New

Deleted Fields

Field Dependencies

Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Order Date	Order_Date__c	Date		▼
Owner	OwnerId	Lookup(User,Group)		✓
Status	Status__c	Picklist		▼
Vehicle	Vehicle__c	Lookup(Vehicle)		✓ ▼
Vehicle Customer	Vehicle_Customer__c	Lookup(Vehicle Customer)		✓ ▼
Vehicle Order Number	Name	Auto Number		✓ ▼

Vehicle Customer Data Management-Fields

←

→

↺

🏠

🌐

orgfarm-f1ef473c00-dev-ed.develop.my.salesforce-setup.com/lightning/setup/ObjectManager/01lgK0000...

☆

🔖

📄

👤

⋮

📦 | 🎬 Blue Lock 2nd Seas... | 🎬 9anime - Watch Ani... | 🔄 Menu Management | 🌐 AJAX product quant... | 📁 All Bookmarks

☁️

🔍 Search Setup

★

+

🔒

?

⚙️

🔔

👤

Setup

Home

Object Manager

SETUP > OBJECT MANAGER

Vehicle Customer

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Search Layouts

List View Button Layout

Restriction Rules

Scoping Rules

Object Access

Triggers

Flow Triggers

Validation Rules

Conditional Field Formatting

Fields & Relationships

8 Items, Sorted by Field Label

Quick Find

New

Deleted Fields

Field Dependencies

Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Text(60)		▼
Created By	CreatedById	Lookup(User)		
Email	Email__c	Email		▼
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Phone	Phone__c	Phone		▼
Preferred Vehicle Type	Preferred_Vehicle_Type__c	Picklist		▼
Vehicle Customer Name	Name	Text(80)		✓ ▼

Vehicle Test Drive Data Management-Fields

Setup

Home

Object Manager

Vehicle Test Drive

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Search Layouts

List View Button Layout

Restriction Rules

Scoping Rules

Object Access

Triggers

Flow Triggers

Validation Rules

Conditional Field Formatting

Fields & Relationships

8 Items, Sorted by Field Label

Quick Find

New

Deleted Fields

Field Dependencies

Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Status	Status__c	Picklist		
Test Drive Date	Test_Drive_Date__c	Date		
Vehicle	Vehicle__c	Lookup(Vehicle)		✓
Vehicle Customer	Vehicle_Customer__c	Lookup(Vehicle Customer)		✓
Vehicle Test Drive Name	Name	Text(80)		✓

