Salesforce CRM Project Documentation

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

Project Overview:

The WhatsNext Vision Motors Salesforce CRM project is a comprehensive solution aimed at transforming the vehicle ordering and customer management processes within the automotive industry.

This CRM system is built to centralize and automate key business functions such as order tracking, stock management, test drive scheduling, and dealer assignment. The platform enables customers to place orders only when vehicles are in stock and automatically maps their orders to the nearest dealer based on location, improving both accuracy and convenience.

With powerful automation features like email reminders for test drives, real-time stock validation, and batch updates for pending orders, the system addresses the growing demand for operational efficiency and enhanced customer experience.

Objectives:

The main goal of building the WhatsNext Vision Motors CRM system is to create a streamlined, automated platform that enhances the customer journey and improves internal efficiency.

Enhance Customer Experience

The CRM system ensures that customers only place orders for vehicles that are currently in stock. This prevents delays, avoids confusion, and builds customer trust by offering a transparent and seamless ordering process.

Smart Dealer Assignment

Orders are automatically assigned to the nearest authorized dealer based on the customer's location. This reduces manual routing, speeds up service delivery, and ensures customers are always connected with the most relevant point of contact.

Improve Operational Efficiency

By digitizing key business functions, the CRM minimizes manual work, reduces errors, and enables faster order fulfillment. The system supports better customer management and provides insights that empower the business to make data-driven decisions.

Phase 1: Requirement Analysis & Planning

Understanding Business Requirements

The business needed a centralized CRM platform to manage vehicles, dealers, customers, orders, test drives, and service requests with real-time visibility and automation. Users also required the ability to restrict order creation when stock was unavailable and automate communication such as test drive reminders.

Defining Project Scope and Objectives

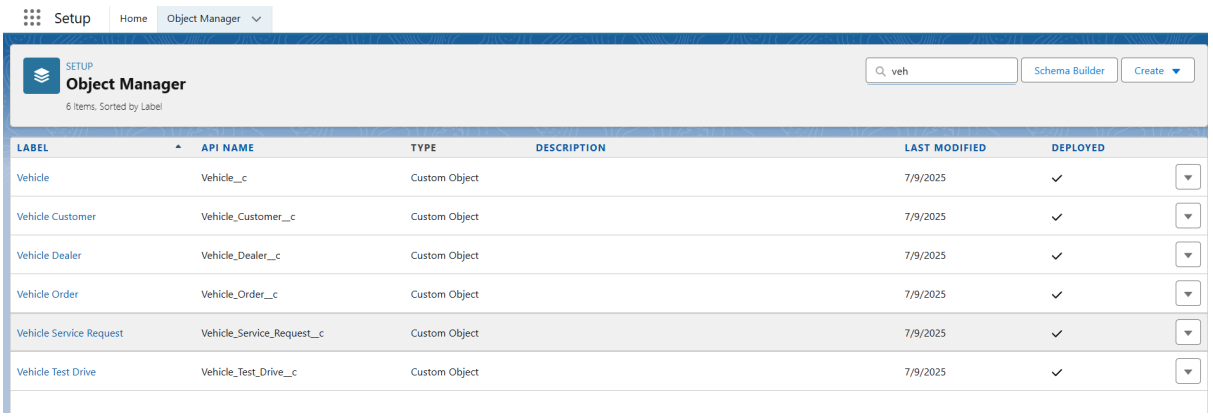
The project scope focused on creating a custom Salesforce CRM that:

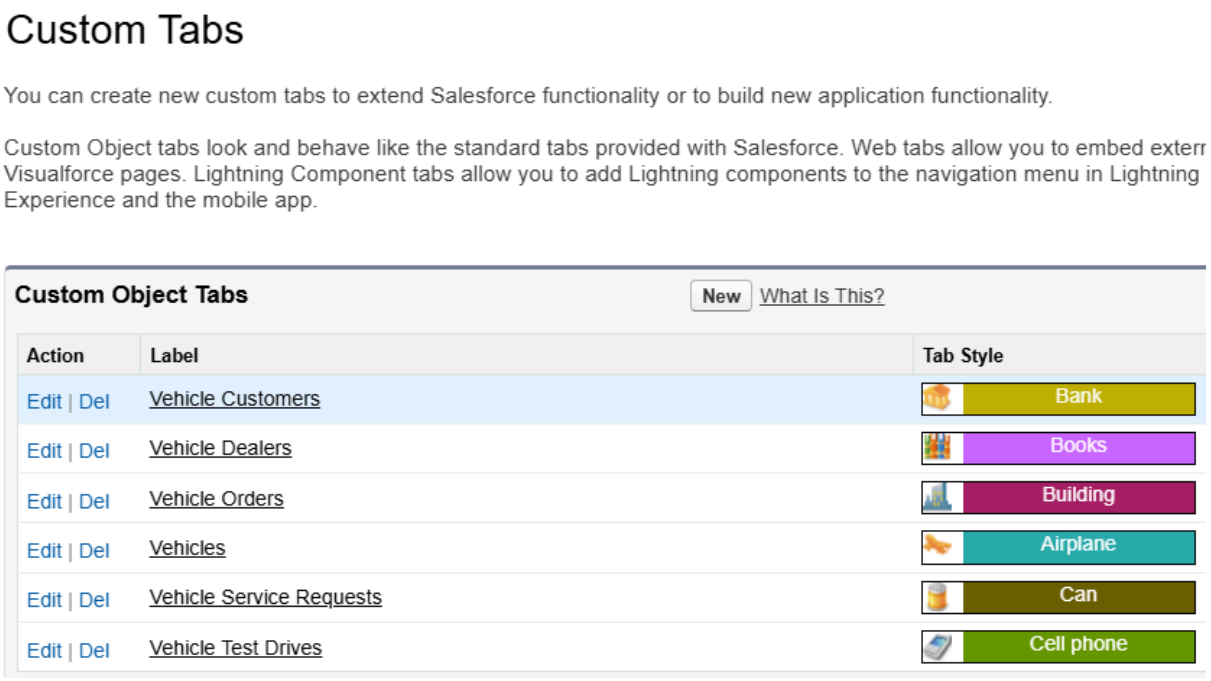
- Stores and manages vehicle and dealer information.
- Tracks customer orders, test drives, and service requests.
- Automatically validates stock availability during order placement.
- Assigns orders to the nearest dealer based on customer location.
- Sends email reminders for scheduled test drives.

