

Salesforce CRM Project Documentation

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

Project Overview

WhatNext Vision Motors, a growing company in the automotive industry, wanted to improve how they manage customers, vehicles, and dealer operations. Their old manual process caused delays, wrong stock information, and customer dissatisfaction.

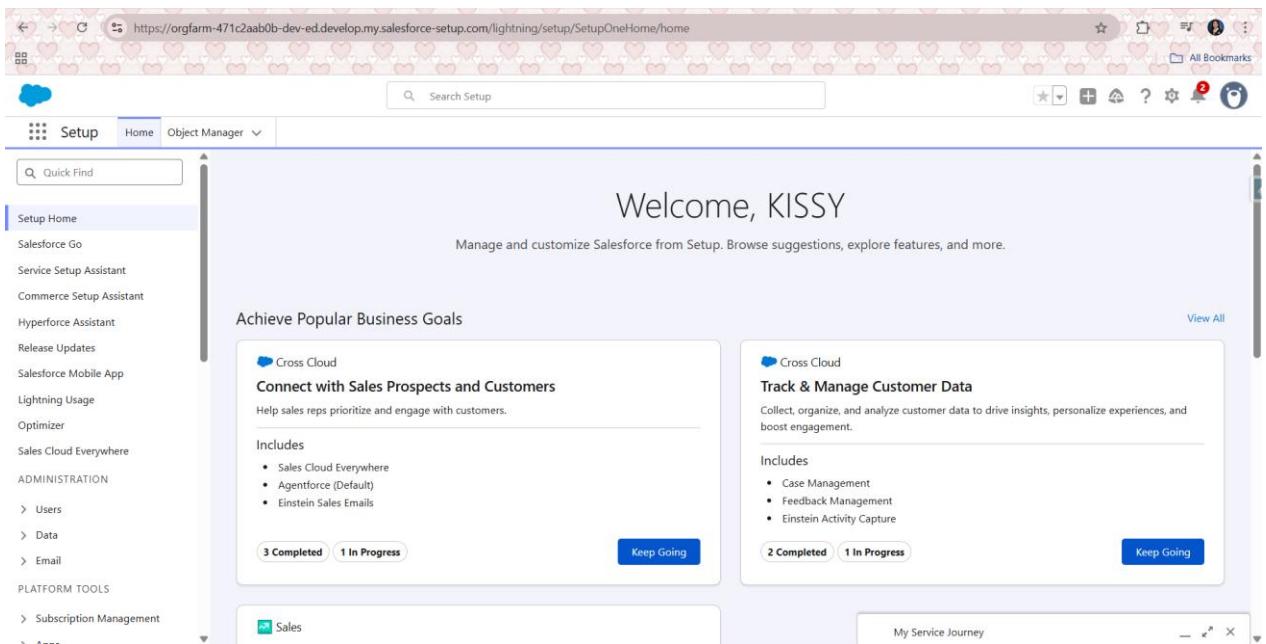
To solve these issues, a customized Salesforce CRM system was created. This system aims to automate vehicle order processing, ensure correct stock management, assign the nearest dealer, and send test-drive reminders automatically.

The system also uses Lightning Apps and Dynamic Forms to provide a clean and user-friendly interface. Overall, the CRM makes operations faster, reduces errors, and prepares the company for future upgrades such as AI-based vehicle recommendations or chatbot support.

Objectives

The key objectives of the Salesforce CRM Project for WhatNext Vision Motors: Shaping the Future of Mobility with Innovation and Excellence are the following:

1. **Automate Order and Dealer Assignment**
 - o Automatically assign the closest dealer based on the customer's city.
2. **Prevent Out-of-Stock Orders**
 - o Use validation rules and Apex triggers to stop customers from ordering vehicles with zero stock.
3. **Send Test Drive Reminders**
 - o Automatically email customers one day before their scheduled test drive.
4. **Improve User Experience**
 - o Use Lightning Apps and Dynamic Forms for a simple and responsive (UI) User Interface.
5. **Maintain a Scalable Backend**
 - o Use Apex classes and batch jobs to handle stock updates and order confirmations in bulk.



Phase 1: Requirement Analysis & Planning

The first phase of the project focused on understanding the business needs of WhatNext Vision Motors and converting them into system requirements for Salesforce. The main goal was to create a CRM that supports the entire vehicle management process from tracking inventory, to handling customer orders, and managing post-sales interactions.

Business Requirements

The system must:

- Store all vehicle, dealer, and customer data in one place.
- Check stock availability during order creation.
- Assign the nearest dealer automatically.
- Track test drives and service requests.
- Automate processes to reduce manual work.