Vehicle Service Request Data Management-Fields

Setup

Home

Object Manager

SETUP > OBJECT MANAGER

Vehicle Service Request

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Search Layouts

List View Button Layout

Restriction Rules

Scoping Rules

Object Access

Triggers

Flow Triggers

Validation Rules

Conditional Field Formatting

Fields & Relationships

9 Items, Sorted by Field Label

Quick Find

New

Deleted Fields

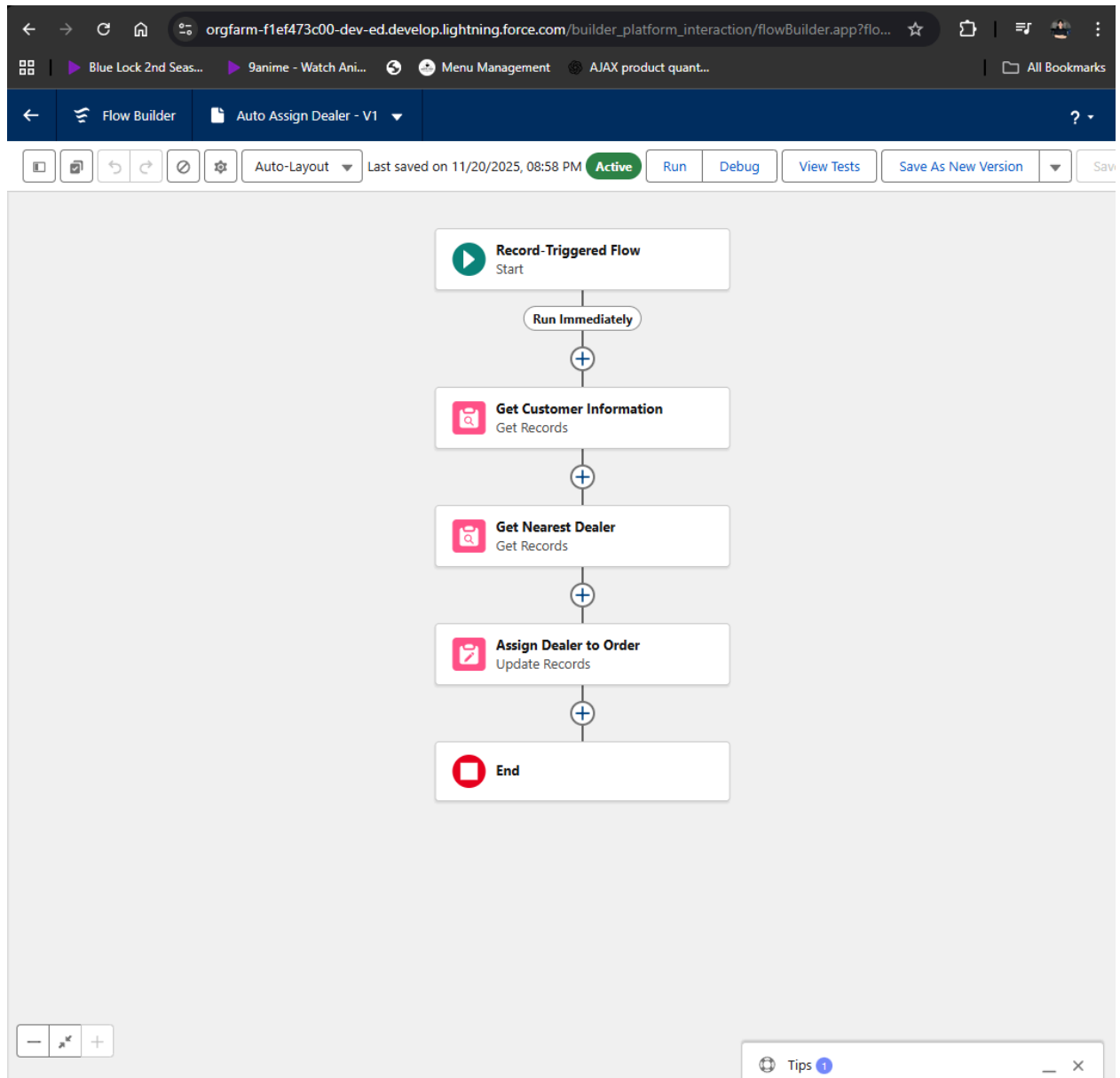
Field Dependencies

Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Issue Description	Issue_Description__c	Text(60)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Service Date	Service_Date__c	Date		
Status	Status__c	Picklist		
Vehicle	Vehicle__c	Lookup(Vehicle)		✓
Vehicle Customer	Vehicle_Customer__c	Lookup(Vehicle Customer)		✓
Vehicle Service Name	Name	Text(80)		✓

Phase 7: Flow Implementation

- Auto-Assignment Flow: Assigns the nearest dealer based on customer address.
- Status Update Flow: Automatically updates order status based on stock.
- Test Drive Reminder Flow: Sends email reminders to customers before their scheduled test drive.
- Configured flow triggers, elements, and conditions using Flow Builder.



orgfarm-f1ef473c00-dev-ed.develop.lightning.force.com/builder_platform_interaction/flowBuilder.app?fl...

Blue Lock 2nd Seas... 9anime - Watch Ani... Menu Management AJAX product quant... All Bookmarks

Flow Builder Test Drive Reminder - V1

Auto-Layout Last saved on 11/20/2025, 09:57 PM Active Run Debug View Tests Save As New Version

```
graph TD; Start([Record-Triggered Flow Start]) --> RunImmediately([Run Immediately]); Start --> ReminderBeforeTestDrive([Reminder Before Test Drive]); RunImmediately --> End1([End]); ReminderBeforeTestDrive --> GetCustomerInfo[Get Customer Information<br/>Get Records]; GetCustomerInfo --> SendTestDriveReminder[Send Test Drive Reminder<br/>Action]; SendTestDriveReminder --> End2([End]);
```