Design Data Model and Security Model

A custom data model was designed using six main custom objects:

- Vehicle__c
- Vehicle_Dealer__c
- Vehicle_Customer__c
- Vehicle_Order__c
- Vehicle_Test_Drive__c
- Vehicle_Service_Request__c





Phase 2: Salesforce Development – Backend & Configurations

Setup Environment Workflow

The development environment was set up using a Salesforce Developer Org, where all customizations and configurations were implemented.

Version control was maintained using GitHub to track code and configuration changes. Changes were deployed using Change Sets to ensure smooth movement from development to production environments.

Customization of Objects, Fields, Validation Rules, and Automation :

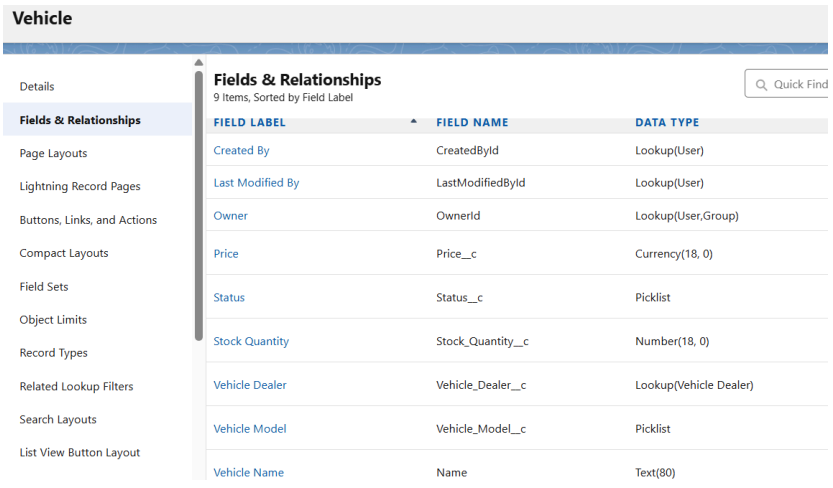
Custom Objects and Fields

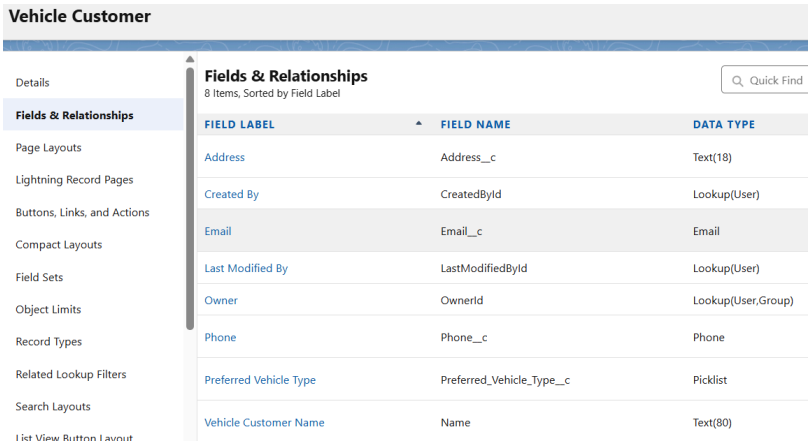
The following custom objects were created and configured to support the business flow:

- **Vehicle** – Stores vehicle name, stock count, model, etc.
- **Dealer** – Stores dealer location and vehicle availability
- **Customer** – Stores customer details and address
- **Order** – Captures vehicle orders and order status

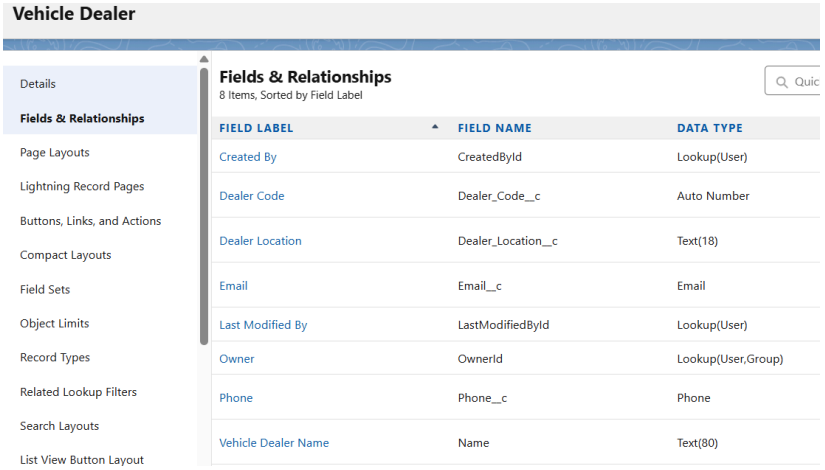
Relationships:

- Order → Vehicle: Lookup
- Order → Dealer: Lookup
- Order → Customer: Master-Detail or Lookup (based on implementation)





Vehicle Dealer

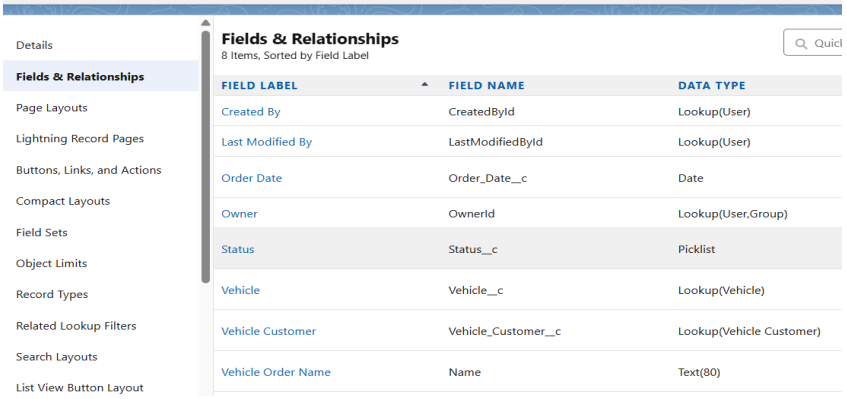


- Details
- **Fields & Relationships**
- Page Layouts
- Lightning Record Pages
- Buttons, Links, and Actions
- Compact Layouts
- Field Sets
- Object Limits
- Record Types
- Related Lookup Filters
- Search Layouts
- List View Button Layout

