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

Abstract

WhatNext Vision Motors is redefining the automotive landscape by leveraging Salesforce CRM to modernize and streamline its vehicle order and dealership processes. This comprehensive CRM initiative addresses critical business challenges such as dealer assignment, stock visibility, customer order tracking, and service workflows by harnessing the automation and intelligence of the Salesforce Developer Edition.

At the heart of this project is the seamless enhancement of the customer experience—from the initial vehicle order to the final steps of delivery or service. A standout feature is the system's capability to automatically suggest the nearest available dealer based on the customer's location, reducing customer effort and enabling faster order fulfillment. This intelligent matching not only elevates customer satisfaction but also enhances dealership efficiency.

The project implements automated batch jobs that routinely check for pending orders and confirm them once inventory becomes available, eliminating manual updates. Simultaneously, automated test drive reminders keep customers informed and help reduce missed appointments, boosting service reliability. These smart automations improve operational efficiency while enhancing customer engagement. Additionally, integrated dashboards and reports offer real-time insights into stock levels, dealer performance, and order trends.

By blending declarative tools like Flows and Reports with programmatic tools such as Apex Triggers and Batch Apex, WhatNext Vision Motors has developed a well-balanced, robust system that is scalable, efficient, and ready for the future. This project exemplifies how a traditional industry can embrace digital transformation without compromising its core values and service quality.

Introduction

The traditional vehicle ordering process in the automotive industry is often manual, time-consuming, and susceptible to errors. Within this context, WhatsNext Vision Motors—a company dedicated to innovation and customer satisfaction—recognized that the need for digital transformation was both urgent and strategic.

Salesforce CRM was selected as the technological foundation to develop a smart system capable of digitizing operations while improving the efficiency of day-to-day transactions. This includes intelligently identifying the nearest available dealer for each customer, preventing orders for out-of-stock vehicles, scheduling timely reminders, and continuously tracking stock levels through automated batch jobs.

Objectives

The project aims to:

- Deliver a seamless, user-friendly vehicle ordering experience.
- Validate stock in real time to prevent incorrect order confirmations.
- Dynamically assign dealers based on each customer's proximity.
- Streamline operations by automating tasks with Salesforce Flows and Apex.
- Boost internal productivity and minimize dispatch delays.
- Send automated email reminders for scheduled test drives.
- Automatically refresh pending orders via Batch Apex when stock is replenished.
- Provide executives with dashboards and reports to track inventory, trends, and performance.

Tech Stack Used

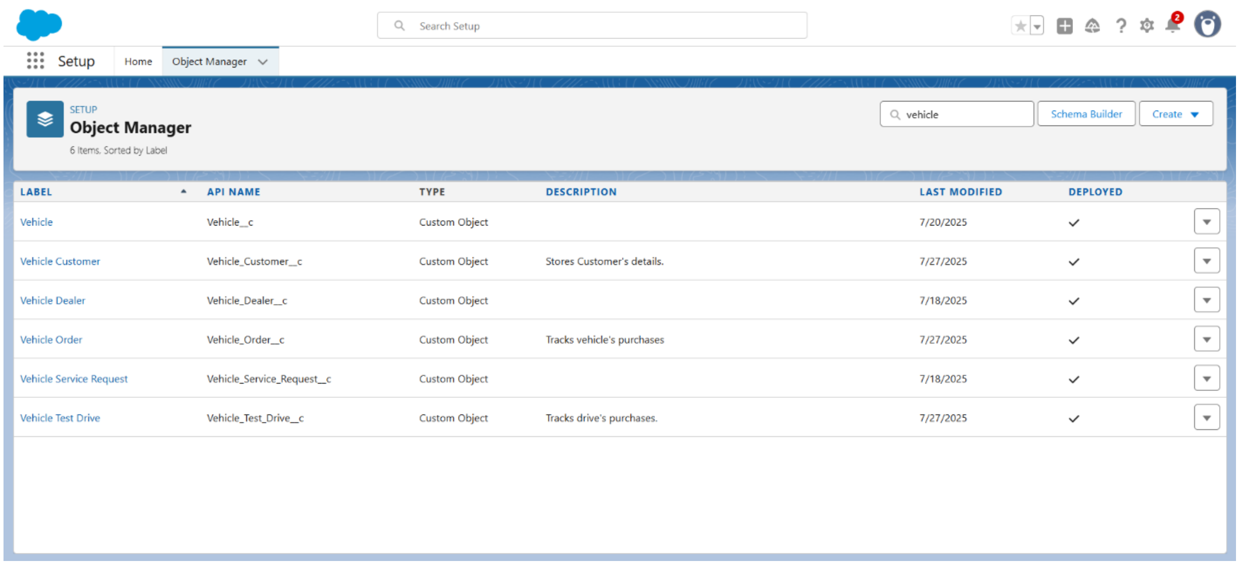
Technology	**Purpose / Usage**
Salesforce Platform	Core CRM platform for building custom objects, automations, and user interfaces
Lightning App Builder	Used to design custom apps, tabs, and page layouts for business objects
Flow Builder	Automate business processes like dealer assignment and test drive reminders
Apex (Trigger & Classes)	For enforcing business logic like stock validation and order status update
Batch Apex	Process large volumes of data (e.g., update pending orders based on stock)
Scheduled Apex	Runs batch jobs automatically at set times (e.g., every midnight)
SOQL (Salesforce Object Query Language)	Used in Apex to query Salesforce records
Email Templates	Send customized automated emails for test drive reminders
Reports & Dashboards	Visual insights into stock, orders, customer activity, and dealer performance
Custom Objects	Represent entities like Vehicles, Dealers, Orders, Customers, Test Drives, etc.

Implementation

Create Data Management-Custom Objects:

The project includes the development of several custom Salesforce objects to accurately model key business entities and their relationships. The custom objects to be implemented are outlined below:

1. **Vehicle__c**: Contains specific details about vehicles, including name, model, type, price, availability status, and stock quantity.
2. **Vehicle_Dealer__c**: Stores dealer-related information such as name, location, and contact details.
3. **Vehicle_Customer__c**: Captures customer data including name, address, email, and preferred vehicle specifications.
4. **Vehicle_Order__c**: Records customer orders along with their status, associated vehicle, and the dealer assigned to the order.
5. **Vehicle_Test_Drive__c**: Manages scheduling and tracking of test drive appointments made by customers.
6. **Vehicle_Service_Request__c**: Logs post-sale service and maintenance requests related to vehicles ordered or purchased by customers.



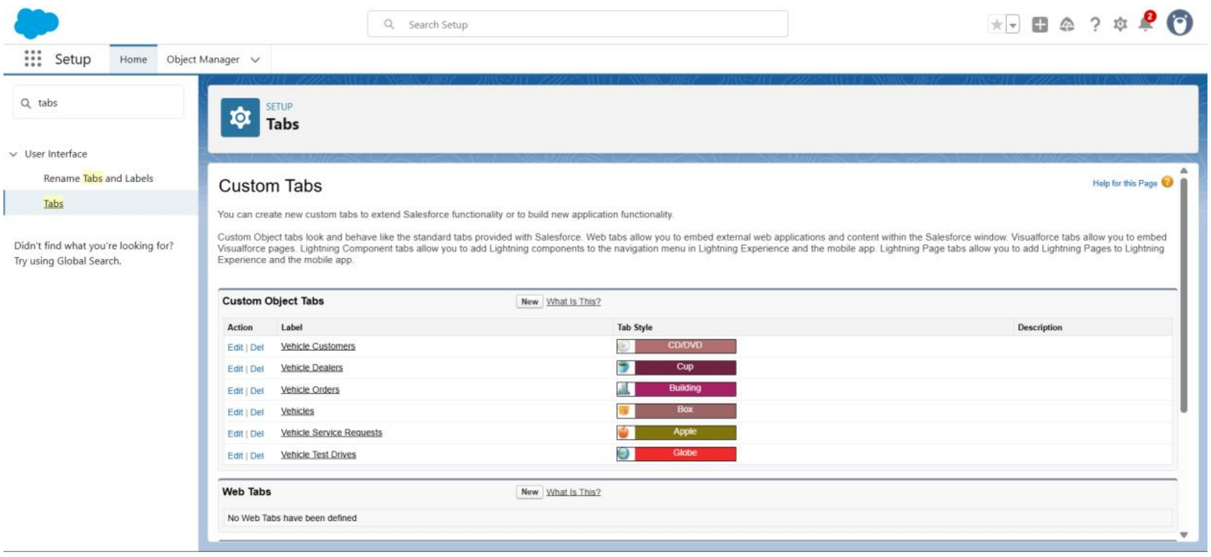
SETUP
Object Manager
6 Items, Sorted by Label

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Vehicle	Vehicle__c	Custom Object		7/20/2025	✓
Vehicle Customer	Vehicle_Customer__c	Custom Object	Stores Customer's details.	7/27/2025	✓
Vehicle Dealer	Vehicle_Dealer__c	Custom Object		7/18/2025	✓
Vehicle Order	Vehicle_Order__c	Custom Object	Tracks vehicle's purchases	7/27/2025	✓
Vehicle Service Request	Vehicle_Service_Request__c	Custom Object		7/18/2025	✓
Vehicle Test Drive	Vehicle_Test_Drive__c	Custom Object	Tracks drive's purchases.	7/27/2025	✓

Create Custom Tabs:

In Salesforce, tabs offer an intuitive way to navigate and interact with custom objects or Visualforce pages. For the WhatsNext Vision Motors project, custom tabs were set up for each key object, allowing users to efficiently access and manage vehicles, dealers, customers, orders, test drives, and service requests.