Defining Project Scope

To meet these needs, the system includes:

- Custom objects for vehicles, orders, customers, dealers, test drives, and service requests.
- Record-triggered flows for dealer assignment and email reminders.
- Apex triggers to validate stock and update inventory.

- Batch Apex for processing pending orders.

Data Model

Six custom objects were created to represent the business structure of the Capstone Project.

Object Name	Purpose
Vehicle__c	Stores vehicle details and stock informations
Vehicle_Dealer__c	Dealer information
Vehicle_Customer__c	Stores customer data
Vehicle_Order__c	Tracks vehicle orders
Vehicle_Test_Drive__c	Schedules and tracks test drives
Vehicle_Service_Request__c	Manages service history and issues

These objects are connected through lookup relationships to maintain accurate and consistent data.

The image contains two screenshots of the Salesforce Setup interface.

Top Screenshot: Shows the "Object Manager" section under the "Object Manager" tab. It lists six custom objects: Vehicle, Vehicle Customer, Vehicle Dealer, Vehicle Order, Vehicle Service Request, and Vehicle Test Drive. Each object has its API name, type (Custom Object), description, last modified date (11/13/2025), and deployment status (✓). A search bar at the top right filters results by the word "vehicle".

Label	API Name	Type	Description	Last Modified	Deployed
Vehicle	Vehicle__c	Custom Object		11/13/2025	✓
Vehicle Customer	Vehicle_Customer__c	Custom Object		11/13/2025	✓
Vehicle Dealer	Vehicle_Dealer__c	Custom Object		11/13/2025	✓
Vehicle Order	Vehicle_Order__c	Custom Object		11/13/2025	✓
Vehicle Service Request	Vehicle_Service_Request__c	Custom Object		11/13/2025	✓
Vehicle Test Drive	Vehicle_Test_Drive__c	Custom Object		11/13/2025	✓

Bottom Screenshot: Shows the "Tabs" section under the "Tabs" tab. It displays the "Custom Tabs" section. A message indicates that Custom Object tabs look and behave like standard tabs. Below this, there's a table for creating new custom tabs, listing actions (Edit | Del) and labels (Vehicle Customers, Vehicle Dealers, Vehicle Orders, Vehicles, Vehicle Service Requests, Vehicle Test Drives) next to their corresponding tab styles (People, Building, Box, Car, Form, Gears).

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	

Security Model

- **Standard Salesforce profiles** were used, and **Permission Sets** were added to give users access to the custom objects.
- **Field-Level Security and the Role Hierarchy were applied to make sure users can only view or edit information related to their roles.**
- **Field History Tracking was turned on** for important fields, such as **Stock_Quantity__c (Vehicle)** and **Status__c (Order)**, to support auditing and monitoring.

Phase 2: Salesforce Development – Backend & Configurations o Setup

Environment & DevOps Workflow

To begin the development process, a **Salesforce Developer Org** was set up for building and testing all customizations and automation features.

- **Environment:** Salesforce Lightning Experience (Developer Edition) was used.
- **User Profiles/Roles:** Standard profiles were used for testing, and no custom profiles were created.
- **Deployment Method:** Metadata was deployed from the sandbox to production using **Change Sets**.

Customization of Objects, Fields, Validation Rules, and Automation

Custom Objects and Fields

The following custom objects were created and configured to support the business processes:

- **Vehicle** – Stores information such as vehicle name, model, and stock quantity.
- **Dealer** – Stores dealer details, including location and available vehicles.
- **Customer** – Stores customer information and address.
- **Order** – Records vehicle orders and their status.

Relationships:

- Order → Vehicle: Lookup
- Order → Dealer: Lookup

- Order → Customer: Master-Detail or Lookup (depending on the implementation)

SETUP > OBJECT MANAGER Vehicle		
Fields & Relationships		
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		
Fields & Relationships 10 items, Sorted by Field Label		
FIELD LABEL	FIELD NAME	DATA TYPE
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	Vehicle_Name__c	Text(80)

SETUP > OBJECT MANAGER Vehicle Customer		
Fields & Relationships		
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		
Fields & Relationships 9 items, Sorted by Field Label		
FIELD LABEL	FIELD NAME	DATA TYPE
Address	Address__c	Text(150)
Created By	CreatedById	Lookup(User)
Customer Name	Customer_Name__c	Text(80)
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)
Scoping Rules		