Fields & Relationships
8 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(18)
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

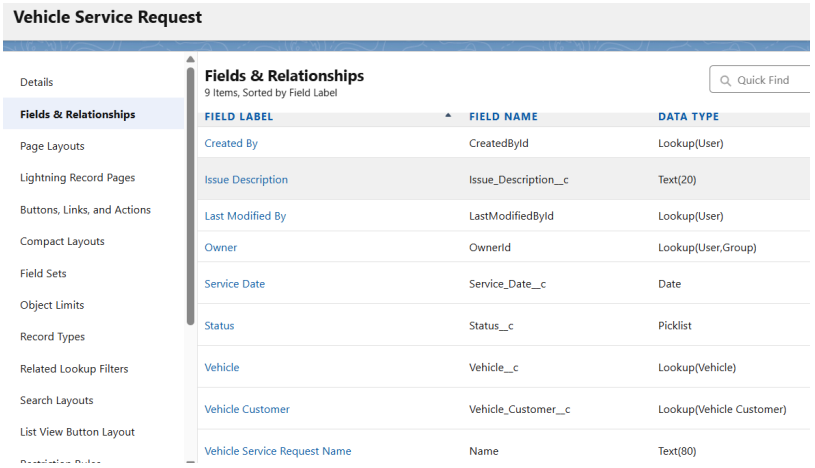


- Details
- **Fields & Relationships**
- Page Layouts
- Lightning Record Pages
- Buttons, Links, and Actions
- Compact Layouts
- Field Sets
- Object Limits
- Record Types
- Related Lookup Filters
- Search Layouts
- List View Button Layout

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)

Vehicle Service Request

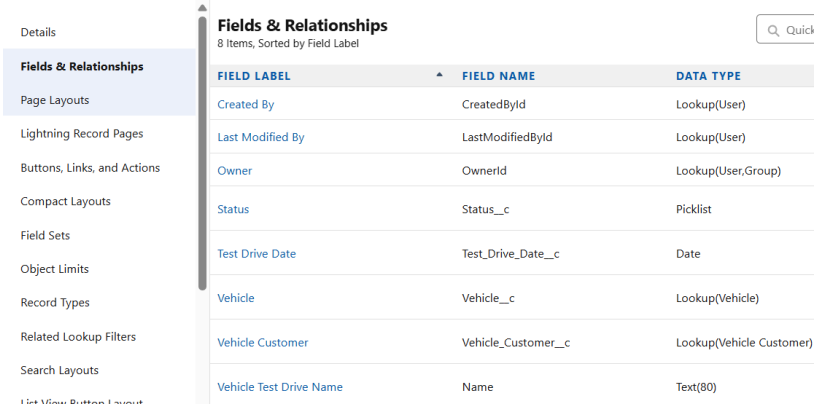


- Details
- **Fields & Relationships**
- Page Layouts
- Lightning Record Pages
- Buttons, Links, and Actions
- Compact Layouts
- Field Sets
- Object Limits
- Record Types
- Related Lookup Filters
- Search Layouts
- List View Button Layout

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(20)
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)

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

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)

Automation -Workflow Tools :

Record-Triggered Flows

1. Auto-Assign Dealer Based on Customer Location

This record-triggered flow automatically assigns the nearest vehicle dealer to a customer's order based on the `City` field in the customer address.

Trigger Object: `Vehicle_Order__c`

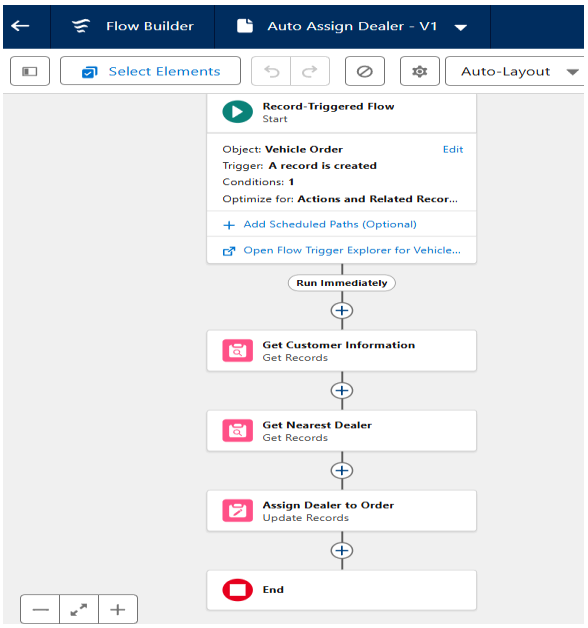
Trigger Condition: When a new Vehicle Order is created.

Criteria: The order must have a related customer and no dealer assigned yet.

Logic:

- **Get the customer's city from the related `Customer__c` record.**
- **Use a Get Records element to find a matching `Vehicle_Dealer__c` in that city.**
- **Assign the found dealer to the `Vehicle_Order__c.Dealer__c` field.**

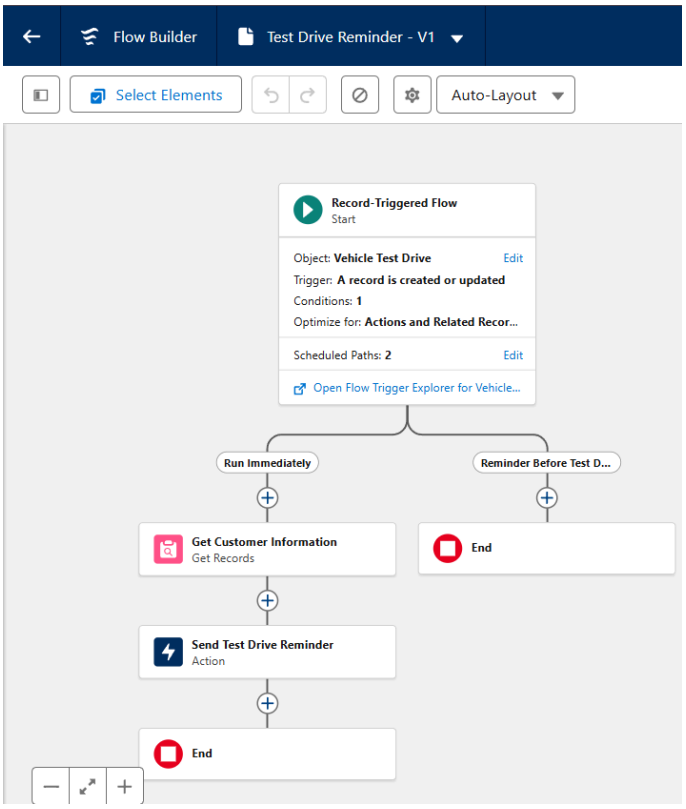
Action: Update the Vehicle Order record with the nearest dealer.