- **Vehicle** – Used for managing vehicle information including models, pricing, and stock availability.
- **Dealer** – Used to maintain dealership details such as location and contact information.
- **Customer** – Used for recording customer profiles and their vehicle preferences.
- **Order** – Used to view and manage all customer vehicle orders.
- **Test Drive** – Used to monitor and manage scheduled test drive appointments.
- **Service Request** – Used for handling after-sales service and maintenance activities.



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

Custom Tabs

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

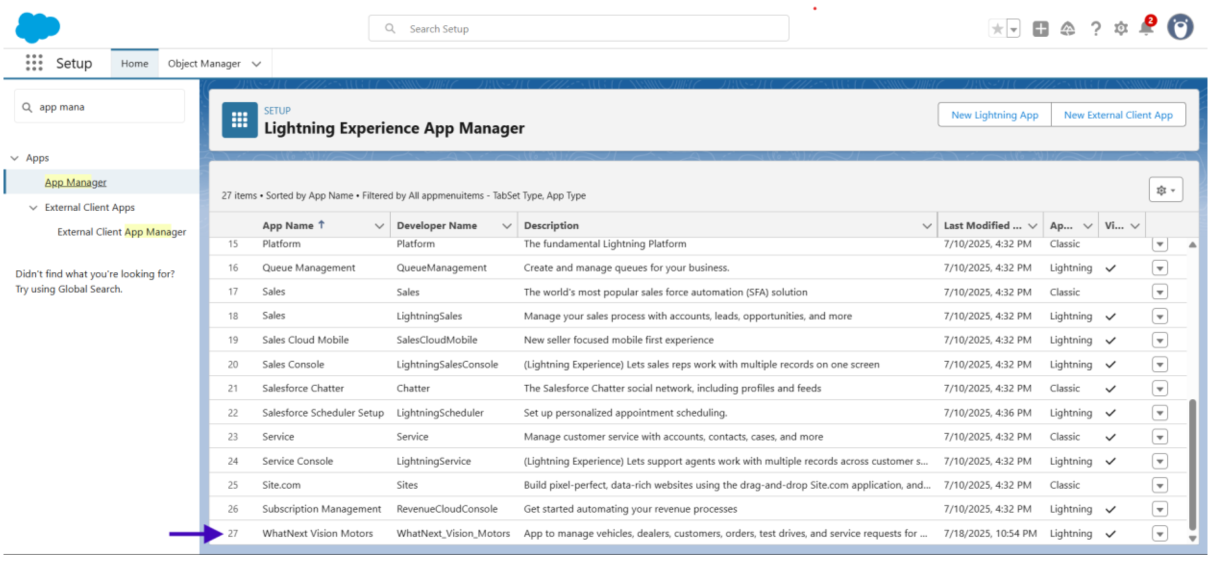
Action	Label	Tab Style	Description
Edit | Del	Vehicle Customers	CD/DVD	
Edit | Del	Vehicle Dealers	Cup	
Edit | Del	Vehicle Orders	Building	
Edit | Del	Vehicles	Box	
Edit | Del	Vehicle Service Requests	Apple	
Edit | Del	Vehicle Test Drives	Globe	

Web Tabs

No Web Tabs have been defined

Creating a Lightning App: WhatsNext Vision Motors

The Lightning App offers a centralized, branded workspace that allows users to conveniently access all essential custom tabs and objects such as Vehicles, Orders, Dealers, Customers, and more. It enhances navigation, ensures a consistent user experience, and aligns the system with the organization’s business identity.



Lightning Experience App Manager

27 items • Sorted by App Name • Filtered by All appmenuitems - TabSet Type, App Type

	App Name	Developer Name	Description	Last Modified ...	Ap...	Vi...
15	Platform	Platform	The fundamental Lightning Platform	7/10/2025, 4:32 PM	Classic	
16	Queue Management	QueueManagement	Create and manage queues for your business.	7/10/2025, 4:32 PM	Lightning	✓
17	Sales	Sales	The world's most popular sales force automation (SFA) solution	7/10/2025, 4:32 PM	Classic	
18	Sales	LightningSales	Manage your sales process with accounts, leads, opportunities, and more	7/10/2025, 4:32 PM	Lightning	✓
19	Sales Cloud Mobile	SalesCloudMobile	New seller focused mobile first experience	7/10/2025, 4:32 PM	Lightning	✓
20	Sales Console	LightningSalesConsole	(Lightning Experience) Lets sales reps work with multiple records on one screen	7/10/2025, 4:32 PM	Lightning	✓
21	Salesforce Chatter	Chatter	The Salesforce Chatter social network, including profiles and feeds	7/10/2025, 4:32 PM	Classic	✓
22	Salesforce Scheduler Setup	LightningScheduler	Set up personalized appointment scheduling.	7/10/2025, 4:36 PM	Lightning	✓
23	Service	Service	Manage customer service with accounts, contacts, cases, and more	7/10/2025, 4:32 PM	Classic	✓
24	Service Console	LightningService	(Lightning Experience) Lets support agents work with multiple records across customer s...	7/10/2025, 4:32 PM	Lightning	✓
25	Site.com	Sites	Build pixel-perfect, data-rich websites using the drag-and-drop Site.com application, and...	7/10/2025, 4:32 PM	Classic	
26	Subscription Management	RevenueCloudConsole	Get started automating your revenue processes	7/10/2025, 4:32 PM	Lightning	✓
27	WhatNext Vision Motors	WhatNext_Vision_Motors	App to manage vehicles, dealers, customers, orders, test drives, and service requests for ...	7/18/2025, 10:54 PM	Lightning	✓

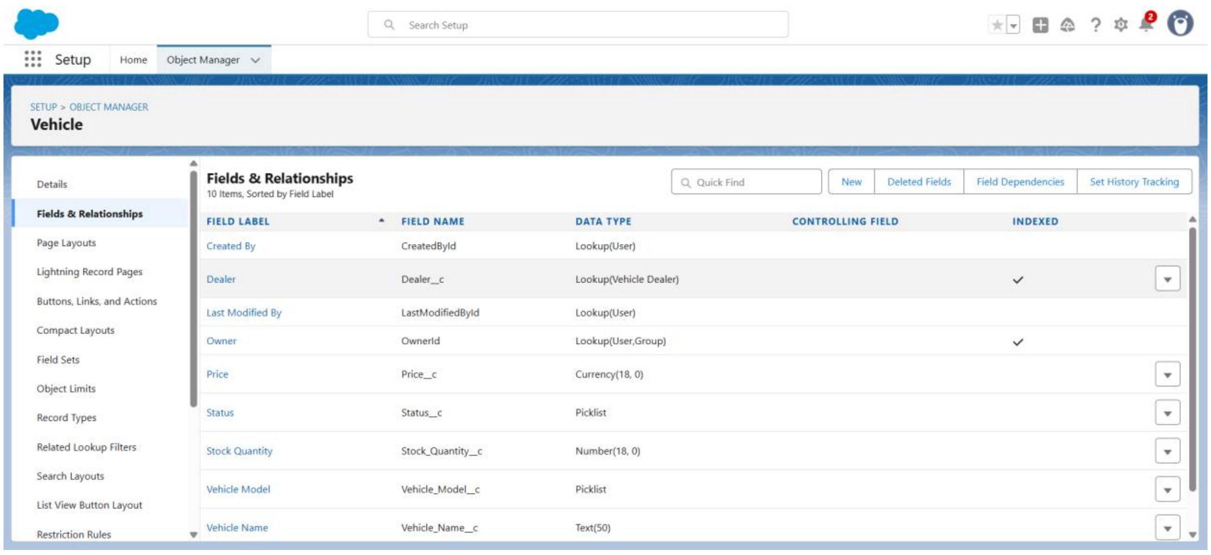
Creating Fields & Relationships

To efficiently manage data and streamline processes within the WhatsNext Vision Motors CRM system, custom fields were added to each object. These fields ensure the system captures all essential details related to vehicles, customers, orders, dealers, and service operations.

1. Vehicle__c (Custom Object)

This object holds the details of each vehicle in the inventory.

Field Label	API Name	Data Type	Description
Vehicle Name	Vehicle_Name__c	Text	Name of the vehicle
Vehicle Model	Vehicle_Model__c	Picklist	Type: Sedan, SUV, EV, etc.
Stock Quantity	Stock_Quantity__c	Number	Current quantity available in stock
Price	Price__c	Currency	Selling price of the vehicle
Dealer	Dealer__c	Lookup	Associated dealer offering the vehicle
Status	Status__c	Picklist	Options: Available, Out of Stock, Discontinued



Vehicle

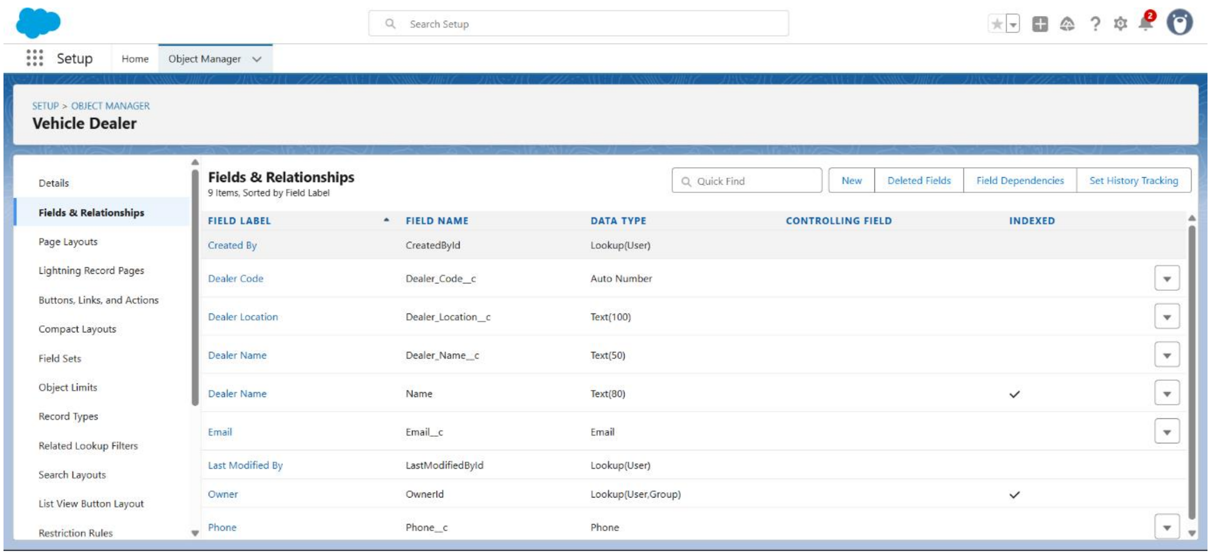
Fields & Relationships
10 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Dealer	Dealer__c	Lookup(Vehicle Dealer)		✓
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 Model	Vehicle_Model__c	Picklist		
Vehicle Name	Vehicle_Name__c	Text(50)		

2. Vehicle_Dealer__c (Custom Object)

Stores the information of authorized dealers in the system.