SETUP > OBJECT MANAGER Vehicle Dealer		
Fields & Relationships		
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		
Fields & Relationships 9 items, Sorted by Field Label		
FIELD LABEL	FIELD NAME	DATA TYPE
Created By	CreatedById	Lookup(User)
Dealer Code	Dealer_Code__c	Auto Number
Dealer Location	Dealer_Location__c	Text(100)
Dealer Name	Dealer_Name__c	Text(80)
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)
Scoping Rules		

SETUP > OBJECT MANAGER Vehicle Order		
Fields & Relationships		
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		
Fields & Relationships 8 items, Sorted by Field Label		
FIELD LABEL	FIELD NAME	DATA TYPE
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 Name	Name	Text(80)
Scoping Rules		

SETUP > OBJECT MANAGER Vehicle Service Request		
Fields & Relationships		
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		
Fields & Relationships 9 items, Sorted by Field Label		
FIELD LABEL	FIELD NAME	DATA TYPE
Created By	CreatedById	Lookup(User)
Issue Description	Issue_Description__c	Text(255)
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 Request Name	Name	Text(80)
Scoping Rules		

SETUP > OBJECT MANAGER Vehicle Test Drive		
Fields & Relationships		
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		
Fields & Relationships 8 items, Sorted by Field Label		
FIELD LABEL	FIELD NAME	DATA TYPE
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)
Scoping Rules		

Validation Rules

- **Out-of-Stock Order Blocker:**

This rule stops users from creating an order when the selected vehicle has zero stock.

Automation: Workflow Tools

- A **Record-Triggered Flow** on the Order object automatically assigns the nearest dealer based on the customer's address.
- **Scheduled Flows** were used to send test-drive reminder notifications.

Apex Classes and Triggers

Apex Classes:

Apex classes were developed to organize the trigger logic and support automation in the backend:

- VehicleOrderTriggerHandler – manages stock validation and updates inside the trigger.
- VehicleOrderBatch – checks pending orders and confirms them when stock becomes available.
- VehicleOrderBatchScheduler – schedules the batch job to run every day at 12 PM.

All classes follow best practices by using **bulk-safe operations** and **reusable methods**.

Open				
Entity Type	Entities		Related	
Name	Namespace	Name	Extent	Direction
Entity Type				
Classes	VehicleOrderTriggerHa...			
Triggers	VehicleOrderBatch			
Pages	VehicleOrderBatchSch...			
Page Components				
Objects				
Static Resources				
Packages				

Apex Trigger:

An Apex Trigger was created on the **Order** object to perform the following functions:

- Validate vehicle stock availability.
- Automatically assign a dealer (if this task is not already done by a Flow).
- Order status update logic (**Pending or Confirmed**).

The trigger uses a **Trigger Handler pattern** to follow Salesforce best practices.

Open			
Entity Type	Entities		Related
Entity Type	Name	Namespace ▲	
Classes	VehicleOrderTrigger		
Triggers			
Pages			
Page Components			
Objects			
Static Resources			
Packages			

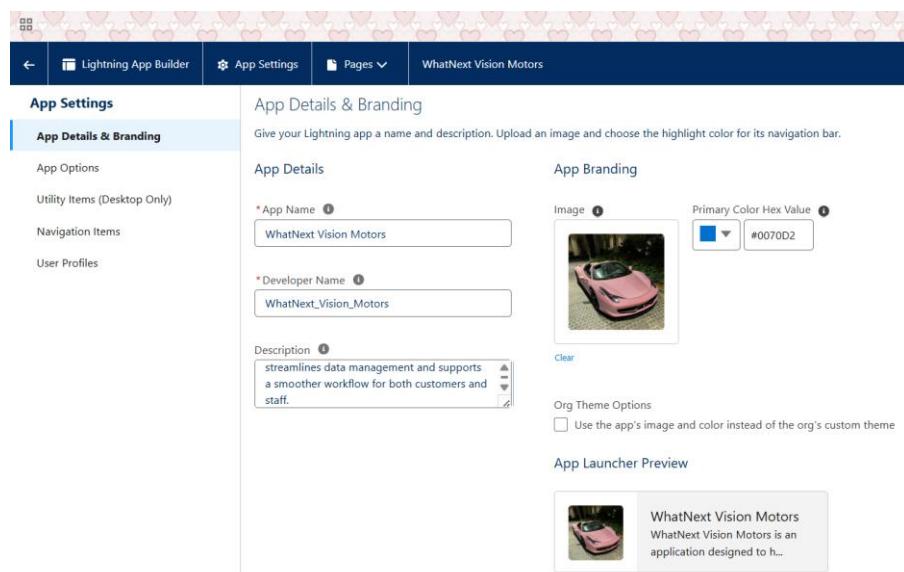
Open Filter Filter the repository (* = any string) Hide Managed Packages Refresh