2.Test Drive Reminder Email Flow

This flow sends an automated email reminder to the customer one day before their scheduled test drive.

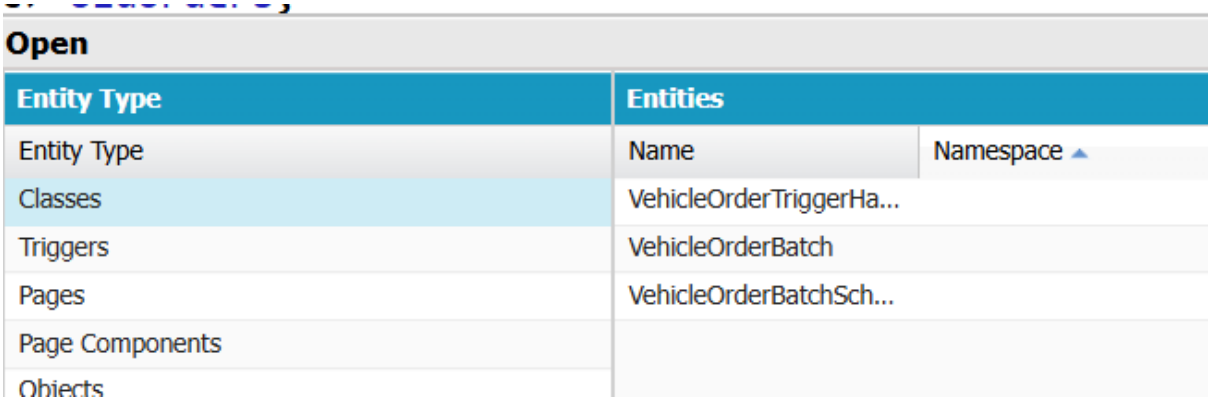
- **Trigger Object: `Vehicle_Test_Drive__c`**
- **Trigger Condition: When a new test drive is scheduled (record is created or updated with a future date).**
- **Scheduled Path:**
  Runs 1 day before the `Test_Drive_Date__c` field value.
- **Logic:**
  - **Get the customer's email from the related record.**
  - **Send an email using a Send Email action.**
  - **Uses an Email Template or custom message body.**
- **Purpose: Improves customer engagement and reduces missed appointments.**



Apex Development- (Apex Classes and Triggers) :

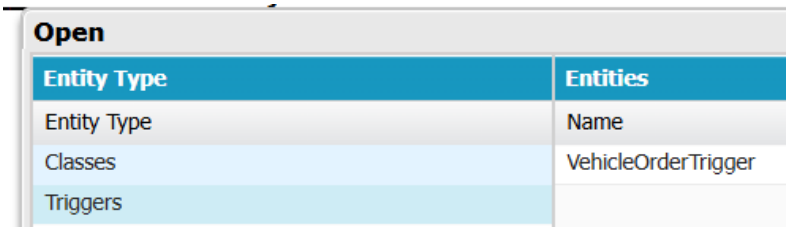
Apex Classes:

Apex Classes were written to modularize the trigger logic and support backend automation:



Apex Trigger:

Order status update logic (Pending or Confirmed)



- **Trigger Handler Class:** `VehicleOrderTriggerHandler`

This Apex class contains reusable logic that is executed when a `Vehicle_Order__c` record is inserted or updated. It follows **best practices** by separating business logic from the trigger.

Key Responsibilities:

- Prevents order placement if the selected vehicle is **out of stock**.
- Automatically **reduces stock quantity** if an order is confirmed.

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

Code Coverage: None | API Version: 64

```apex
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
```

- **Trigger Class:** `VehicleOrderTrigger`

This is the actual **trigger** on the `Vehicle_Order__c` object.

Key Role:

- Calls the `VehicleOrderTriggerHandler` and passes context like `Trigger.new`, `isInsert`, `isUpdate`, etc.
- Responds to changes **before and after insert/update**.

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

Code Coverage: None | API Version: 64

```apex
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
    );
}
```

- **Batch Apex:** `VehicleOrderBatch`

A **batch job** that processes multiple vehicle orders in bulk.

Purpose:

- Scans all orders with status = "**Pending**".
- If related vehicle stock is now available, it **updates the order status to "Confirmed"** and decreases the stock.

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

Code Coverage: None | API Version: 64

```apex
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'
        ]);
    }
```

- **Scheduler Class:** `VehicleOrderBatchScheduler`

This class schedules the `VehicleOrderBatch` to run automatically.

Key Functionality:

- Runs the batch job **daily at a fixed time** (e.g., 12 PM).
- Ensures orders are updated regularly without manual intervention.