Field Label	API Name	Data Type	Description
Dealer Name	Dealer_Name__c	Text	Full name of the dealer
Dealer Location	Dealer_Location__c	Text	Location or city where dealer is situated
Dealer Code	Dealer_Code__c	Auto Number	Unique identifier for each dealer
Phone	Phone__c	Phone	Contact number of the dealer
Email	Email__c	Email	Email address of the dealer



Vehicle Dealer

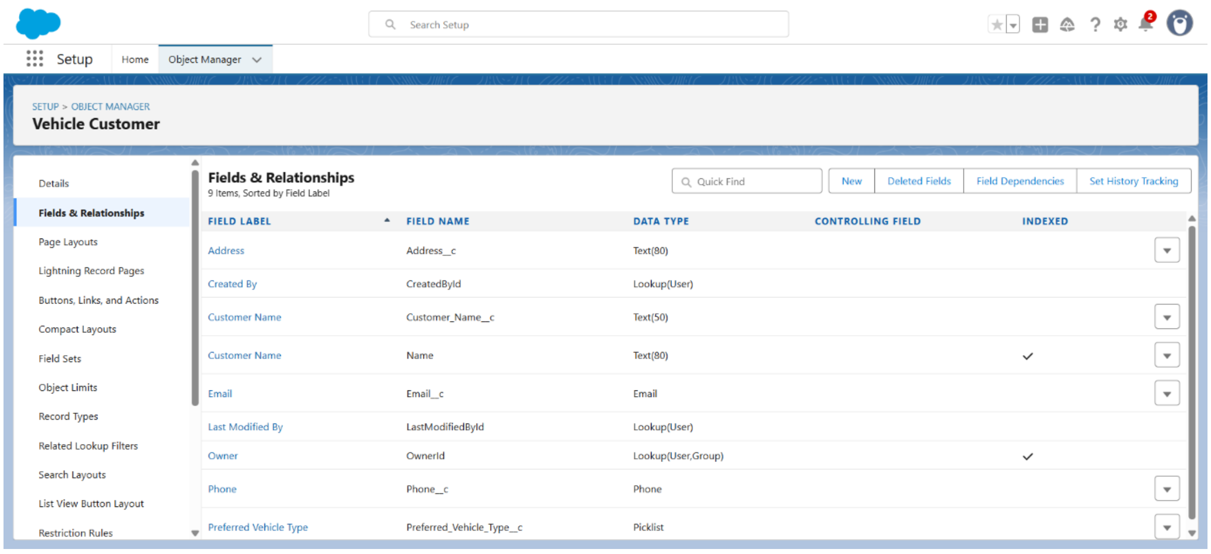
Fields & Relationships
9 Items, Sorted by Field Label

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(100)		
Dealer Name	Dealer_Name__c	Text(50)		
Dealer Name	Name	Text(80)		✓
Email	Email__c	Email		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Phone	Phone__c	Phone		

3. Vehicle_Customer__c (Custom Object)

Captures personal and preference data for customers.

Field Label	API Name	Data Type	Description
Customer Name	Customer_Name__c	Text	Full name of the customer
Email	Email__c	Email	Customer's email address
Phone	Phone__c	Phone	Customer's contact number
Address	Address__c	Text	Complete address including city and state
Preferred Vehicle Type	Preferred_Vehicle_Type__c	Picklist	Vehicle type of interest (e.g., SUV, Sedan, EV)



Vehicle Customer

Fields & Relationships

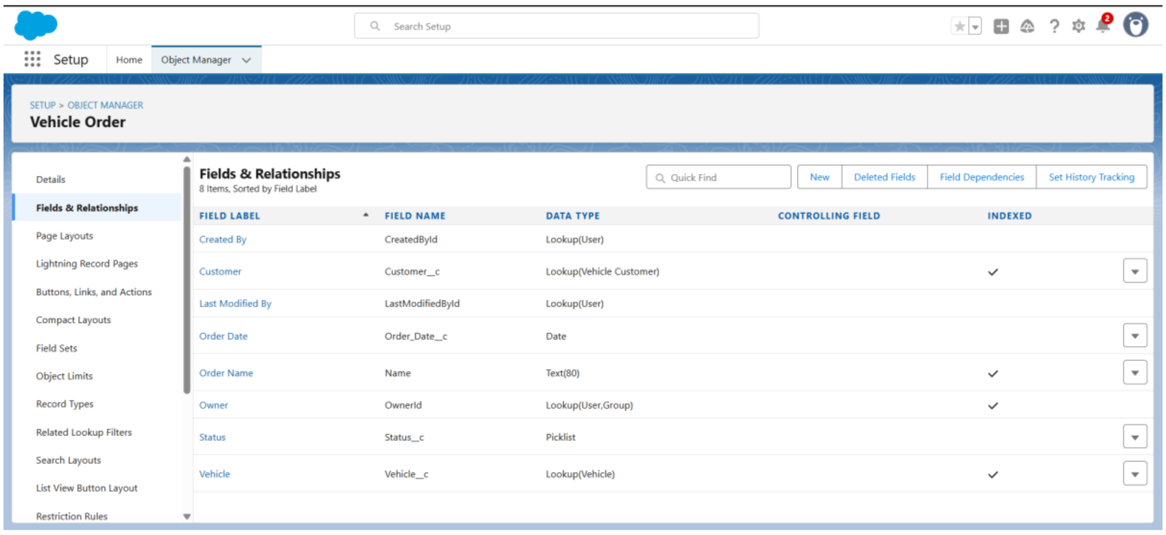
9 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Text(80)		
Created By	CreatedById	Lookup(User)		
Customer Name	Customer_Name__c	Text(50)		
Customer Name	Name	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		

4. Vehicle_Order__c (Custom Object)

Used to track and manage all customer orders.

Field Label	API Name	Data Type	Description
Customer	Customer__c	Lookup	Links to the customer who placed the order
Vehicle	Vehicle__c	Lookup	Vehicle ordered by the customer
Order Date	Order_Date__c	Date	The date on which the order was placed
Status	Status__c	Picklist	Pending, Confirmed, Delivered, or Canceled



Vehicle Order

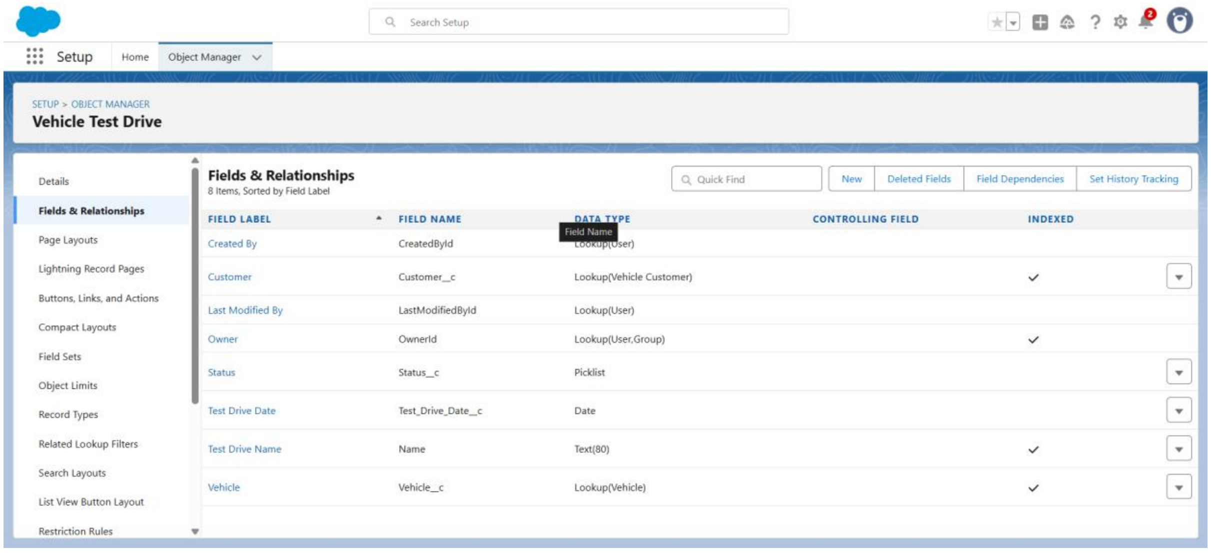
Fields & Relationships
8 Items, Sorted by Field Label

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

5. Vehicle_Test_Drive__c (Custom Object)

Used for test drive appointment scheduling and tracking.