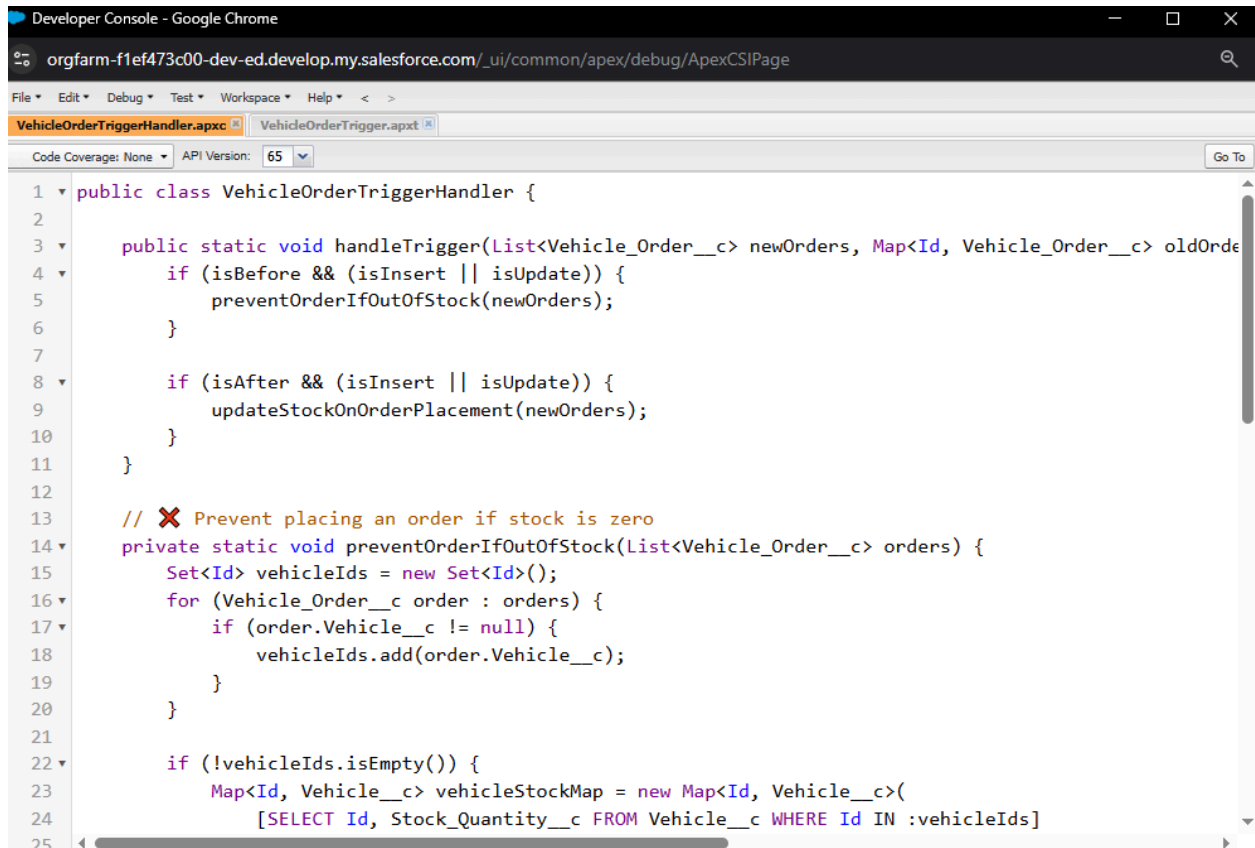
The diagram illustrates a 'Record-Triggered Flow' for 'Test Drive Reminder - V1'. It begins with a 'Record-Triggered Flow Start' node, which branches into two parallel paths. The left path consists of a 'Run Immediately' connector followed by an 'End' node. The right path starts with a 'Reminder Before Test Drive' connector, followed by a sequence of three steps: 'Get Customer Information' (Get Records), 'Send Test Drive Reminder' (Action), and finally another 'End' node. The flow is currently in an 'Active' state, as indicated by the green button in the top toolbar.