File | Edit | Debug | Test | Workspace | Help | < >

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

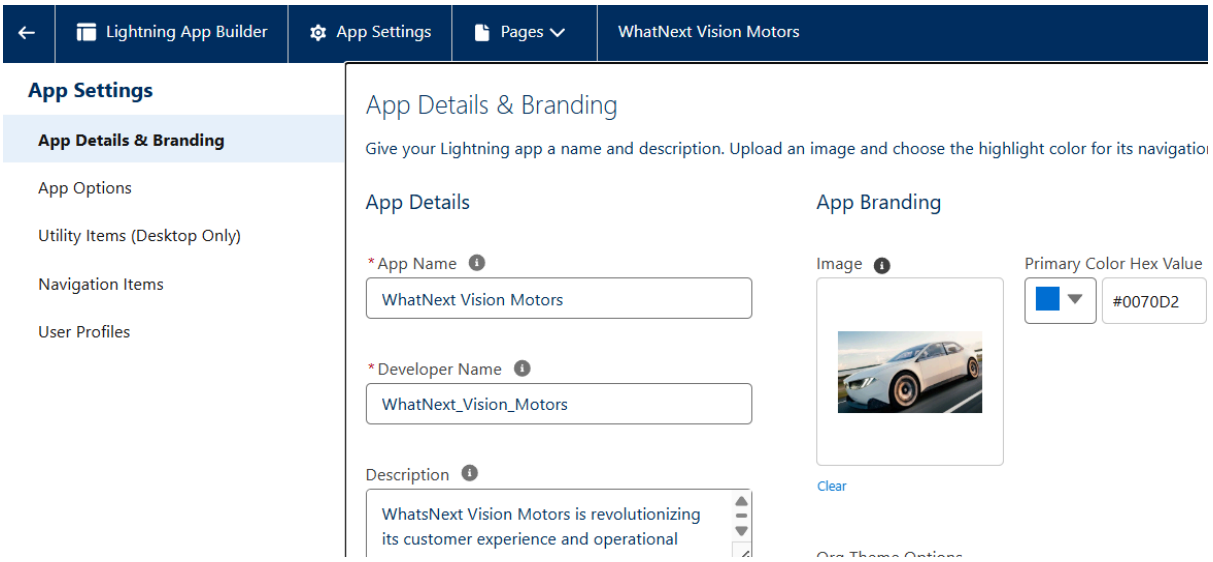
Code Coverage: None | API Version: 64

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

Phase 3: UI/UX Development & Customization

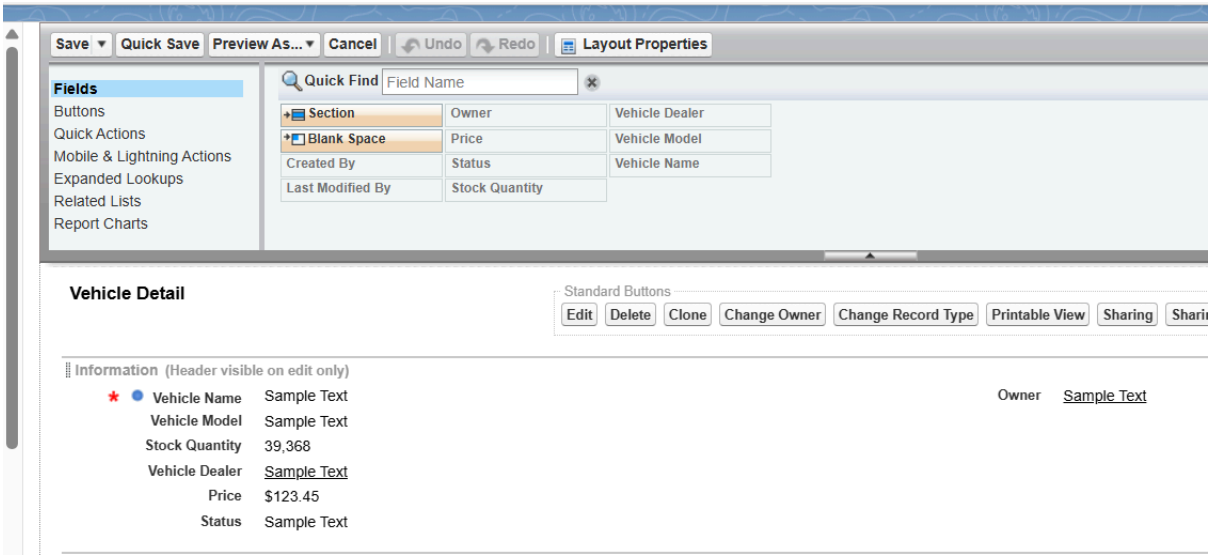
Lightning App Setup via App Manager

A custom Lightning App named **“WhatNext Vision Motors”** was created using App Manager. This app includes relevant custom tabs like Vehicles, Dealers, Orders, Customers, Test Drives, and Service Requests for easy navigation.



Page Layouts and Dynamic Forms :

Page layouts were customized for key objects such as `Vehicle__c`, `Vehicle_Order__c`, and `Vehicle_Test_Drive__c` to ensure clean UI and contextual field visibility. Dynamic Forms were used to place fields directly on the Lightning Record Page and conditionally show fields based on values like order status or vehicle availability.