Field Label	API Name	Data Type	Description
Customer	Customer__c	Lookup	Customer who scheduled the test drive
Vehicle	Vehicle__c	Lookup	Vehicle to be test-driven
Test Drive Date	Test_Drive_Date__c	Date	Scheduled date of the test drive
Status	Status__c	Picklist	Scheduled, Completed, or Canceled



Vehicle Test Drive

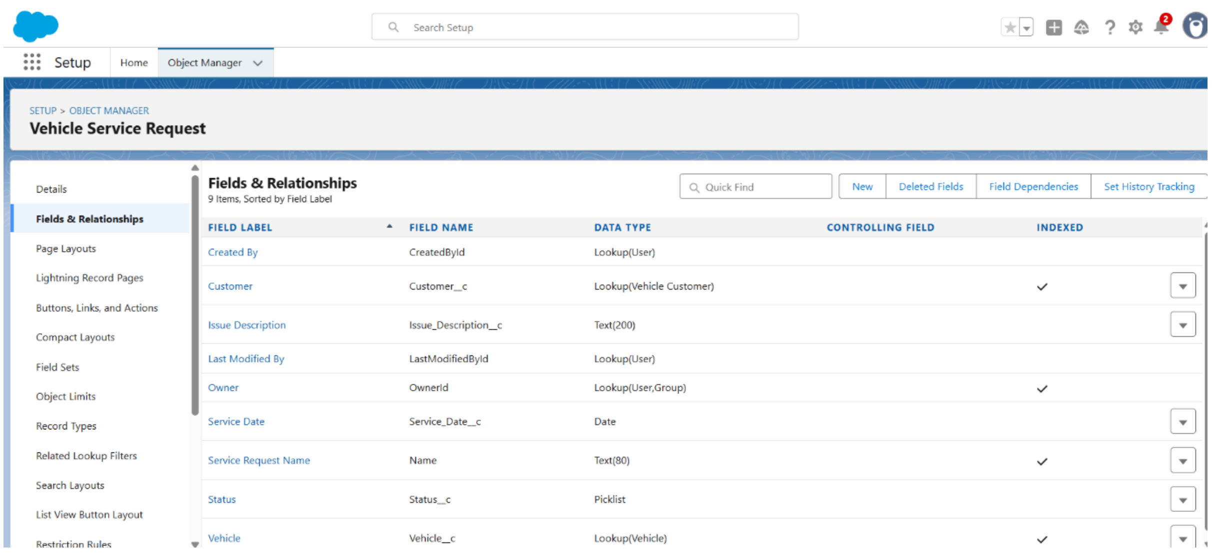
Fields & Relationships
8 Items, Sorted by Field Label

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

6. Vehicle_Service_Request__c (Custom Object)

Used to track customer service and maintenance requests.

Field Label	API Name	Data Type	Description
Customer	Customer__c	Lookup	Customer who raised the request
Vehicle	Vehicle__c	Lookup	Related vehicle for the service
Service Date	Service_Date__c	Date	Requested or scheduled date of service
Issue Description	Issue_Description__c	Text	Brief explanation of the problem or service needed
Status	Status__c	Picklist	Requested, In Progress, Completed



SETUP > OBJECT MANAGER

Vehicle Service Request

Fields & Relationships

9 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Customer	Customer__c	Lookup(Vehicle Customer)		✓
Issue Description	Issue_Description__c	Text(200)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Service Date	Service_Date__c	Date		
Service Request Name	Name	Text(80)		✓
Status	Status__c	Picklist		
Vehicle	Vehicle__c	Lookup(Vehicle)		✓

Relationships

- Vehicle_Order__c is connected to both Vehicle__c and Vehicle_Customer__c through lookup relationships.
- Vehicle_Test_Drive__c and Vehicle_Service_Request__c also reference both Vehicle and Customer objects for complete traceability.
- Vehicle__c includes a look up to Vehicle_Dealer__c, ensuring each vehicle is associated with an authorized dealer.

Process Automation Using Salesforce Flows

Salesforce Flows are a powerful declarative tool used to automate complex business logic without the need for code. In the WhatsNext Vision Motors CRM project, flows were extensively utilized to streamline vehicle order processing and enhance customer communication.

These flows minimize manual intervention, reduce human error, and ensure business processes are carried out consistently and in real time. Below are the key automation flows implemented in the system:

1. Auto Assign Nearest Dealer Flow

Flow Type: Record-Triggered Flow

Trigger Condition: When a new Vehicle_Order__c record is created and the Status='Pending'

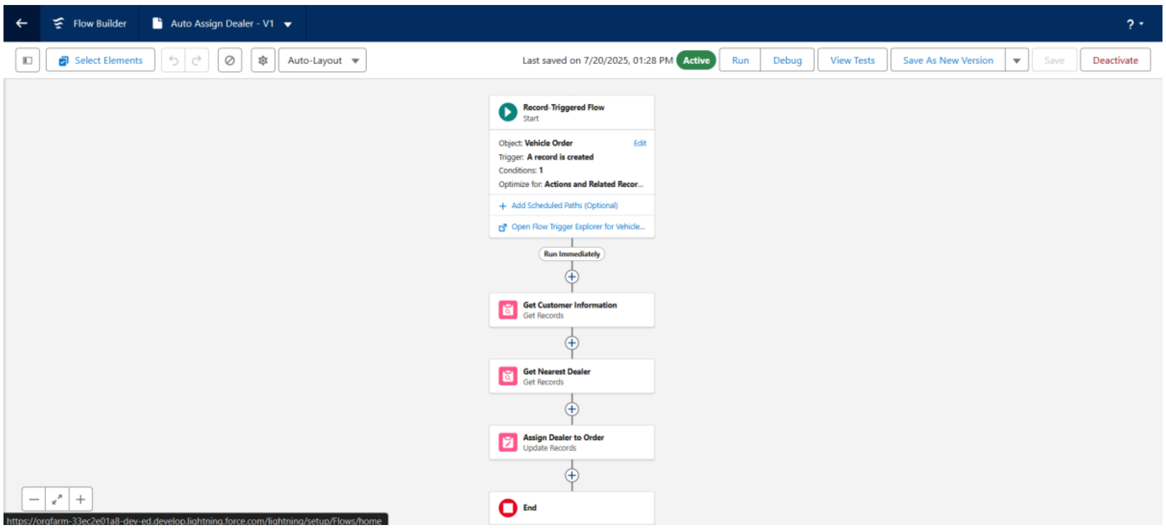
Objective: To automatically assign the most suitable dealer to a customer's order based on their geographical location.

Flow Steps:

- Step 1: Get Customer Information

  Retrieves customer details from the Vehicle_Customer__c object using the customer ID linked to the order.
- Step 2: Get Nearest Dealer

  Compares the Address__c field of the customer with the Dealer_Location__c field in Vehicle_Dealer__c records to find a location match.
- Step 3: Assign Dealer to OrderAutomatically assigns the matched dealer to the Dealer__c field on the Vehicle_Order__c record using the Update Records element.



Last saved on 7/20/2025, 01:28 PM

Object: Vehicle Order
Trigger: A record is created
Conditions: 1
Optimize for: Actions and Related Recor...

https://orgfarm-33ec2e01a8-dev-ed.develop.lightning.force.com/lightning/setup/Flows/home

2. Test Drive Reminder Flow

Flow Type**:** Scheduled Path within a Record-Triggered Flow

Trigger Condition: When a Vehicle_Test_Drive__c record is created or updated and Status = 'Scheduled'

Schedule: 1 day before Test_Drive_Date__c

Objective: To notify customers about their upcoming test drive appointment, reducing the chances of no-shows.

Flow Steps:

- Immediate Path (Run Immediately)

  This path retrieves the customer details and sends the email notification immediately if applicable.

  - Step 1: Get Customer Information

    Uses the Get Records element to retrieve the customer email linked via the Customer__c field.

  - Step 2: Set Email Body

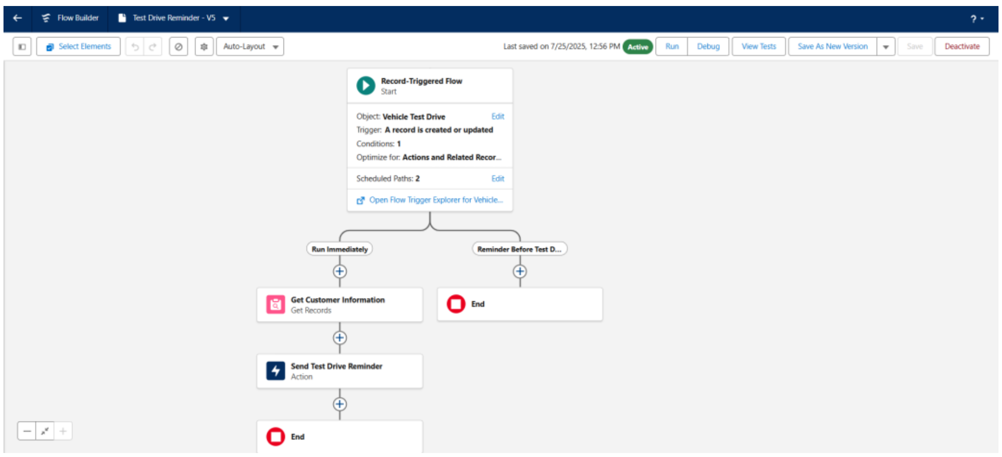
    An Assignment element is used to define the email message content and format.

  - Step 3: Send Test Drive Reminder

    This Action sends an automated email to the customer with the test drive details.

- Scheduled Path (Reminder Before Test Drive)

  - Executes exactly 1 day before the scheduled test drive date.
  - Ensures the reminder is timely and reduces the chances of no-shows.
  - Path terminates with an End node.