Phase 3: UI/UX Development & Customization

Lightning App Setup via App Manager

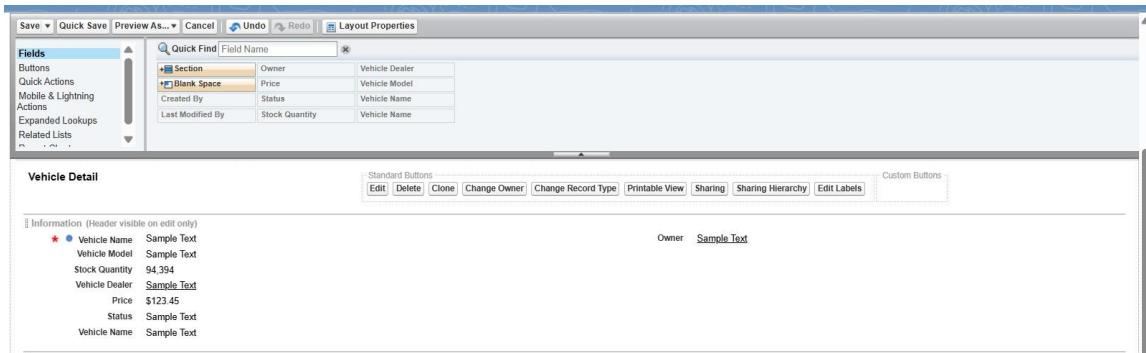
A custom Lightning App called “**WhatNext Vision Motors**” was created using the App Manager. This app includes important custom tabs such as Vehicles, Dealers, Orders, Customers, Test Drives, and Service Requests to make navigation easier.

- Lightning App created: *WhatNext Vision Motors*
- Tabs: Vehicles, Dealers, Customers, Orders, Test Drives, Services
- **Dynamic Forms** were used to show fields based on the record’s status and availability.
- Highlight panels, related lists added to Lightning Pages.



Page Layouts and Dynamic Forms:

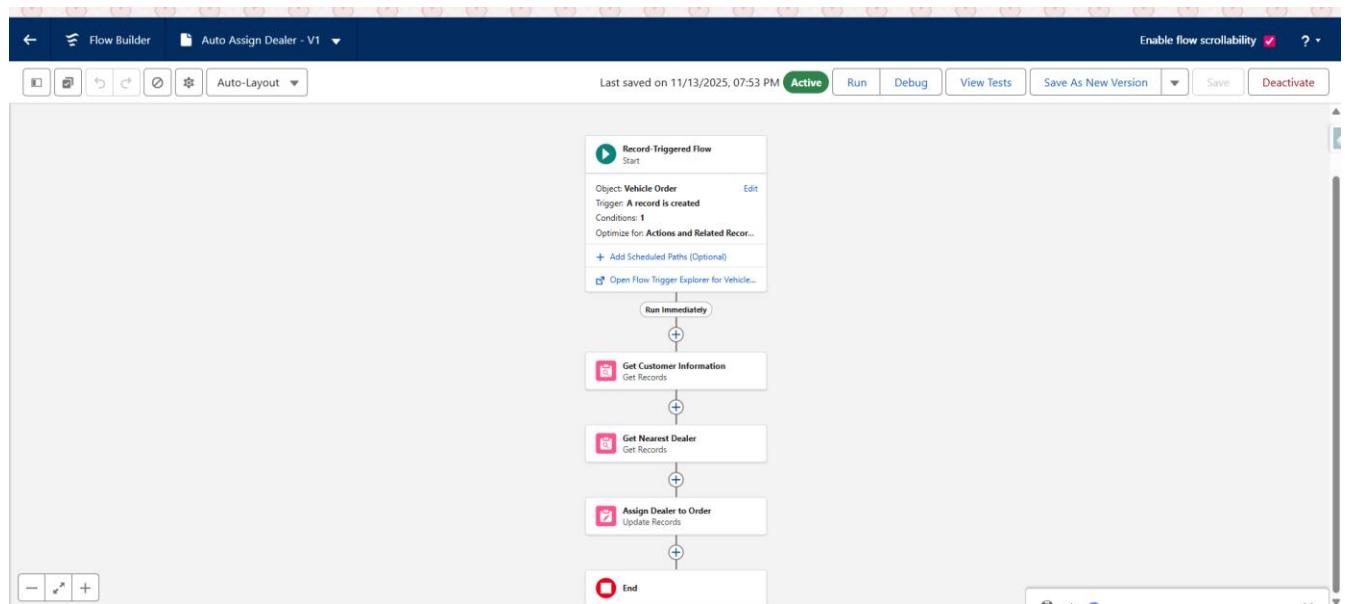
Page layouts were customized for the key objects **Vehicle__c**, **Vehicle_Order__c**, and **Vehicle_Test_Drive__c** to provide a clear interface and show only the fields that users need. Dynamic Forms were applied so that fields appear directly on the Lightning Record Page and are shown or hidden based on conditions such as order status or vehicle availability.