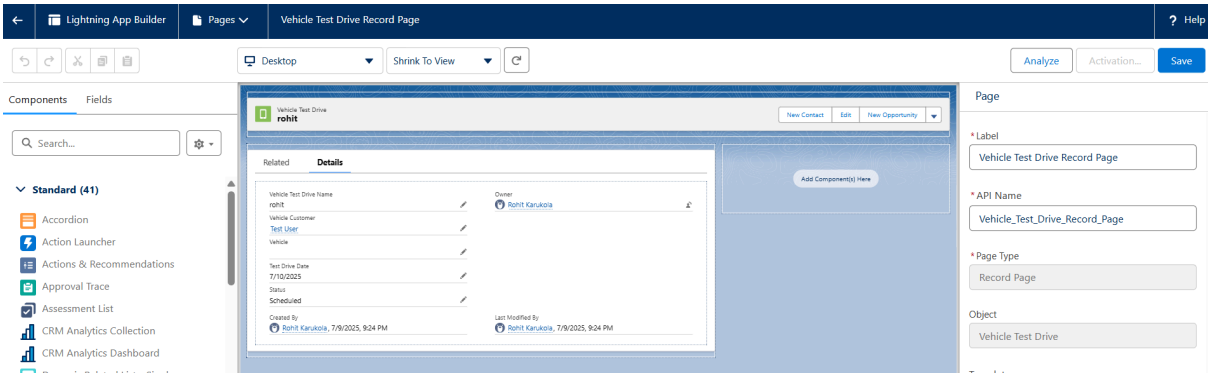


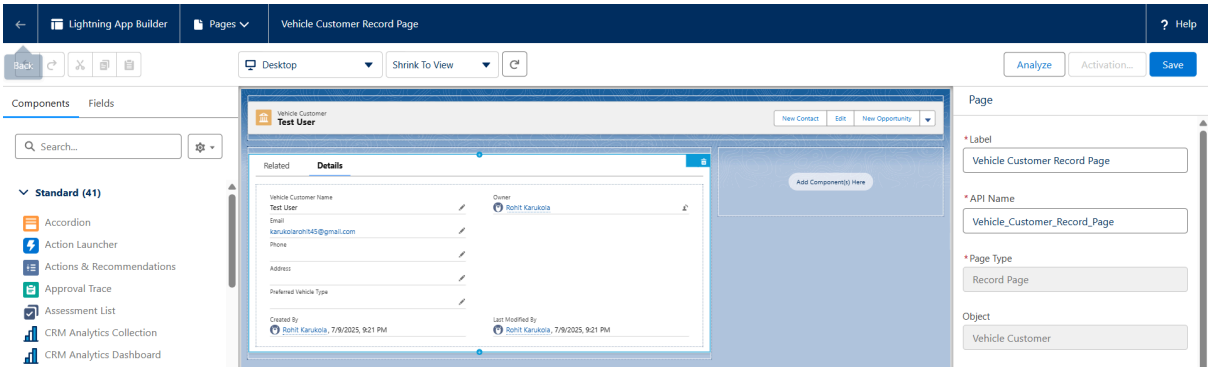
User Management

Standard user profiles were utilized, with permission sets configured to grant access to custom objects and tabs.

Lightning Pages:

Custom Lightning Pages were configured for each object using Lightning App Builder. Components like Related Lists, Highlights Panel, and Tabs were arranged for better user experience.

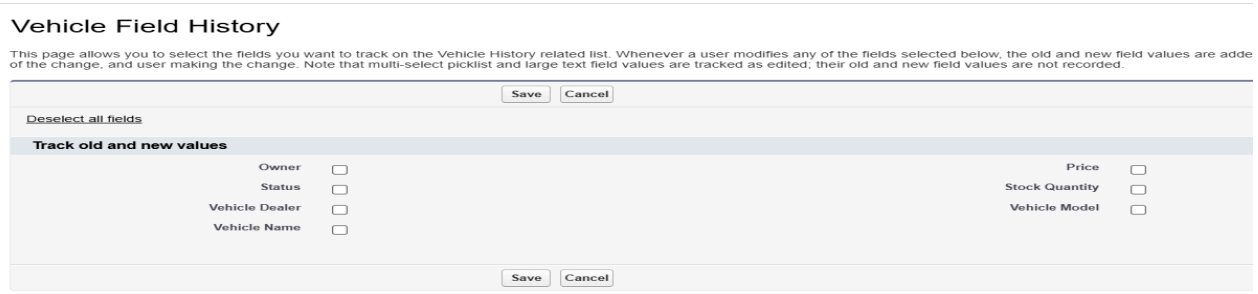




Phase 4:Data Migration, Testing & Security

Field History Tracking

- **Enabled Field History Tracking on `Vehicle__c` (e.g., `Stock_Quantity__c`) and `Vehicle_Order__c` (`Status__c`).**
- **This allowed us to track changes for auditing and rollback if needed.**



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: Brezza

Vehicle Model: EV

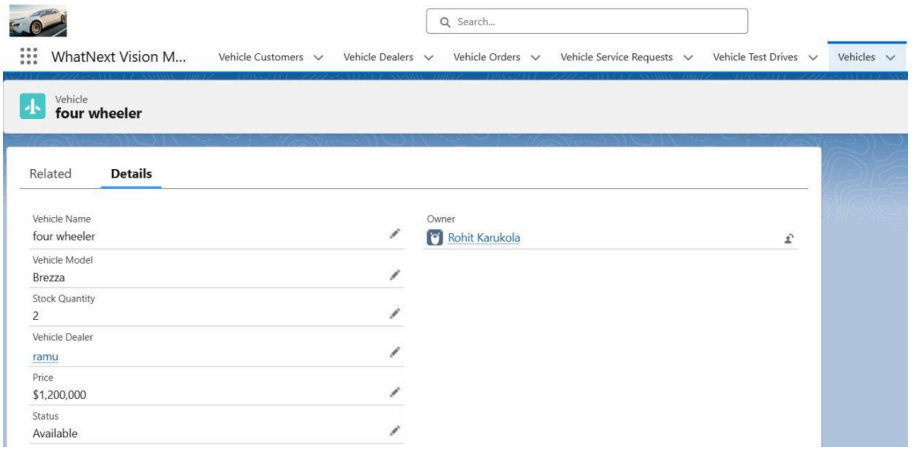
Stock Quantity: 2

Price: 1200000

Status: Available

Dealer: Select existing Vehicle Dealer

OUTPUT:



2. Test Stock = 0 (Error Case):

INPUT:

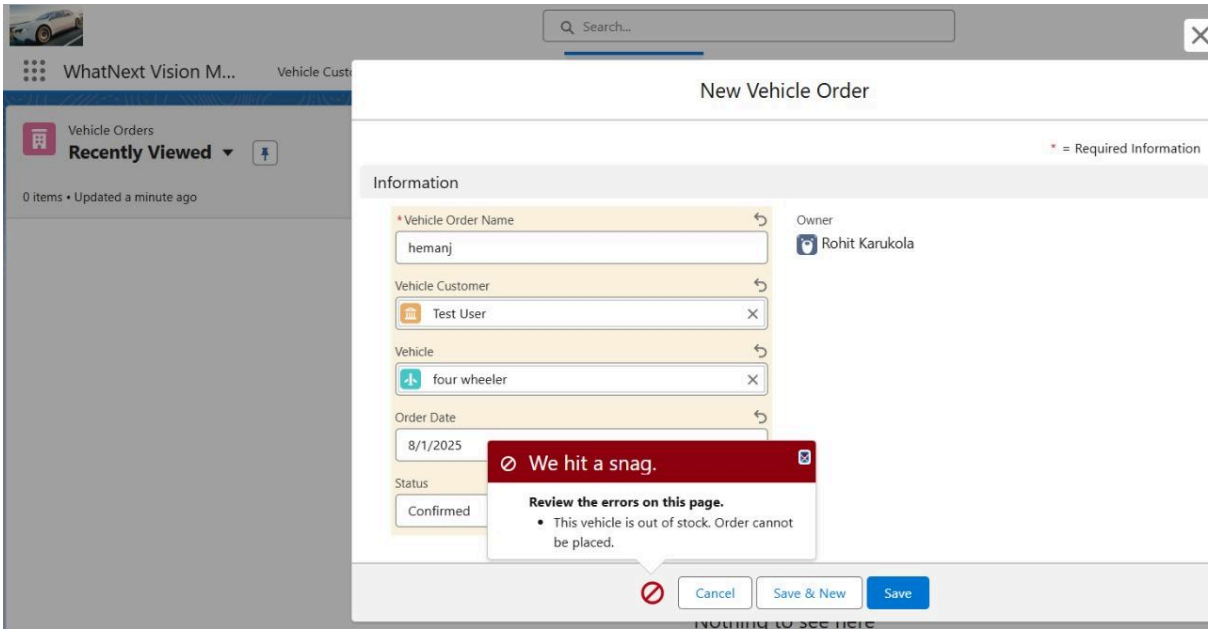
Edit the Stock Quantity of the above vehicle → Set it to 0.

