

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

ABSTRACT

The Salesforce CRM implementation project at WhatsNext Vision Motors aims to revolutionize the customer ordering and service experience in the automotive industry. By leveraging Salesforce's robust platform capabilities, the project focuses on improving order accuracy, enhancing customer satisfaction, and increasing operational efficiency. Core functionalities include automatic dealer assignment based on customer location, real-time stock validation, and scheduled automation for bulk order status updates. This initiative is designed to minimize manual intervention, reduce processing delays, and provide a seamless customer journey from inquiry to vehicle delivery.

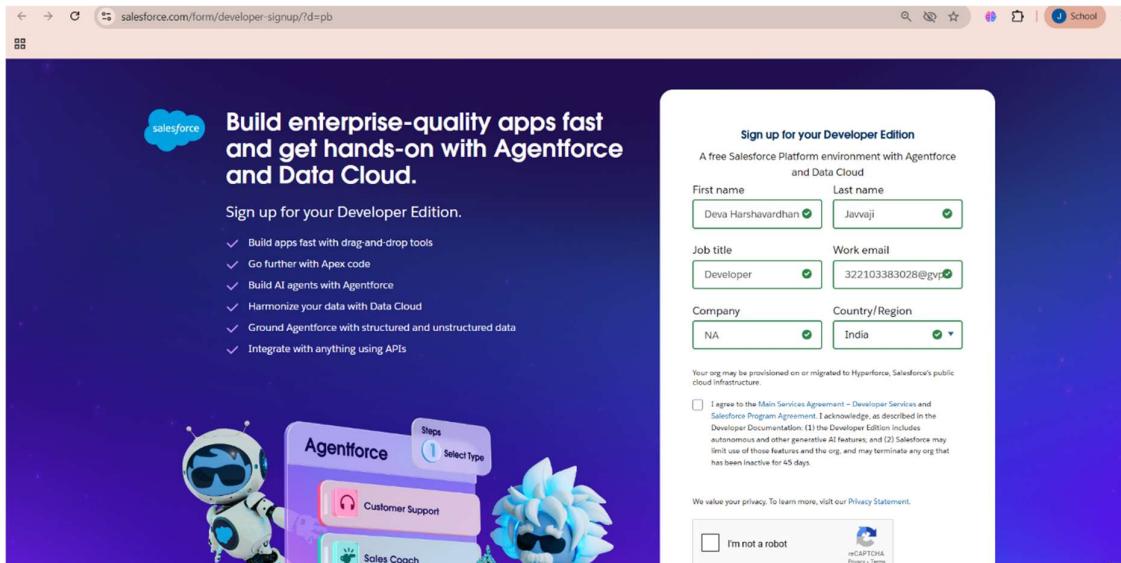
OBJECTIVE

- To implement Salesforce CRM for centralized management of vehicle inventory, dealer details, customer orders, test drives, and service requests.
- To automate key workflows including:
 - Nearest dealer assignment based on customer address.
 - Order restriction if vehicle stock is unavailable.
 - Email notifications for test drives and stock alerts.
- To develop Apex triggers and trigger handlers that enforce stock validation and automatic dealer assignment logic in a modular and maintainable way.
- To build batch Apex jobs for periodically updating stock levels and sending scheduled notifications for inventory management and order status tracking.
- To improve overall customer satisfaction and reduce administrative overhead through efficient process automation.