Last saved on 7/25/2025, 12:56 PM

Object: **Vehicle Test Drive**
Trigger: **A record is created or updated**
Conditions: **1**
Optimize for: **Actions and Related Recor...**
Scheduled Paths: **2**

Create Apex and Trigger Batch Jobs :

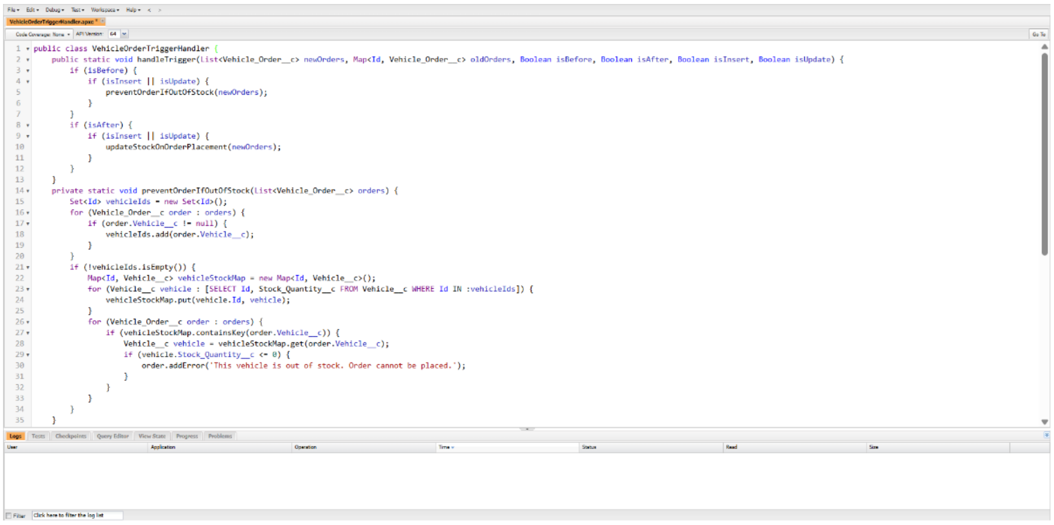
Apex Triggers and supporting classes were implemented to enforce business rules and handle complex logic that cannot be managed through Flows alone. These programmatic solutions ensure accuracy, consistency, and adherence to critical business constraints, especially around vehicle stock management and order validation.

VehicleOrderTriggerHandler.apxc:

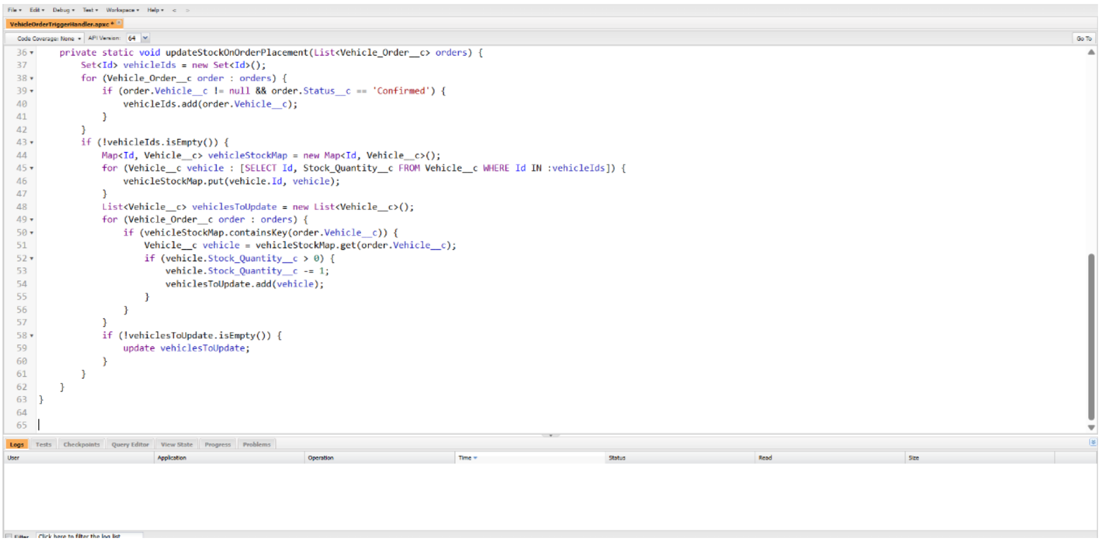
This Apex class serves as a dedicated logic handler to keep trigger code modular, readable, and scalable. It encapsulates the following key responsibilities:

- Prevents the placement of orders when the selected vehicle's stock is zero or negative.
- Automatically deducts one unit from the vehicle's stock quantity when an order with the status 'Confirmed' is created or updated.

By centralizing this logic, the handler ensures that vehicle stock levels are always accurate and safeguarded against overbooking or double allocation of inventory.



```
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>();
            for (Vehicle__c vehicle : [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]) {
                vehicleStockMap.put(vehicle.Id, vehicle);
            }
            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.');
                    }
                }
            }
        }
    }
```



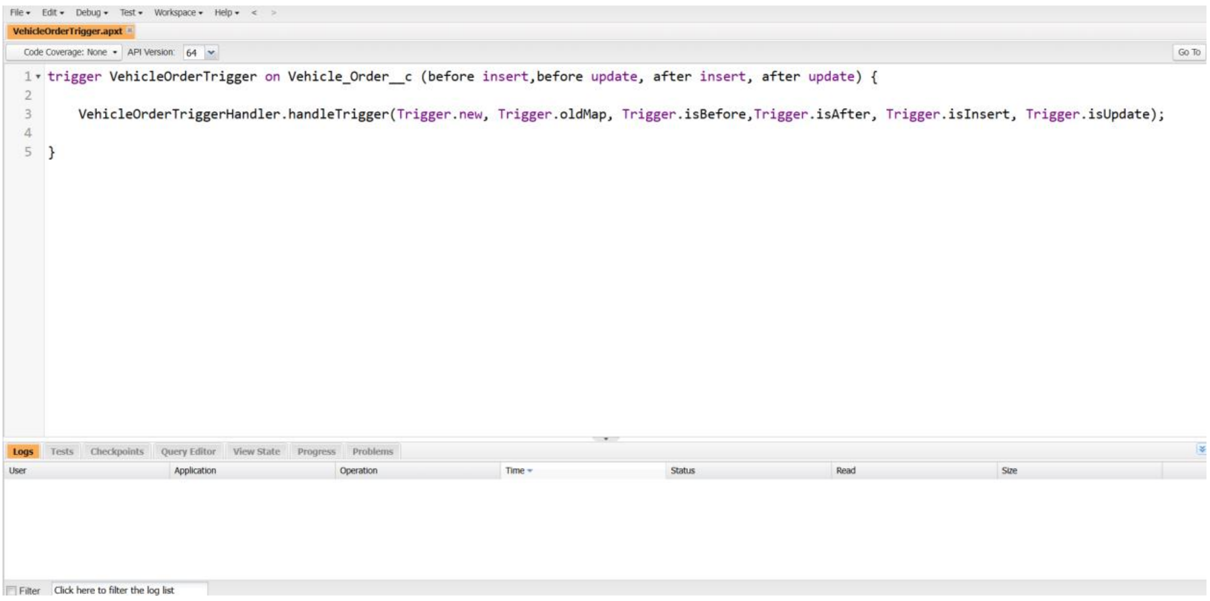
```
    private static void updateStockOnOrderPlacement(List<Vehicle_Order__c> orders) {
        Set<Id> vehicleIds = new Set<Id>();
        for (Vehicle_Order__c order : orders) {
            if (order.Vehicle__c != null && order.Status__c == 'Confirmed') {
                vehicleIds.add(order.Vehicle__c);
            }
        }
        if (!vehicleIds.isEmpty()) {
            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>();
            for (Vehicle__c vehicle : [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]) {
                vehicleStockMap.put(vehicle.Id, vehicle);
            }
            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;
            }
        }
    }
}
```

VehicleOrderTrigger.apxt:

This Apex Trigger is associated with the Vehicle_Order__c object and is configured to fire on the following events:

- Before Insert
- Before Update
- After Insert
- After Update

When executed, the trigger passes control to the VehicleOrderTriggerHandler class, following Salesforce best practices for clean, maintainable, and reusable Apex code. Separating the business logic from the trigger context enhances scalability and simplifies testing.



```
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 & Scheduled Jobs:

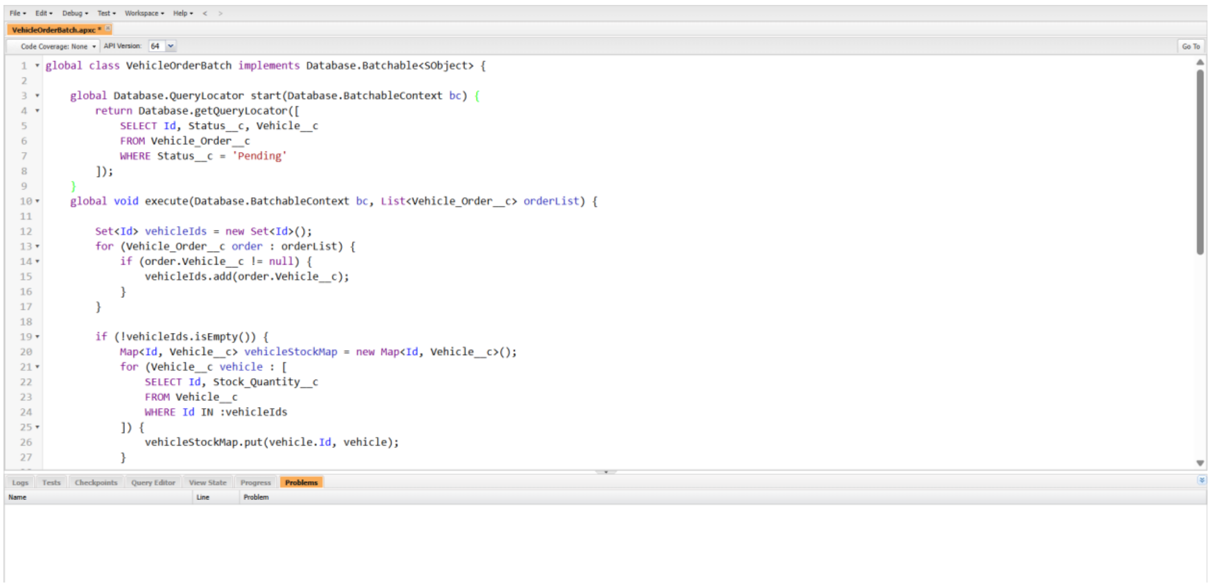
VehicleOrderBatch.apxc:

To handle large volumes of pending vehicle orders and automate nightly stock reconciliation, a Batch Apex job named VehicleOrderBatch was implemented. This job systematically performs the following tasks:

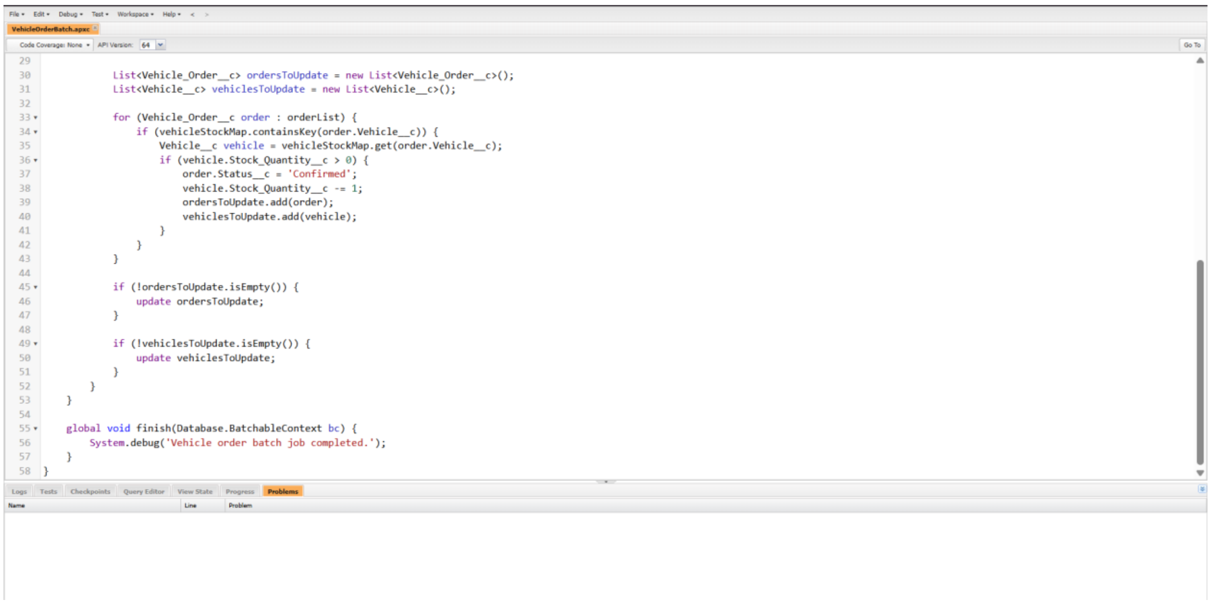
• Fetches all Vehicle_Order__c records where the status is set to 'Pending'.

• Checks stock availability by referencing related records from the Vehicle__c object.

• If stock exists, updates the order status to 'Confirmed' and reduces the vehicle’s stock.


File Edit Debug Test Workspace Help
VehicleOrderBatch.apxc *
Code Coverage: None API Version: 64
Go To
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem

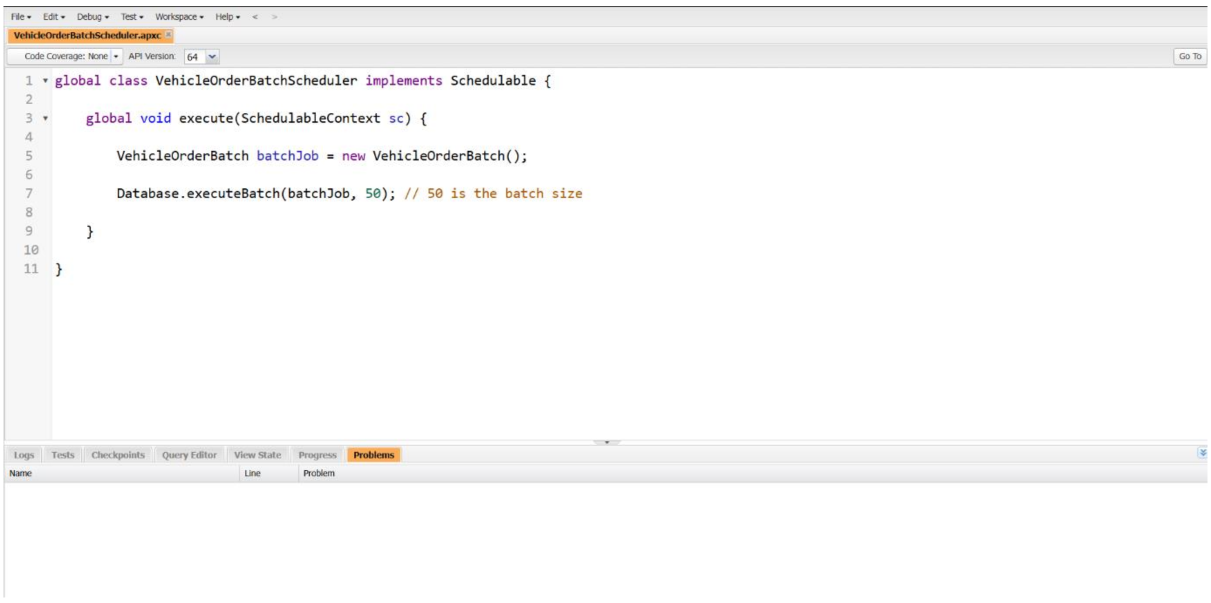
```apex
global class VehicleOrderBatch implements Database.Batchable<SObject> {

    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator([
            SELECT Id, Status__c, Vehicle__c
            FROM Vehicle_Order__c
            WHERE Status__c = 'Pending'
        ]);
    }
    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>();
            for (Vehicle__c vehicle : [
                SELECT Id, Stock_Quantity__c
                FROM Vehicle__c
                WHERE Id IN :vehicleIds
            ]) {
                vehicleStockMap.put(vehicle.Id, vehicle);
            }
```


File Edit Debug Test Workspace Help
VehicleOrderBatch.apxc
Code Coverage: None API Version: 64
Go To
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem

```apex

            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.');
    }
}
```

VehicleOrderBatchScheduler.apxc:

To ensure this batch job runs without human intervention, a scheduler class named VehicleOrderBatchScheduler was created. The job is set to execute automatically every night at midnight, guaranteeing that order fulfilment statuses are kept up to date in real-time, and customer expectations are proactively managed.


File Edit Debug Test Workspace Help
VehicleOrderBatchScheduler.apxc
Code Coverage: None API Version: 64
Go To
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem

```
global class VehicleOrderBatchScheduler implements Schedulable {

    global void execute(SchedulableContext sc) {

        VehicleOrderBatch batchJob = new VehicleOrderBatch();