Flow 1: Auto Dealer Assignment

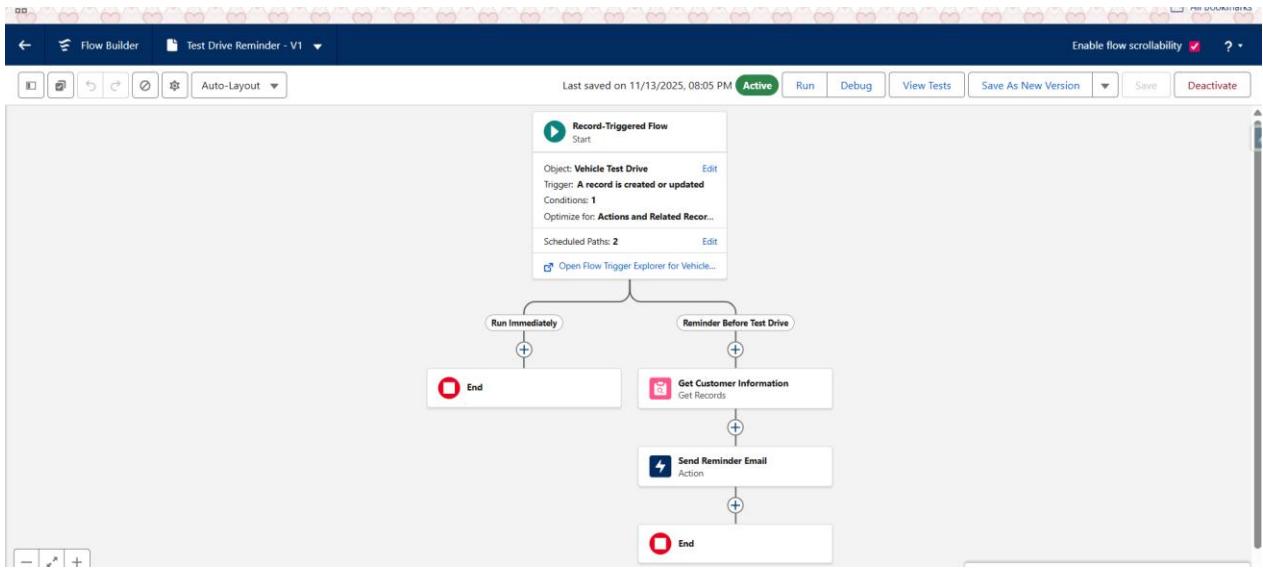
This flow runs on Vehicle_Order__c creation and:

- Retrieves the customer's address.
- Identifies a dealer located in the same city.
- Assigns that dealer to the order



Flow 2: Test Drive Reminder

- This **Record-Triggered Flow** runs when a **Vehicle_Test_Drive__c** record is created or updated.
- Sends an email reminder **one day before** the scheduled test drive.



Apex Trigger & Handler

- Trigger: VehicleOrderTrigger
- Handler: VehicleOrderTriggerHandler
 - Prevents out-of-stock orders
 - Updates stock when order is confirmed

The screenshot shows the Salesforce Developer Console interface with two tabs open:

- VehicleOrderTriggerHandler.apxc**: This tab contains Apex code for a trigger handler class. The code defines a class `VehicleOrderTriggerHandler` with a static method `handleTrigger` that takes a list of `Vehicle_Order__c` objects. It checks if the trigger is before or after insert/update and prevents orders from being inserted if they are out of stock.
- VehicleOrderTrigger.apxt**: This tab contains the trigger definition. It triggers the `VehicleOrderTriggerHandler` class on the `Vehicle_Order__c` object, specifying triggers for `before insert`, `before update`, `after insert`, and `after update` events.

```

1 public class VehicleOrderTriggerHandler {
2
3     public static void handleTrigger(List<Vehicle_Order__c> newOrders, I
4
5         if (isBefore) {
6             if (isInsert || isUpdate) {
7                 preventOrderIfOutOfStock(newOrders);
8             }
9         }
10
11        if (isAfter) {
12            if (isInsert || isUpdate) {
13                updateStockOnOrderPlacement(newOrders);
14            }
15        }
}