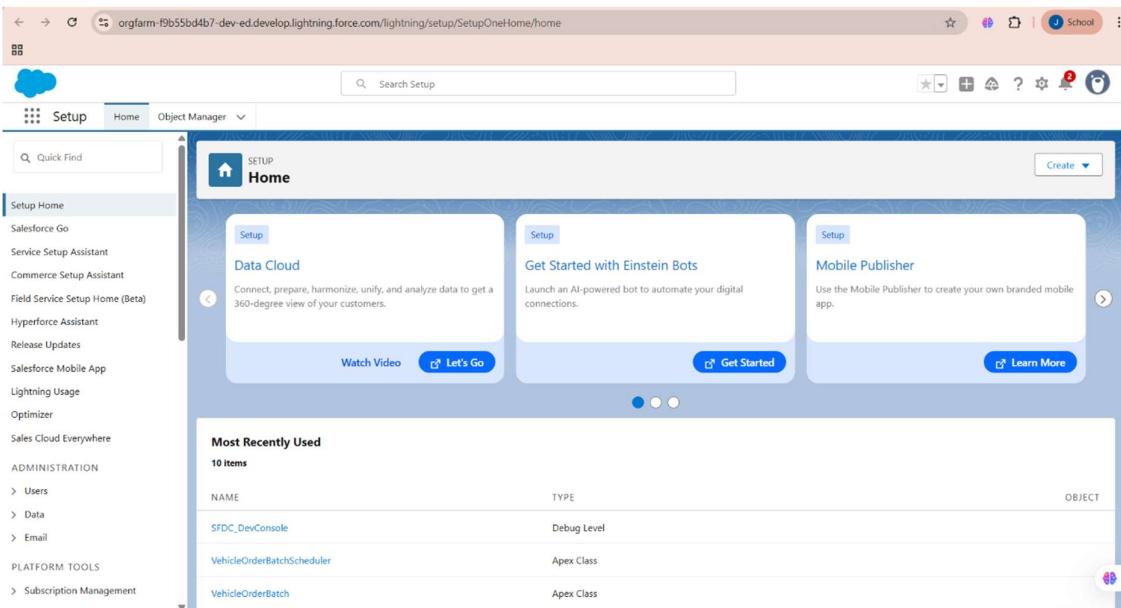
DETAILED EXECUTION OF PROJECT PHASES

Epic-1: Salesforce Credentials Creation

I have created a Salesforce Developer Account using my personal details and set up the necessary credentials to begin my project titled "WhatNext Vision Motors: Shaping the Future of Mobility with Innovation and Excellence." This account will serve as the development environment for building and showcasing my work. With the verifier granted access, they can now monitor and evaluate the progress and contributions I make toward this innovative mobility solution.



After Resetting the password login to the trailhead page.



Epic-2: Data Management-Objects

As part of my project "WhatNext Vision Motors: Shaping the Future of Mobility with Innovation and Excellence," I have successfully created six essential custom objects within Salesforce to manage and structure project data effectively. These objects include Vehicle__c for storing vehicle details, Vehicle_Dealer__c for managing authorized dealer information, Vehicle_Customer__c for capturing customer records, Vehicle_Order__c for tracking vehicle purchases, Vehicle_Test_Drive__c for handling test drive bookings, and Vehicle_Service_Request__c for managing service-related queries. I have established appropriate relationships among these objects to ensure smooth data flow and integration—for example, linking vehicles and dealers to orders, customers to both test drives and orders, and vehicles to service requests. This object structure forms the backbone of the entire project, supporting seamless data management and real-time interaction across all components.

The screenshot displays two screenshots of the Salesforce Object Manager interface.