Go to Vehicle Orders tab → Click New.

- Vehicle: BREZZA
- Status: Confirmed

- Customer: Select any existing customer

OUTPUT:



3. Test Stock > 0 (Confirmed Order)

INPUT:

Steps:

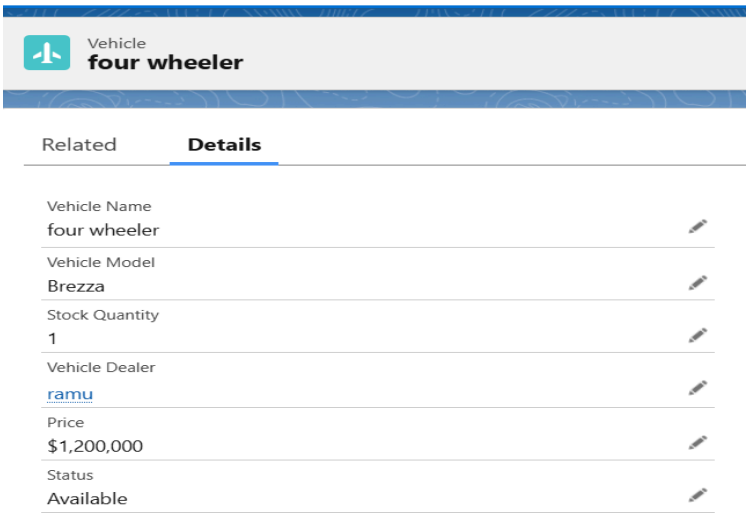
1. Set vehicle Stock Quantity back to 2.

2. Create a Vehicle Order:

    - Status: Confirmed

    - Vehicle: BREZZA
    - Vehicle stock should reduce from 2 → 1 automatically.

OUTPUT:



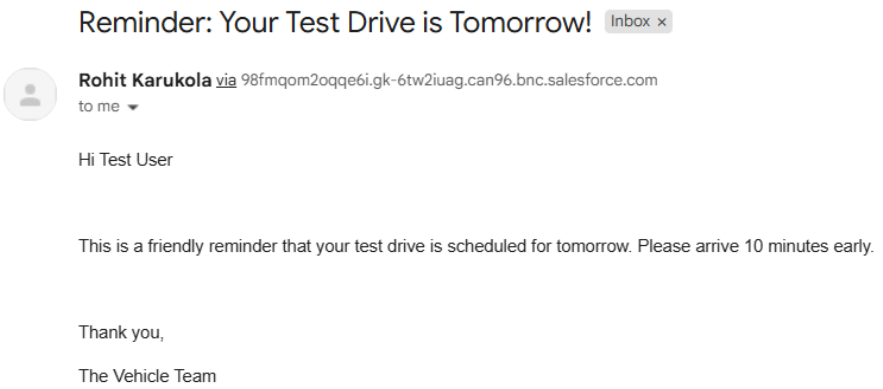
4. Test Drive Reminder Email :

Customer: Select any customer with email.[k****@gmail.com (SAMPLE)]

Status: Scheduled

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

OUTPUT:



Test Batch Job for Pending Orders :

INPUT:

Create a Pending Order when stock is 0:

1. Set BREZZA stock to `0`.
2. Create a Vehicle Order:
    - Status: Pending

Update stock:

- Set Stock Quantity = 2

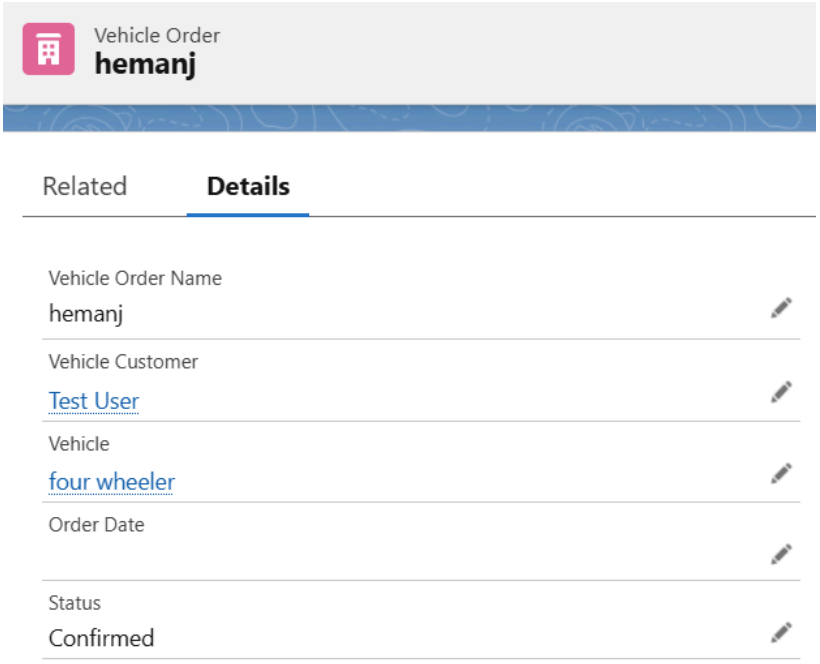
Run batch manually :