```

```

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);
}

```

Apex Batch Class

- Class: VehicleOrderBatch
- Runs daily
- Checks for pending orders and available stock
- Updates status to *Confirmed* and adjusts stock

```

1 global class VehicleOrderBatch implements Database.Batchable<sObject> {
2
3     global Database.QueryLocator start(Database.BatchableContext bc) {
4         return Database.getQueryLocator([
5             SELECT Id, Status__c, Vehicle__c
6             FROM Vehicle_Order__c
7             WHERE Status__c = 'Pending'
8         ]);
9     }
10
11    global void execute(Database.BatchableContext bc, List<Vehicle_Order__c> orderList) {
12        Set<Id> vehicleIds = new Set<Id>();
13        for (Vehicle_Order__c order : orderList) {
14            if (order.Vehicle__c != null) {
15                vehicleIds.add(order.Vehicle__c);
16            }
17        }
18    }
19
20    // If vehicleIds is empty, do nothing

```

Scheduled Apex

- Class: VehicleOrderBatchScheduler
- Cron job runs daily at 12 PM
- Executes batch class automatically

```

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 4: Data Migration, Testing & Security

Data Loading Process

To load initial data into Salesforce, such as vehicles, dealers, and customers, the following tools were used:

Tools Used:

- **Data Import Wizard:**
Used to import data for standard objects like Accounts and Contacts.
- **Data Loader:**
Used for large data volumes and for custom objects like **Vehicle__c**, **Dealer__c**, and **Order__c**.

Steps:

1. Exported CSV files containing sample records.
2. Mapped the CSV columns to the corresponding Salesforce fields.
3. Used **Data Loader** to insert records for:
 - **Vehicle__c**
 - **Dealer__c**
 - **Customer__c**
 - **Order__c** (with valid relationships to other objects)

Field History Tracking, Duplicate Rules, and Matching Rules

Field History Tracking:

Field History Tracking was enabled for the following objects to monitor changes:

- **Vehicle__c**: Stock__c field
- **Order__c**: Status__c and Dealer__c fields

Duplicate & Matching Rules:

- **Matching Rule**: A custom rule on **Customer__c** based on **Email__c** and **Phone__c**.
- **Duplicate Rule**: Prevents the insertion of duplicate customer records.

Profiles, Roles, Permission Sets, and Sharing Rules Profiles and Roles:

- Standard profiles, such as **Standard User** and **System Administrator**, were used.

Role Hierarchy was set up as follows:

- CEO
 - └— Sales Manager
 - └— Sales Rep **Permission**

Sets:

- An **Order Management Access** permission set was created.
- Assigned to users who need **create/read access** to Orders and Vehicles.

Sharing Rules:

- **Public Read/Write** access for most custom objects.