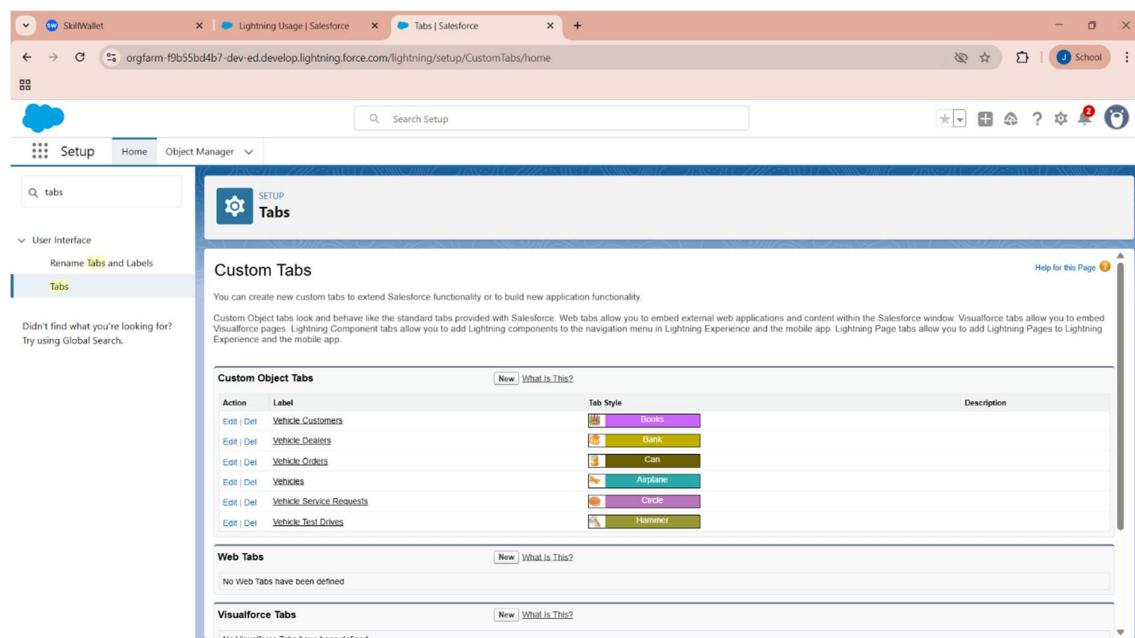
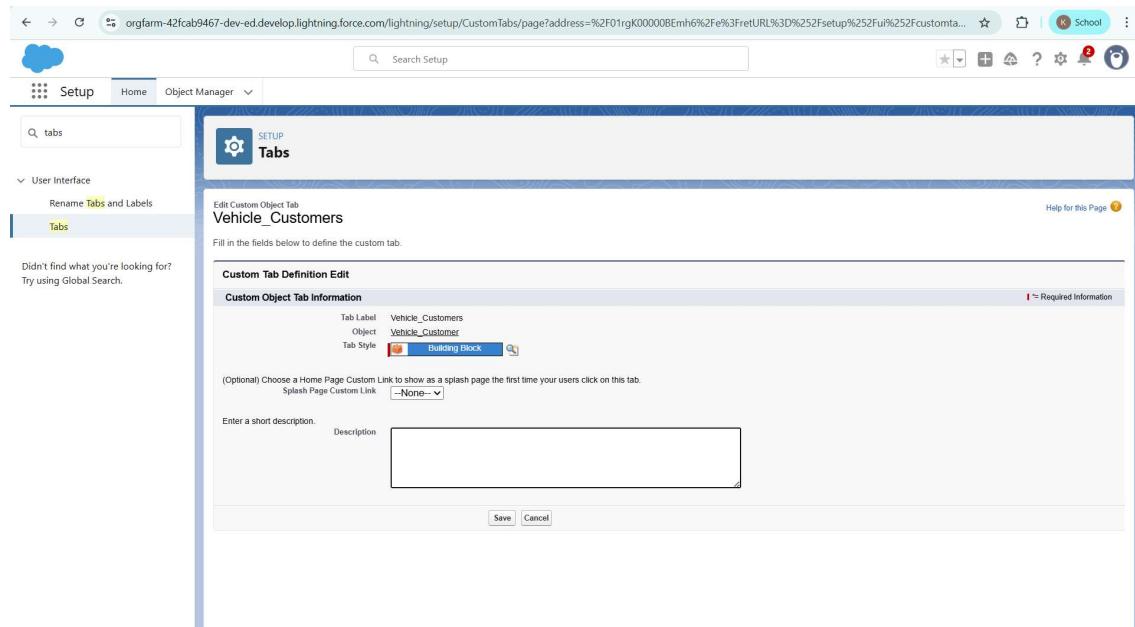
Top Screenshot: Shows the 'Custom Object Definition Edit' page for the 'Vehicle' object. The 'Label' field is set to 'Vehicle' and the 'Plural Label' is set to 'Vehicles'. The 'Object Name' field is also set to 'Vehicle'. A note at the bottom states: 'The object name appears in name labels, API names, related lists, rollups, and search results. For example, the record name for Account is "Account Name" and for Case it is "Case Number". Note that the Record Name field is always'.

Bottom Screenshot: Shows the 'Object Manager' list view with six items. The table columns are: LABEL, API NAME, TYPE, DESCRIPTION, LAST MODIFIED, and DEPLOYED. The items listed are:

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Vehicle	Vehicle__c	Custom Object		7/23/2025	✓
Vehicle Customer	Vehicle_Customer__c	Custom Object	Stores customer details	7/23/2025	✓
Vehicle Dealer	Vehicle_Dealer__c	Custom Object		7/23/2025	✓
Vehicle Order	Vehicle_Order__c	Custom Object	Tracks vehicle purchases	7/23/2025	✓
Vehicle Service Request	Vehicle_Service_Request__c	Custom Object	Tracks vehicle servicing requests	7/23/2025	✓
Vehicle Test Drive	Vehicle_Test_Drive__c	Custom Object		7/23/2025	✓

Epic-3: Data Management-Tabs

To enhance accessibility and usability within the Salesforce environment, I created custom tabs for all six custom objects used in the project: Vehicle__c, Vehicle_Dealer__c, Vehicle_Customer__c, Vehicle_Order__c, Vehicle_Test_Drive__c, and Vehicle_Service_Request__c. These tabs allow users to easily navigate, view, and manage records associated with each object directly from the Salesforce interface. By setting up these custom tabs, I ensured that all relevant data is readily available for stakeholders, streamlining the user experience and improving overall efficiency in managing vehicle-related processes within the project.



Epic-4: Data Management-App Manager

As part of organizing and streamlining the user experience in my Salesforce project, I created a Lightning App named "WhatNext Vision Motors" to serve as a centralized workspace for managing all aspects of the vehicle business process. This app integrates all key components of the project—including Vehicles, Dealers, Customers, Orders, Test Drives, and Service Requests—into a single, user-friendly interface. I included essential navigation items and custom objects to ensure quick access to relevant records and functionalities. Additionally, I assigned the System Administrator profile to ensure full access and control over the app's features. This app provides a cohesive and efficient way for users to interact with the system, improving workflow and data visibility across all departments involved in the mobility solution.