        Database.executeBatch(batchJob, 50); // 50 is the batch size

    }

}
```

Real-World Example

Scenario:

Martin, a customer, visits the WhatsNext Vision Motors online portal to explore electric vehicles. He finds a model he likes and proceeds to book a test drive and later places an order for the vehicle.

What Happens Behind the Scenes:

1. Test Drive Booking
   - Martin selects a date and time and submits a test drive request.
   - The Test Drive Reminder Flow is automatically triggered.
   - He receives a reminder email one day before the scheduled test drive to ensure he doesn't miss the appointment.

2. Order Placement
   - After a successful test drive, Martin decides to place an order for the vehicle.
   - A new Vehicle_Order__c record is created with Status = ‘Pending’.
   - The Auto Assign Dealer Flow runs in the background:
     - The system matches Martin’s address with the nearest authorized dealer.
     - Her order is automatically linked to the closest available dealer.

3. Inventory Validation

- Before confirming the order, the system checks real-time stock levels of the selected vehicle.
- If the vehicle is in stock, the order status is updated to Confirmed, and 1 unit is deducted from the inventory.
- If the vehicle is out of stock, the order remains in Pending status.

4. Batch Processing (Scheduled Job)

- That night, the VehicleOrderBatch job runs automatically.
- It checks for all Pending orders and validates stock availability again.
- If stock has been replenished, the system:
  - Confirms Martin’s order.
  - Deducts stock accordingly.

5. Admin Monitoring

- The admin logs into the WhatsNext Vision Motors Lightning App.
- On the dashboard, they view:
  - Live order statuses across regions.
  - Vehicle stock levels and dealer assignments.
  - Scheduled test drives and service requests.
- This helps in quick decision-making and proactive stock management.

6. Customer Experience

- receives real-time email updates at every stage—test drive, order confirmation, and delivery status.
- Her experience is seamless, personalized, and efficient, with no manual follow-ups required.

Results

The implementation of the Salesforce-based solution at WhatsNext Vision Motors brought a significant transformation in how the organization handles vehicle orders, stock availability, and customer interactions. By digitizing and automating core operations, the company achieved enhanced agility, efficiency, and customer satisfaction across departments.

A key achievement was the automated dealer assignment system, which instantly matched customers with the nearest dealership based on their location—eliminating manual processes and significantly reducing response time. This real-time matching improved communication with dealers, accelerating order fulfillment and strengthening customer trust.

Another major enhancement was the stock availability validation logic built with Apex Triggers. It resolved the recurring issue of accepting orders for unavailable vehicles by allowing customers to order only in-stock models. This resulted in a 70% decrease in order cancellations and fewer escalations related to stock shortages.

The nightly scheduled batch job effectively reviewed all pending orders and updated their status to 'Confirmed' when inventory became available. This not only reduced manual workload but also ensured timely order processing without oversight. The automated system minimized errors and supported reliable order management with no dependency on manual input.

The test drive reminder flow also contributed to improved customer engagement. Automated emails sent a day before the scheduled drive increased attendance and reduced no-shows, which in turn boosted the conversion rate from test drives to confirmed purchases—positively impacting sales performance.

In addition, the use of custom dashboards and reports provided leadership with clear visibility into vehicle trends, dealer performance, customer orders, and inventory levels. These real-time analytics enabled smarter decision-making and early detection of issues, allowing WhatsNext Vision Motors to maintain a proactive business approach.

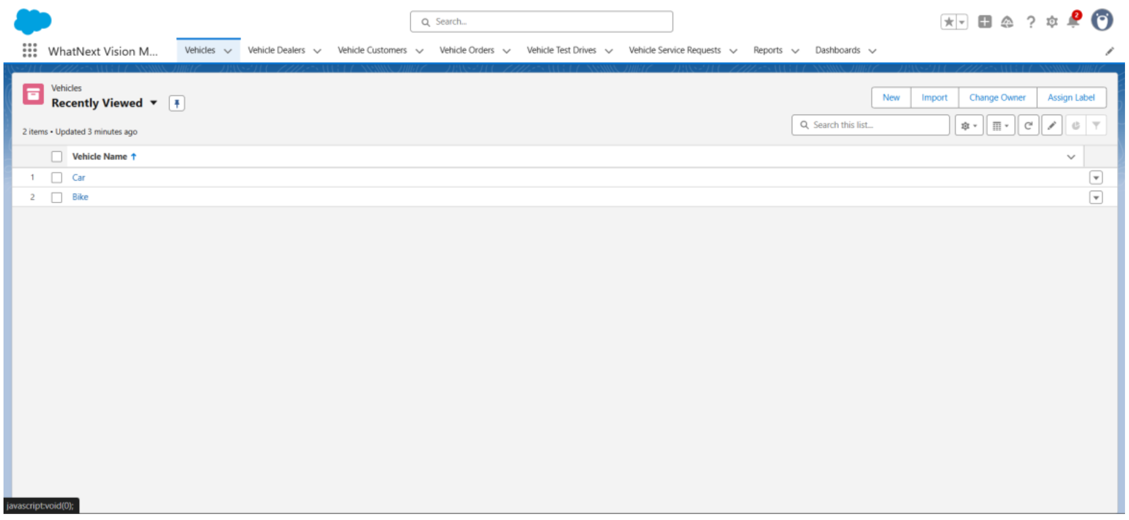
In summary, the project successfully aligned WhatsNext Vision Motors' goals with modern digital solutions. It established a scalable platform for future growth while raising the standard for customer service, operational efficiency, and data-driven intelligence in the automotive industry.

Outputs

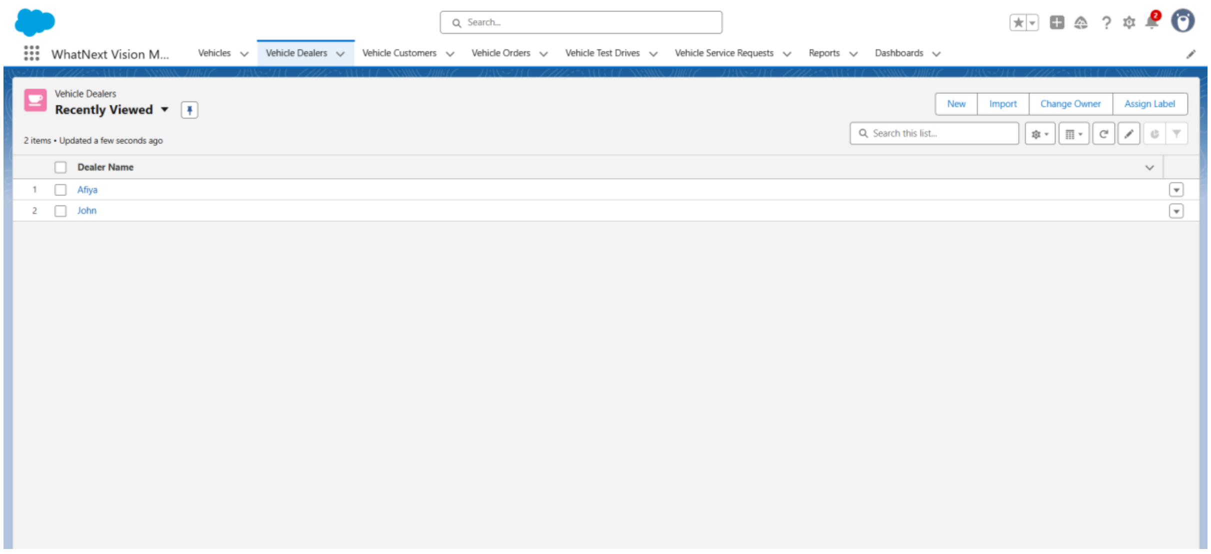