- **Manual Sharing** allowed for sensitive customer records.

Preparation of test cases for each and every salesforce features like booking creation, Approval Process, Automatic Task creation, flows, triggers etc.

1. Create a Vehicle:

INPUT:

Vehicle Name: Honda

Vehicle Model: EV

Stock Quantity: 1

Price: \$80,000

Status: Available

Dealer: Select existing Vehicle Dealer

Vehicle
Honda

Related	Details
Vehicle Name Honda	Owner  KISSY AGUILAR 
Vehicle Model EV	
Stock Quantity 1	
Price \$80,000	
Vehicle Dealer	
Status Available	
Created By  KISSY AGUILAR, 11/13/2025, 2:33 AM	Last Modified By  KISSY AGUILAR, 11/26/2025, 10:59 AM

2. Test Stock = 0 (Error Case):

INPUT:

Edit the Stock Quantity of the above vehicle → Set it to 0.

Go to Vehicle Orders tab → Click New.

- **Vehicle:** Honda
- **Status:** Confirmed
- **Customer:** Select any existing customer

Vehicle Order
O-0012

Related	Details
Vehicle Order Number O-0012	Owner  KISSY AGUILAR 
Vehicle Customer Kissy	
Vehicle Honda	
Order date 11/23/2025	
Status Confirmed	
Assigned Dealer Emma	
Created By  KISSY AGUILAR, 11/23/2025, 5:39 AM	Last Modified By  KISSY AGUILAR, 11/23/2025, 5:39 AM

3. Test Stock > 0 (Confirmed Order)

INPUT:

Steps:

1. Set vehicle Stock Quantity back to 1.

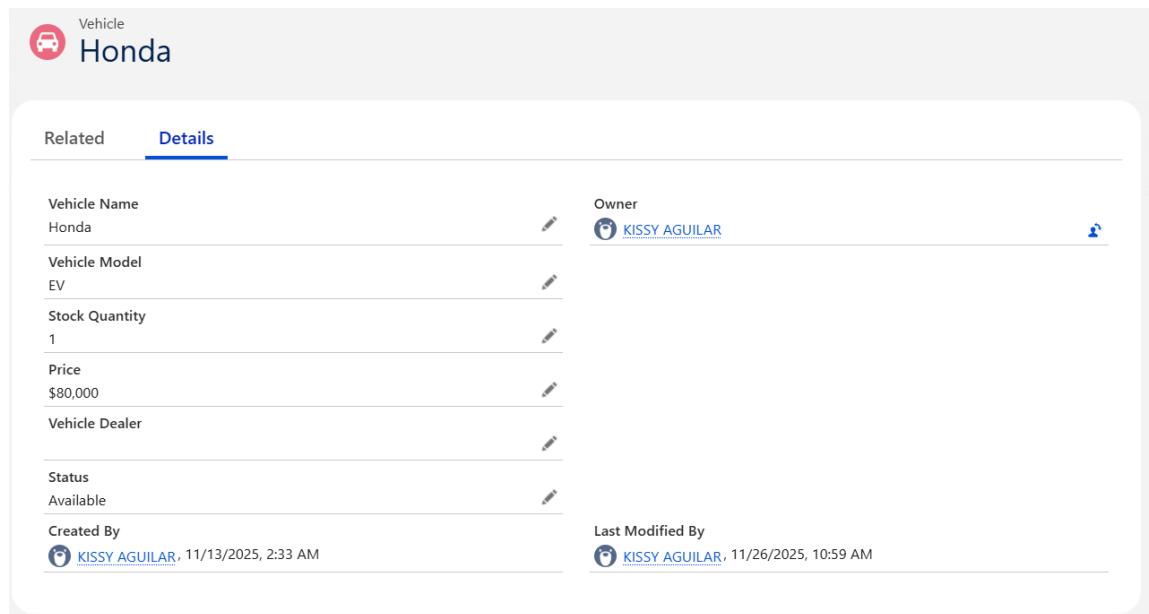
2. Create a Vehicle Order:

- Status: Confirmed

- Vehicle: Test Car

- Vehicle stock should reduce from 2 → 1 automatically.

OUTPUT:



The screenshot shows a vehicle details page for a Honda EV. The vehicle name is Honda, model is EV, and stock quantity is 1. The owner is Kissy Aguilar. The vehicle was created by Kissy Aguilar on 11/13/2025 at 2:33 AM and last modified by her on 11/26/2025 at 10:59 AM.

Related	Details
Vehicle Name	Honda
Vehicle Model	EV
Stock Quantity	1
Price	\$80,000
Vehicle Dealer	
Status	Available
Created By	KISSY AGUILAR, 11/13/2025, 2:33 AM
Last Modified By	KISSY AGUILAR, 11/26/2025, 10:59 AM

4. Test Drive Reminder Email:

Customer: Select any customer with email

Status: Scheduled Test Drive Date: Tomorrow (pick tomorrow's date)

OUTPUT:

Reminder: Your Test Drive is Tomorrow! [Spam](#)

 KISSY AGUILAR via 2ykkzhd2mygfim.gk-emhc5uab.can96.bnc.salesforce.com
to me ▾

Sun 23 Nov, 21:19 (4 days ago)    

Why is this message in spam? This message is similar to messages that were identified as spam in the past.

[Report as not spam](#) 

Dear User Kissy, This is a reminder that your test drive a04gK000002BssQAE is tomorrow. If you need to reschedule please contact us at support@gmail.com.

Thank you!

INPUT:

Create a Pending Order when stock is 0:

1. Set Test Car stock to 0.
2. Create a Vehicle Order:
 - Status: Pending

Update stock: • Set

Stock Quantity = 1

Run batch manually:

```
VehicleOrderBatch job = new VehicleOrderBatch();
Database.executeBatch(job, 50);
```

OUTPUT:

Expected Result:

- Your Pending Order should become Confirmed.
- Vehicle stock should reduce by 1.

Vehicle Order
O-0012

Related	Details
Vehicle Order Number O-0012	Owner  KISSY AGUILAR 
Vehicle Customer Kissy	
Vehicle Honda	
Order date 11/23/2025	
Status Confirmed	
Assigned Dealer Emma	
Created By  KISSY AGUILAR, 11/23/2025, 5:39 AM	Last Modified By  KISSY AGUILAR, 11/23/2025, 5:39 AM

Creation of Test Cases

Vehicle Customer
 Kissy

Related	Details
Vehicle Name Kissy	Owner  KISSY AGUILAR 
Email kissyaguilar1@gmail.com	
Phone 09123456789	
Address Batangas	
Preferred Vehicle Type Sedan	
Created By  KISSY AGUILAR, 11/13/2025, 2:30 AM	Last Modified By  KISSY AGUILAR, 11/23/2025, 4:52 AM

To ensure Apex code is deployable and functional, **Test Classes** were created for:

- OrderTriggerHandler
- DealerAssignmentService
- StockValidationTrigger **Test Class Features:**

- Minimum 75% coverage
- Positive and negative test cases
- Used @isTest annotation with test data setup

Phase 5: Deployment, Documentation & Maintenance

Deployment Strategy

The **Change Set** method was used to deploy features from the Developer Org to the production environment.

Deployment Steps:

1. Created an **Outbound Change Set** in the source org.
2. Added all custom components, including:
 - o Custom objects, fields, flows, validation rules, triggers, and Apex classes.
3. Uploaded the Change Set to the **Target Org** (production/sandbox).
4. Validated and deployed it from **Inbound Change Sets** in the target org.
5. Conducted **post-deployment manual verification** to ensure all features work correctly.

Testing & Sample Scenarios Test

Cases:

- Create vehicle and order with 0 stock → error
- Set stock = 2 → place order → stock becomes 1
- Create pending order → update stock → batch job confirms order

The screenshot shows the Salesforce 'Scheduled Jobs' page under the 'SETUP' tab. The page title is 'Scheduled Jobs'. Below the title, there's a message: 'Percentage of Scheduled Jobs Used: 0% You have currently used 0 scheduled Apex jobs out of an allowed organization limit of 100 active or scheduled jobs. To learn about how this limit is calculated and what contributes to it see the [Lightning Platform Apex Limits](#) topic.' There are two buttons at the top: 'View: All Scheduled Jobs' and 'Create New View'. At the bottom, there are navigation links for letters A-Z and a 'All' link.

Action	Job Name	Submitted By	Submitted	Started	Next Scheduled Run	Type	Cron Trigger ID
Del	Metalytics Data Loader Job for Org : 00DgK00000Emhc5	User_Integration	11/9/2025, 7:24 PM	11/26/2025, 5:59 AM	11/27/2025, 5:59 AM	Autonomous Data Loader Job	08egK00000FPpMu

System Maintenance and Monitoring

To ensure smooth system performance after deployment, the following maintenance strategy was implemented:

1. Monitoring

- Use **Apex Jobs** to monitor scheduled jobs or batch classes.
- Use **Debug Logs** to trace errors or unexpected behavior.
- Enable **Email Alerts** for test drive reminders or failed processes.

2. User Feedback Loop

- The sales and operations teams used the system for a few days after deployment.
- Feedback was collected through **manual walkthroughs** to identify missing features or issues.

3. Updates and Fixes

- Minor updates, such as adding **help text** or updating **field labels**, were made in the sandbox and redeployed using **Change Sets**.
- **Quarterly reviews** were scheduled to implement enhancements or improve the user interface.

Troubleshooting Approach

If issues occur in the production environment, the following steps will be taken:

Step 1: Reproduce the Issue

- Attempt to replicate the problem in a **sandbox** or **developer org**.

Step 2: Enable Debug Logs

- Set **debug logs** for the affected user and analyze the **Flow** or **Apex execution**.

Step 3: Check Apex Jobs or Flows

- For issues related to background processes, review **Apex Job failures** or **Flow error emails**.

Step 4: Fix and Retest

- Update the **Flow** or **Apex logic** as needed.
- Retest the changes in the **sandbox** and redeploy using a **Change Set**.

Conclusion

The Salesforce implementation at **WhatNext Vision Motors** successfully achieved its goal of improving the customer ordering process and operational workflows.

Key achievements include:

- Automatic assignment of the nearest dealer using **Flows or Triggers**.
- **Stock validation** to prevent out-of-stock orders.
- Scheduled logic to update **order statuses** (if Batch Apex is used).
- Improved **customer experience** through automation.
- Reduced **manual work** for internal teams.

This project not only improves the company's customer-facing processes but also provides a solid foundation for future Salesforce expansion and automation. With this initiative, **WhatNext Vision Motors** has advanced closer to its vision of **innovation and excellence in mobility**.