The image contains two screenshots of the Salesforce interface. The top screenshot shows the 'Navigation Items' configuration in the Lightning App Builder. It displays a list of available items on the left and selected items on the right. Available items include Accounts, Activation Targets, Activation, All Sites, Alternative Payment Methods, Analytics, App Launcher, Appointment Categories, Appointment Invitations, Approval Requests, and Approval Submission Details. Selected items include Vehicle Customers, Vehicle Dealers, Vehicle Orders, Vehicle Service Requests, Vehicle Test Drives, Vehicles, Reports, and Dashboards. The bottom screenshot shows a custom object named 'Vehicle Customer' with a list view titled 'Recently Viewed'. The list contains three items: Ashok, Deva, and Sandeep. The interface includes standard Salesforce navigation and search tools.

Epic-5: Data Management-Fields

As part of building the "**WhatNext Vision Motors**" Salesforce application, I created six custom objects with all the necessary fields to handle business operations like vehicle management, customer tracking, dealer coordination, orders, test drives, and service requests. Each object was carefully structured with appropriate field types such as Text, Picklist, Lookup, Date, Currency, and Auto Number to ensure accurate and efficient data storage.

1. Vehicle_c:

Vehicle_Name__c, Vehicle_Model__c, Stock_Quantity__c, Price__c, Dealer__c, Status__c

The screenshot shows the Salesforce Setup interface for the "Vehicle" object. The left sidebar lists various setup options like Page Layouts, Lightning Record Pages, and Field Sets. The main content area is titled "Fields & Relationships" and displays a table of fields. The table columns are FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The fields listed are:

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

2. Vehicle_Dealer_c:

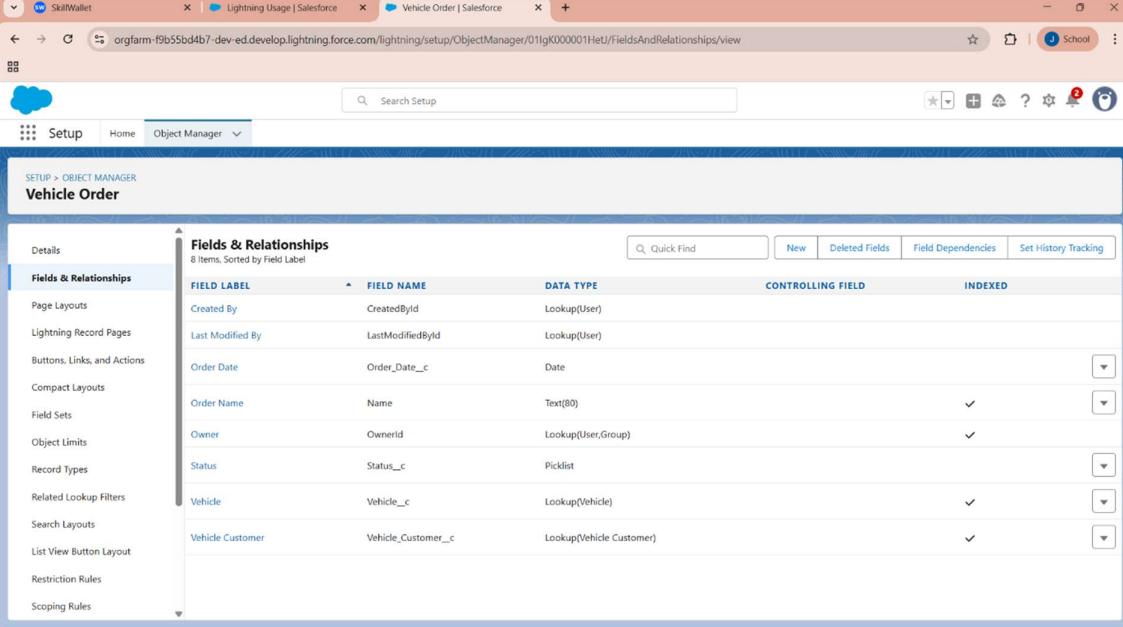
Dealer_Name__c, Dealer_Location__c, Dealer_Code__c, Phone__c, Email__c

The screenshot shows the Salesforce Setup interface for the "Vehicle Dealer" object. The left sidebar lists various setup options. The main content area is titled "Fields & Relationships" and displays a table of fields. The table columns are FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The fields listed are:

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

3. Vehicle_Order__c:

Customer__c, Vehicle__c, Order_Date__c, Status__c

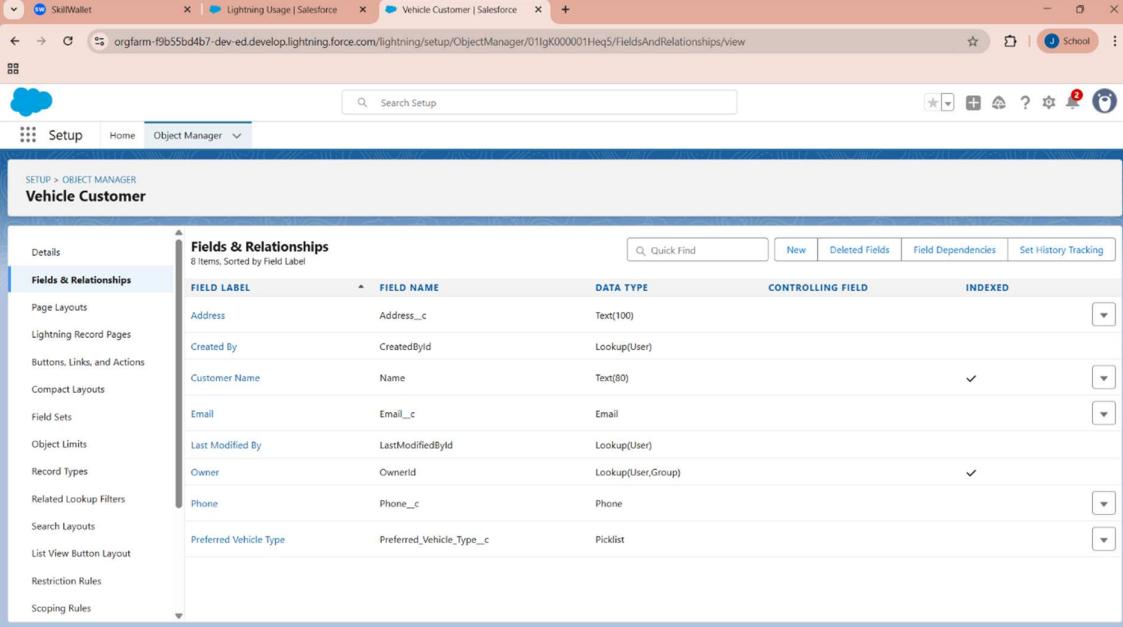


The screenshot shows the Salesforce Object Manager interface for the 'Vehicle Order' object. The left sidebar lists various setup options like Page Layouts, Lightning Record Pages, and Field Sets. The main content area is titled 'Fields & Relationships' and displays a table of fields. The table has columns for FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The indexed column contains several checkmarks.

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

4. Vehicle_Customer__c:

Customer_Name__c, Email__c, Phone__c, Address__c, Preferred_Vehicle_Type__c

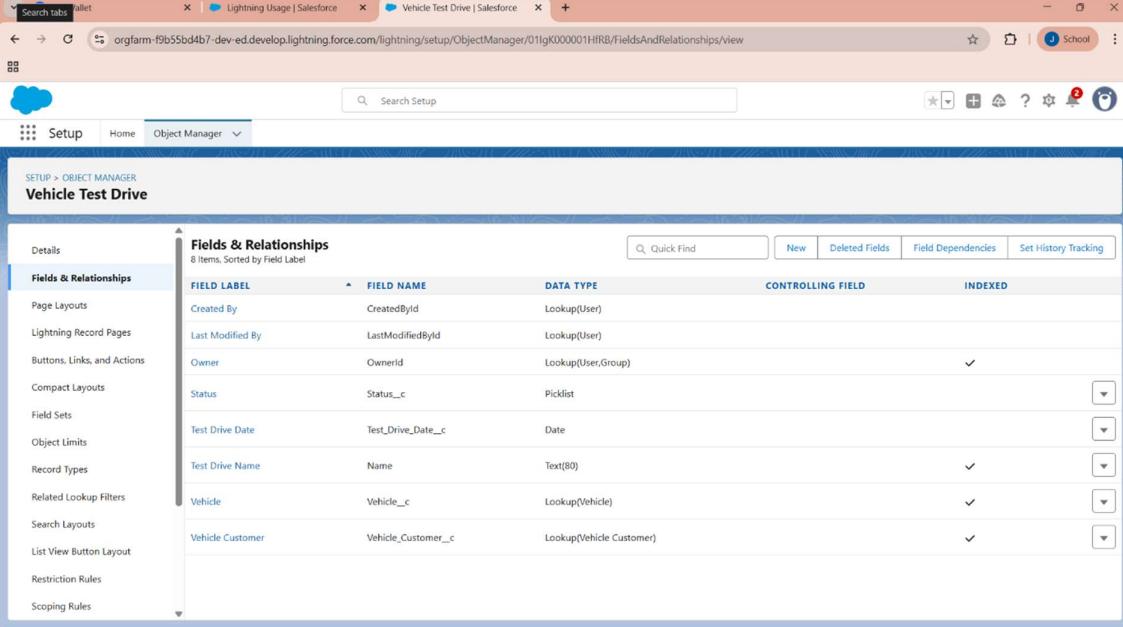


The screenshot shows the Salesforce Object Manager interface for the 'Vehicle Customer' object. The left sidebar lists various setup options. The main content area is titled 'Fields & Relationships' and displays a table of fields. Similar to the previous screenshot, it includes a table with columns for FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. Checkmarks are present in the indexed column.