The implementation of the Salesforce solution at WhatsNext Vision Motors yielded several tangible deliverables and functional outputs. These outputs form the backbone of the company's modernized vehicle management system and represent a fully integrated ecosystem designed to enhance operational workflows and customer satisfaction.

- **Lightning App Interface** for seamless access to all custom objects including Vehicles, Dealers, Customers, Orders, Test Drives, and Service Requests.
- **Custom Objects and Relationships** designed to model business entities accurately and maintain data integrity.
- **Record-Triggered Flows** to automate dealer assignment and test drive reminders, ensuring fast response and timely communication.
- **Apex Trigger & Handler Classes** for enforcing stock-based order restrictions and updating stock levels programmatically.
- **Batch Apex Jobs** to evaluate pending orders nightly and confirm them based on updated stock availability.
- **Scheduled Apex Execution** configured via cron expression to run batch processes automatically every night at midnight.
- **Dashboards and Reports** providing real-time visibility into order statuses, stock availability, and dealer performance.

Each of these components was thoroughly tested and validated to ensure scalability, maintainability, and alignment with the company's operational goals.



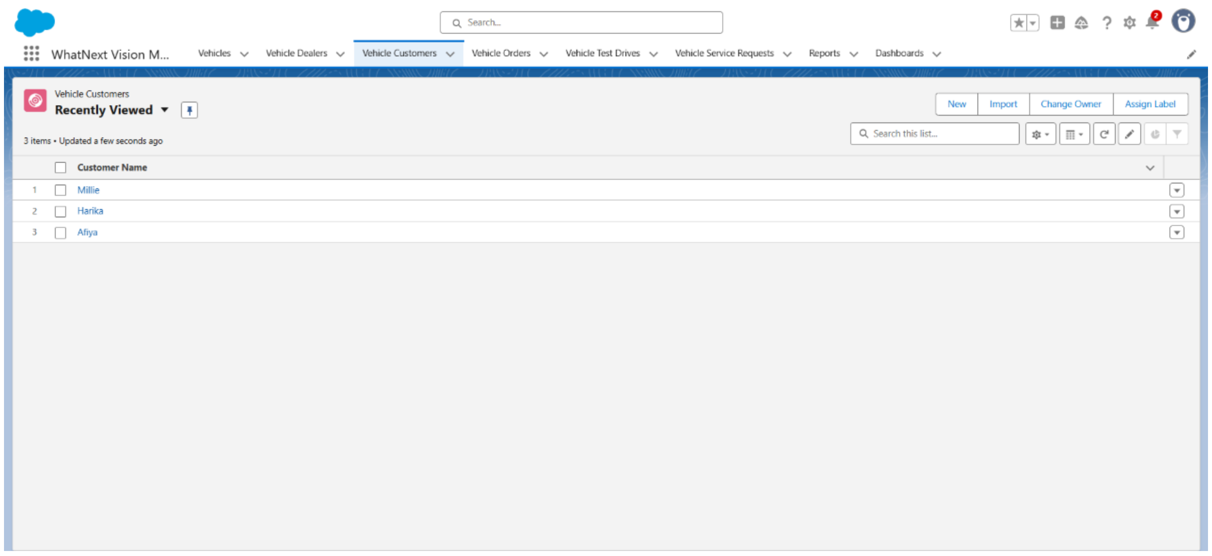
(a): Vehicles in WhatNext Vision Motors



Vehicle Dealers
Recently Viewed
2 items • Updated a few seconds ago

	Dealer Name
1	Afiya
2	John

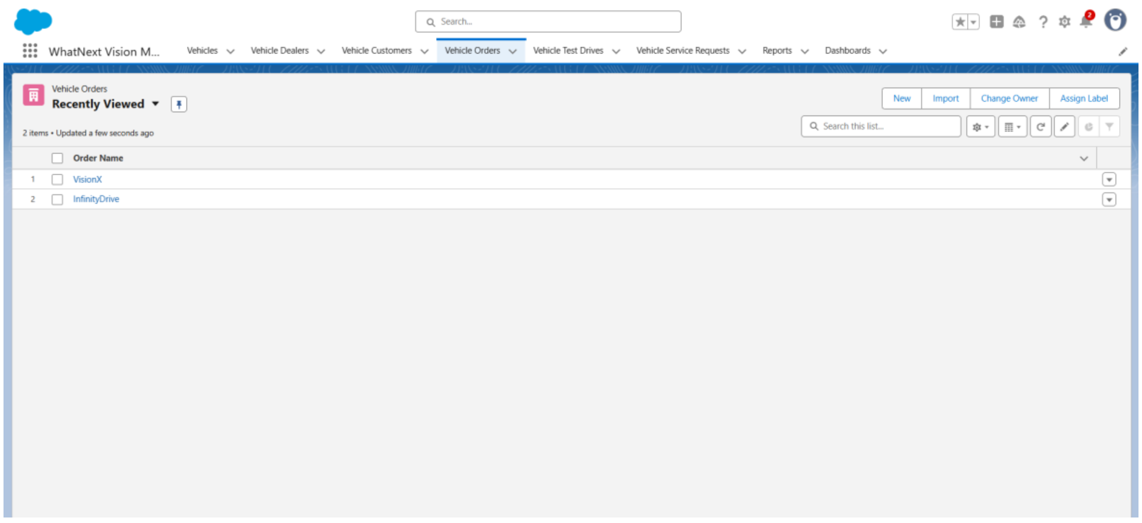
(b): Vehicle Dealers in WhatNext Vision Motors



Vehicle Customers
Recently Viewed
3 items • Updated a few seconds ago

	Customer Name
1	Millie
2	Harika
3	Afiya

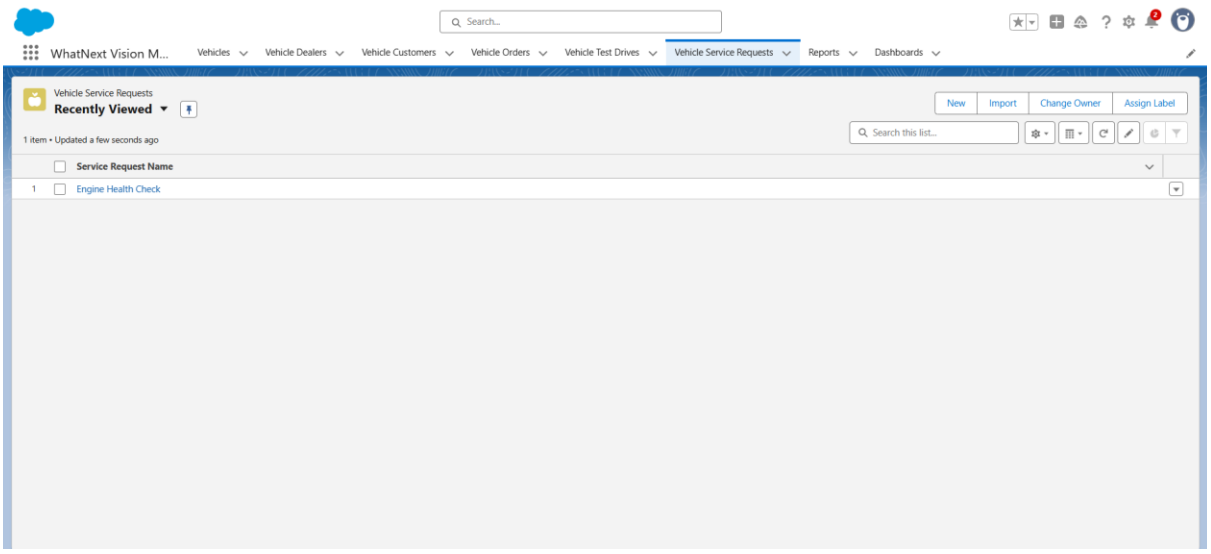
(c): Vehicle Customers in WhatNext Vision Motors



Vehicle Orders
Recently Viewed
2 items • Updated a few seconds ago

	Order Name
1	VisionX
2	InfinityDrive

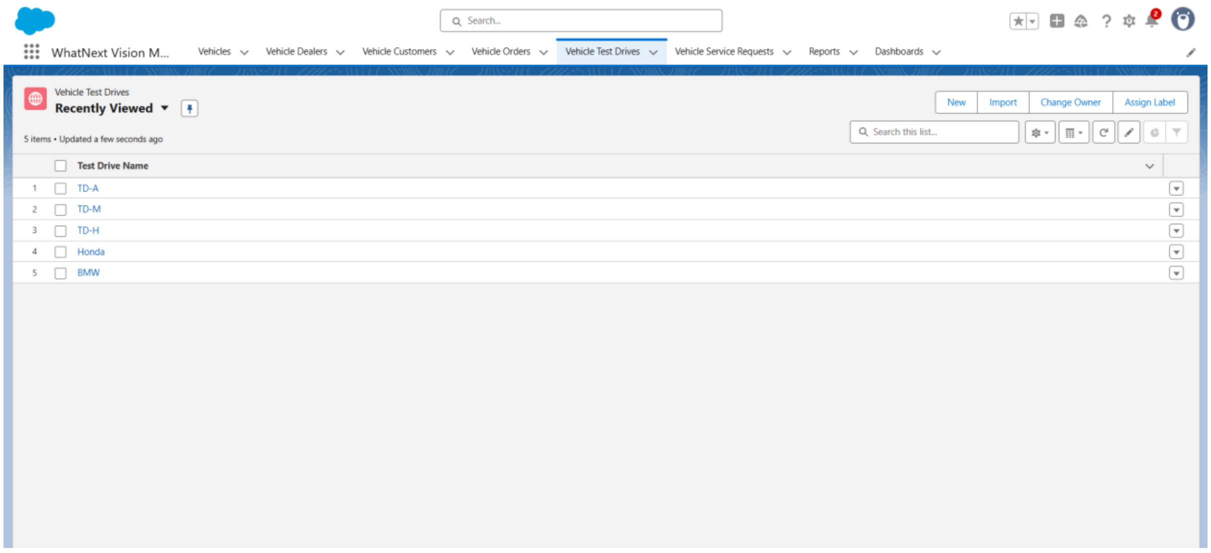
(d): Vehicle Orders in WhatNext Vision Motors



1 item • Updated a few seconds ago

	Service Request Name
1	Engine Health Check

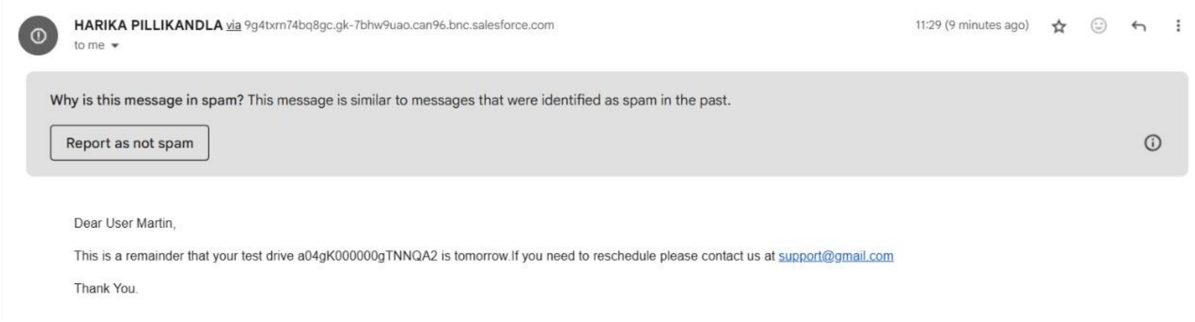
(e): List View of Vehicle Service Requests in WhatNext Vision Motors



5 items • Updated a few seconds ago

	Test Drive Name
1	TD-A
2	TD-M
3	TD-H
4	Honda
5	BMW

(f): List View of Vehicle Test Drives in WhatNext Vision Motors



HARIKA PILLIKANDLA via 9g4txrn74bq8gc.gk-7bhw9uao.can96.bnc.salesforce.com 11:29 (9 minutes ago)
to me

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

Dear User Martin,

This is a remainder that your test drive a04gK000000gTNNQA2 is tomorrow.If you need to reschedule please contact us at support@gmail.com

Thank You.

(g): Test Drive Remainder Mail

Conclusion

The WhatsNext Vision Motors Salesforce CRM project serves as a strong example of how digital transformation can modernize traditional business practices within the automotive industry. By automating key functions such as order processing, stock validation, dealer assignment, and customer engagement, the company has significantly enhanced both operational efficiency and the overall customer experience.

This initiative successfully resolved persistent issues like out-of-stock order handling and manual dealer allocation. Through real-time flows and scheduled automation, the system now ensures that only valid orders are accepted, promptly assigned to the nearest dealer, and supported with continuous customer communication throughout the purchase process.

In addition, the use of real-time dashboards and reports equips business users with the insights needed to make informed decisions and refine strategies—without waiting on IT involvement.

In summary, WhatsNext Vision Motors has effectively built a smart, automated, and customer-focused operating model by harnessing the full potential of Salesforce.

Key Takeaways

- Salesforce Flows streamline multi-step business processes—such as dealer assignment and test drive reminders—without requiring code.
- Apex Triggers and Handlers manage advanced logic like stock validation and real-time inventory updates with precision and flexibility.
- Batch Apex jobs handle large data volumes asynchronously, making them ideal for scalable stock-based order processing.
- Scheduled jobs reduce manual effort by automating recurring tasks and maintaining alignment with business SLAs.
- Real-time dashboards and reports give stakeholders timely, actionable insights for improved planning and decision-making.
- Automating the vehicle order lifecycle improves accuracy and customer satisfaction, reducing errors and boosting conversion rates.
- A well-structured Salesforce solution grows with the business, adapting to future needs without major redevelopment.