— ✎ +

Tips 1

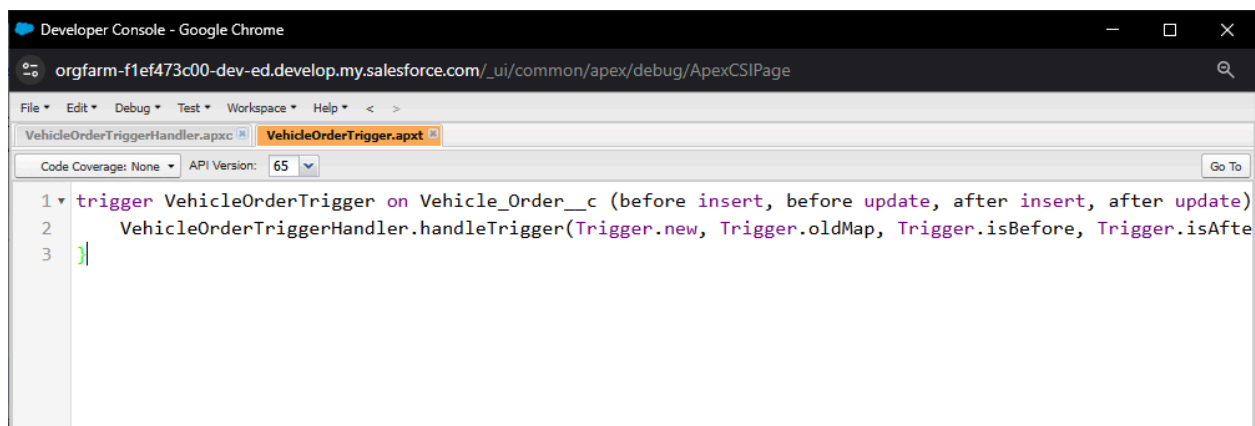
Phase 8: Apex Trigger and Classes

- Developed an Apex Trigger to validate vehicle stock before allowing order creation.
- Created Apex Classes to modularize logic for dealer selection and stock handling.
- Applied Trigger Handler Framework for best practices.



The screenshot shows the Salesforce Developer Console with the file `VehicleOrderTriggerHandler.apxc` open. The code defines a class `VehicleOrderTriggerHandler` with the following methods:

```
1 public class VehicleOrderTriggerHandler {
2
3     public static void handleTrigger(List<Vehicle_Order__c> newOrders, Map<Id, Vehicle_Order__c> oldOrders) {
4         if (isBefore && (isInsert || isUpdate)) {
5             preventOrderIfOutOfStock(newOrders);
6         }
7
8         if (isAfter && (isInsert || isUpdate)) {
9             updateStockOnOrderPlacement(newOrders);
10        }
11    }
12
13    // ✗ Prevent placing an order if stock is zero
14    private static void preventOrderIfOutOfStock(List<Vehicle_Order__c> orders) {
15        Set<Id> vehicleIds = new Set<Id>();
16        for (Vehicle_Order__c order : orders) {
17            if (order.Vehicle__c != null) {
18                vehicleIds.add(order.Vehicle__c);
19            }
20        }
21
22        if (!vehicleIds.isEmpty()) {
23            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>({
24                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
25            });
26        }
27    }
28
29    private static void updateStockOnOrderPlacement(List<Vehicle_Order__c> newOrders) {
30        // Logic to update stock based on new orders
31    }
32}
```



The screenshot shows the Salesforce Developer Console with the file `VehicleOrderTrigger.apxt` open. The code defines a trigger `VehicleOrderTrigger` on the `Vehicle_Order__c` object:

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


Developer Console - Google Chrome

orgfarm-f1ef473c00-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

VehicleOrderTriggerHandler.apxc | VehicleOrderTrigger.apxt | **VehicleOrderBatch.apxc**

Code Coverage: None | API Version: 65 | Go To

```
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.');
```

Developer Console - Google Chrome

orgfarm-f1ef473c00-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

VehicleOrderTriggerHandler.apxc | VehicleOrderTrigger.apxt | VehicleOrderBatch.apxc | **VehicleOrderBatchScheduler.apxc**

Code Coverage: None | API Version: 65 | Go To

```
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 }|
```

Phase 9: Testing and Debugging

- Created test records to simulate real customer orders and vehicle bookings.
- Triggered flows and observed real-time changes.
- Verified email delivery for test drives.