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Text(100)		
Created By	CreatedById	Lookup(User)		
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		

5. Vehicle_Test_Drive_c:

Customer_c, Vehicle_c, Test_Drive_Date__c, Status_c

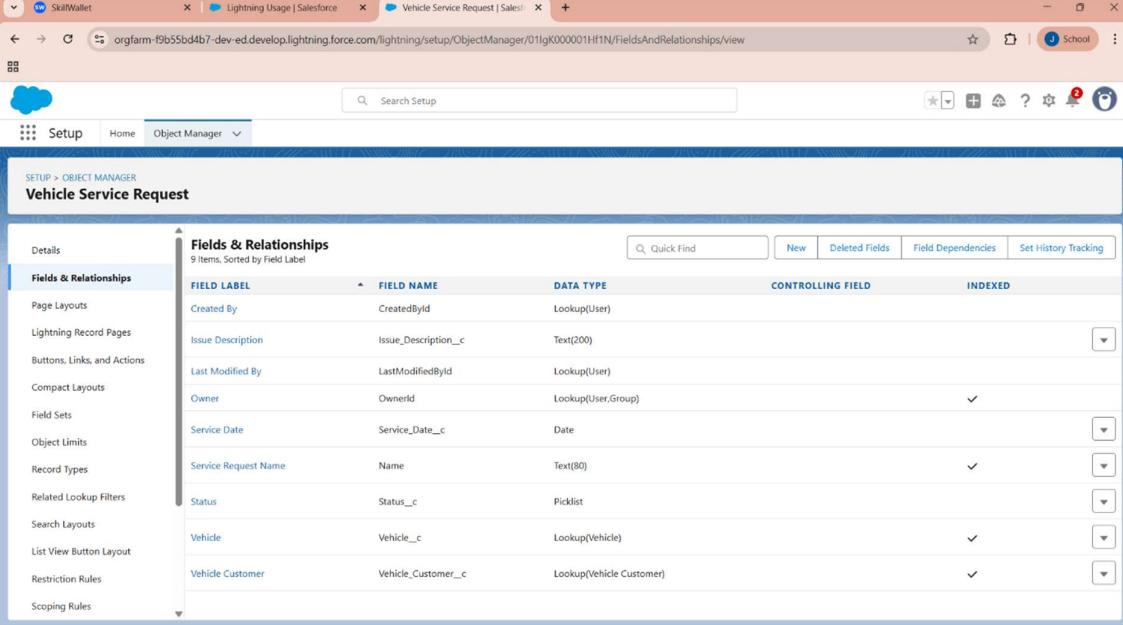


The screenshot shows the Salesforce Setup interface with the title "Vehicle Test Drive". On the left, there's a sidebar with options like Details, Fields & Relationships, Page Layouts, Lightning Record Pages, etc. The main area is titled "Fields & Relationships" and lists 8 items. The table has columns for FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED.

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

6. Vehicle_Service_Request_c:

Customer_c, Vehicle_c, Service_Date__c, Issue_Description__c



The screenshot shows the Salesforce Setup interface with the title "Vehicle Service Request". The sidebar and table structure are identical to the previous screenshot, listing 9 items under "Fields & Relationships".

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
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)		✓
Vehicle Customer	Vehicle_Customer_c	Lookup(Vehicle Customer)		✓

These custom fields enable full tracking of all interactions and transactions within the system. By designing them logically and linking them through relationships, the app provides a seamless experience for managing every aspect of the vehicle sales and service lifecycle.

Epic-6: Automation

As part of automating key processes in the "**WhatNext Vision Motors**" Salesforce application, I created two **Record-Triggered Flows** to enhance efficiency and customer engagement.