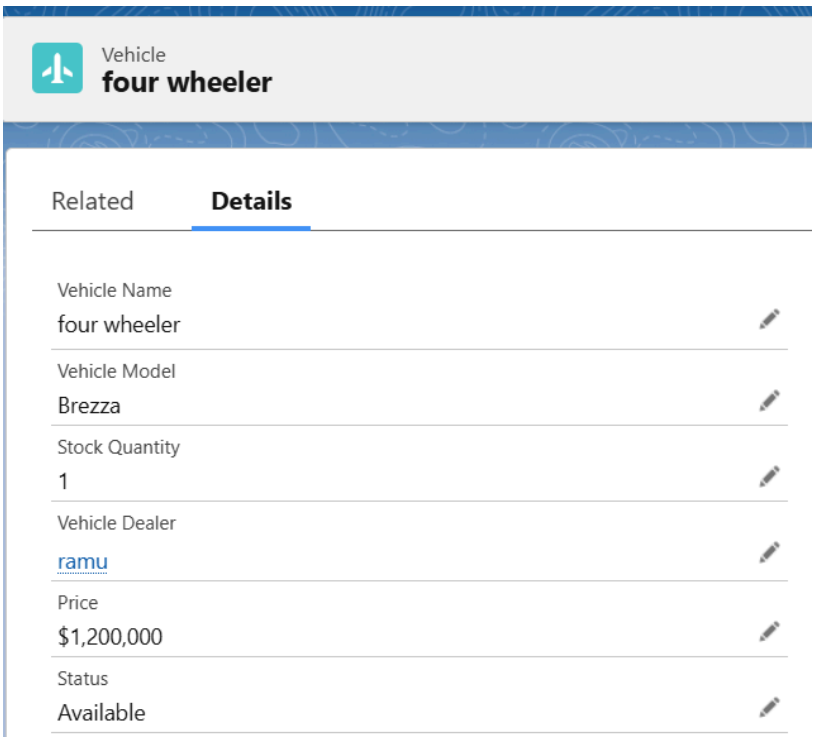
```
VehicleOrderBatch job = new VehicleOrderBatch();

Database.executeBatch(job, 50);
```

OUTPUT:

Expected Result:

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

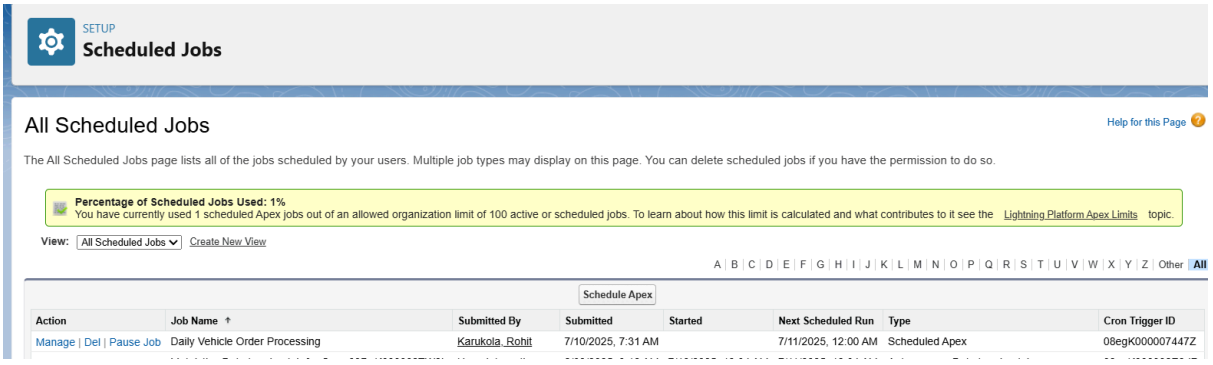




Phase 5:Deployment, Documentation & Maintenance :

Scheduled Job Monitoring:

The scheduled batch job `VehicleOrderBatchScheduler` was successfully deployed and verified in the Scheduled Jobs section.



Deployment Strategy

For this project, deployment was performed using Change Sets, the native Salesforce tool that allows seamless movement of metadata components in production environments.

- Custom Objects (`Vehicle__c`, `Vehicle_Order__c`, `Test_Drive__c`, etc.)

    - Custom Fields and Validation Rules
    - Apex Classes (`VehicleOrderTriggerHandler`, `VehicleOrderBatch`)
    - Flows (Auto Dealer Assignment, Test Drive Reminder)
    - Triggers (`VehicleOrderTrigger`)
- The change set was uploaded to production and deployed after validation.

System Maintenance & Monitoring

To ensure smooth operations post-deployment:

- Scheduled Jobs like `VehicleOrderBatchScheduler` were used to automate order status updates.
- Debug Logs and Flow Error Logs were monitored periodically to detect and resolve any runtime issues.
- Admins are responsible for monitoring Scheduled Jobs via Setup → Jobs → Scheduled Jobs.

Troubleshooting Approach

- Validation Errors were debugged using error messages in Flows and Apex exceptions.
- Flow Builder Debug Mode was used to test logic path during automation development.

Conclusion

The Salesforce CRM project for WhatsNext Vision Motors successfully modernized the vehicle ordering and customer management processes through automation and streamlined workflows.

Key automations like stock validation, test drive reminders, and batch job scheduling ensure operational efficiency and improved customer experience. The project aligns with real-world automotive challenges and sets a solid foundation for scalable future enhancements.

This project has significantly improved customer satisfaction, reduced manual effort, and increased operational efficiency.

Future Enhancements :

1.Chatbot Integration

Integrate a Salesforce-native chatbot (Einstein Bots) to assist customers with FAQs, vehicle availability, test drive bookings, and order status updates — available 24/7 on the website or mobile app.

2.Advanced Analytics & Reporting

Implement **Einstein Analytics (CRM Analytics)** to provide predictive dashboards, trend analysis, and dealer performance reports — enabling smarter, data-driven business decisions.

3.Service Automation

Automate **Vehicle_Service_Request__c** workflows — including auto-assigning service staff, sending service reminders, and tracking maintenance history — to enhance post-sales support.