The screenshot shows a Salesforce Lightning interface. At the top, a navigation bar includes a search bar and a menu with items like 'WhatNext Vision Motors', 'Vehicle Customers', 'Vehicle Dealers', 'Vehicle Orders', and 'Vehicle Test Drives'. A green toast message at the top center states: 'Vehicle Test Drive "test drive" was created.' Below this, the record details for 'test drive' are displayed in a table-like format. The table has two columns: 'Field' and 'Value'. The fields include 'Vehicle Test Drive Name', 'Vehicle Customer', 'Vehicle', 'Test Drive Date', 'Status', 'Created By', and 'Last Modified By'. The values are 'test drive', 'Allen Benedict O. Endaya', 'Ford', '11/22/2025', 'Scheduled', 'Allen Endaya, 11/20/2025, 5:59 AM', and 'Allen Endaya, 11/20/2025, 5:59 AM' respectively. Each field has an edit icon (pencil) next to it. The 'Vehicle Test Drive Name' field also has a 'Details' link. The 'Status' field has a dropdown menu with options 'Contact', 'Edit', and 'New Opportunity'.

Field	Value
Vehicle Test Drive Name	test drive
Vehicle Customer	Allen Benedict O. Endaya
Vehicle	Ford
Test Drive Date	11/22/2025
Status	Scheduled
Created By	Allen Endaya, 11/20/2025, 5:59 AM
Last Modified By	Allen Endaya, 11/20/2025, 5:59 AM

REAL WORLD USE CASE

Picture a customer named Riya browsing the WhatNext Motors website or visiting a dealership to purchase a car. The moment her information is entered into Salesforce, the system begins working in the background:

1. Automatic Dealer Matching

- Her pin code or address is analyzed.
- A Flow locates the closest dealer that has her chosen vehicle available.
- The order gets instantly assigned to that dealer — no manual routing needed.

2. Smart Stock Check

- An Apex trigger verifies the vehicle's availability before finalizing the order.
- If the car isn't in stock, the system stops the order and notifies the user immediately.

3. Live Order Progress

- A scheduled Apex job reviews stock levels every night.
- If supply becomes available, the order is automatically moved from "Pending" to "Confirmed."

4. Automated Follow-Ups

- When Riya schedules a test drive, she receives an automated email reminder one day before the appointment.

5. Full Customer Visibility

- Riya can reach out to the assigned dealer anytime and get accurate updates on her order status.

This mirrors the intelligent CRM workflows used by top automobile companies like Hyundai and Tata Motors to improve customer satisfaction and sharply reduce operational confusion.

CONCLUSION

This capstone project highlights how Salesforce can effectively address operational challenges in the automotive sector through a scalable, low-code approach. The development of the customized Lightning App, WhatNext Vision Motors, modernizes vehicle ordering and service handling by automating the entire workflow from collecting customer inquiries to assigning nearby dealers and sending important updates.

Through the integration of custom objects, automation flows, Apex triggers, and batch processing, the system ensures consistent data accuracy and smooth communication between customers and dealers. Automated email alerts and real-time inventory validation help minimize human workload while enhancing service reliability and customer experience.

By leveraging Record-Triggered Flows, Scheduled Flows, and Batch Apex Jobs, the platform continuously oversees vehicle availability and responds proactively to status changes. The structured use of Apex classes along with a trigger handler framework ensures flexibility, making future enhancements easier to implement.

In summary, this solution successfully meets its objectives of boosting automation, improving efficiency, and strengthening user engagement. It also lays the groundwork for more advanced capabilities such as mobile app support, AI-driven insights, and enhanced analytics, demonstrating the full potential of Salesforce in transforming dealership operations into a smarter and more customer-focused environment.

FUTURE SCOPE

1. **IoT Vehicle Tracking:** Integrate IoT devices to provide real-time location and status updates of vehicles during delivery.
2. **Voice Assistant Integration:** Allow customers to place orders or schedule services using voice commands through smart assistants.
3. **Predictive Maintenance Alerts:** Use AI to predict upcoming vehicle maintenance needs based on usage patterns and service history.
4. **Loyalty and Rewards System:** Implement a rewards program for repeat customers, tracking purchases and service activities.
5. **Augmented Reality (AR) Vehicle Preview:** Let customers visualize car models in their environment before purchase using AR technology.
6. **Multi-Language Support:** Expand the system to support multiple languages for a wider customer base.
7. **Dealer Performance AI Insights:** Use AI to provide predictive insights and recommendations to dealers for improving service efficiency and sales strategies.