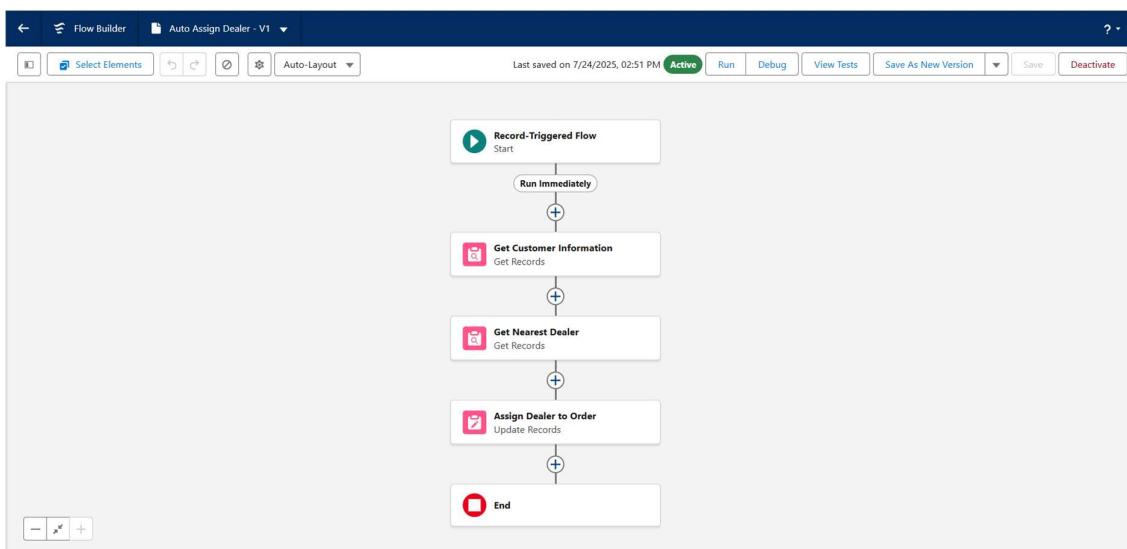
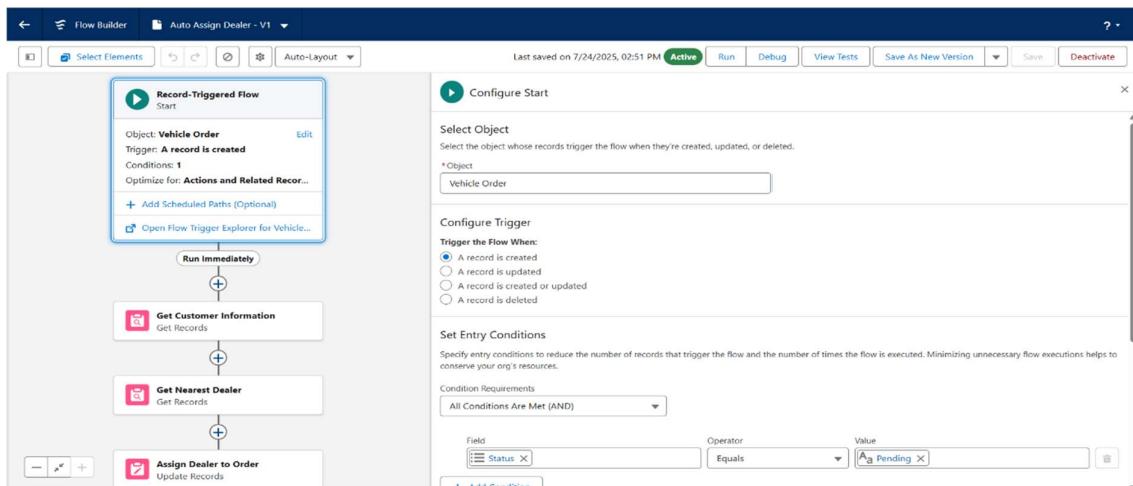
1. Assign Nearest Dealer to Customer:

This flow is triggered when a new customer record is created. It automatically identifies and assigns the nearest authorized dealer based on the customer's location. This ensures a more personalized experience and helps in streamlining the order and service process by connecting customers with the most relevant dealer.

2. Send Test Drive Reminder Email:

This flow is triggered when a new test drive record is created. It sends an automated email reminder to the customer with details about their scheduled test drive. This improves communication, reduces no-shows, and ensures that customers are well-informed and engaged with the brand.

Tested the flow by creating the customer



The screenshot shows a Gmail inbox with 1,255 messages. A yellow warning bar at the top of the message area says "Be careful with this message. This message appears to be sent from your account but Gmail couldn't verify this. Someone might be impersonating your account. If you're not sure the message is from you, use caution when clicking links, downloading attachments, or replying with personal information." It includes "Report spam" and "Looks safe" buttons. The message itself is from "Deva Harshavardhan Javvaji" and is titled "Reminder: Your Test Drive is Scheduled!". The message content is:

Dear Deva,

This is a reminder that your test drive for the Sedan is scheduled on July 25, 2025.

We look forward to seeing you!

- Vision Motors Team

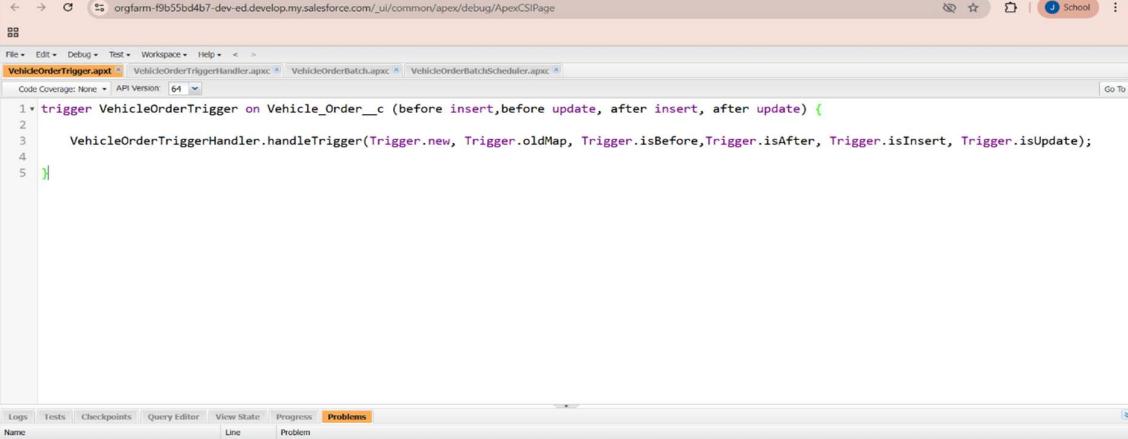
At the bottom are "Reply" and "Forward" buttons.

Epic-7: Apex and Batch Class

To automate the **vehicle availability tracking and order processing workflow** in the *WhatNext Vision Motors* project, I implemented Apex-based automation using **triggers, handler classes, batch jobs, and schedulers**. These components ensure that vehicle orders are processed automatically and efficiently on a daily basis, reducing manual intervention and increasing accuracy.

◆ VehicleOrderTrigger

- This is the **Apex trigger** written on the `Vehicle_Order__c` object.
- It is responsible for executing logic whenever a vehicle order is created or updated.
- The trigger delegates its logic to the handler class to keep the code modular and maintainable.



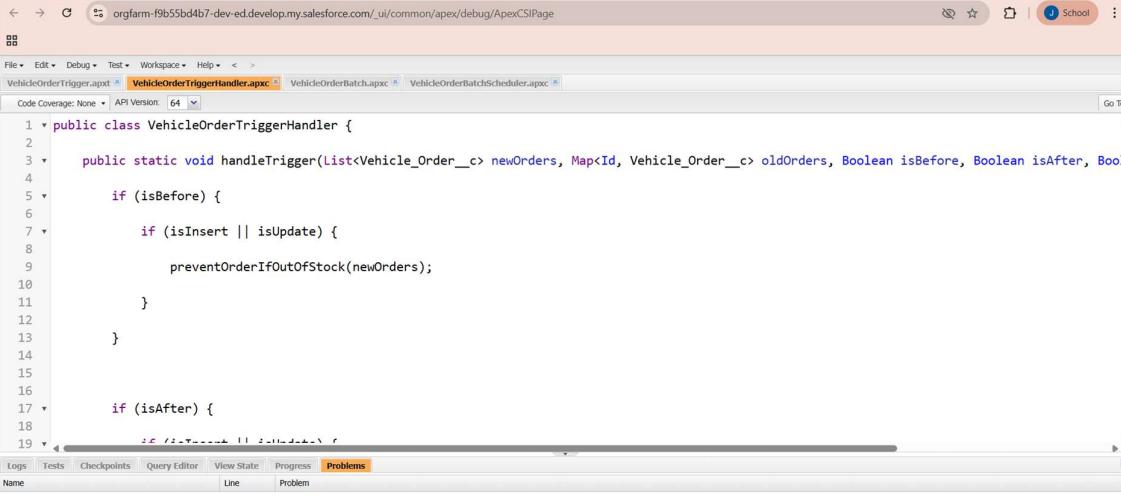
The screenshot shows the Salesforce Apex editor interface. The title bar indicates the URL is `orgfarm-f9b55bd4b7-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. The tabs at the top show `VehicleOrderTrigger.apxc` (which is the active tab), `VehicleOrderTriggerHandler.apxc`, `VehicleOrderBatch.apxc`, and `VehicleOrderBatchScheduler.apxc`. The code editor contains the following Apex trigger:

```
1 * trigger VehicleOrderTrigger on Vehicle_Order__c (before insert,before update, after insert, after update) {
2
3     VehicleOrderTriggerHandler.handleTrigger(Trigger.new, Trigger.oldMap, Trigger.isBefore,Trigger.isAfter, Trigger.isInsert, Trigger.isUpdate);
4
5 }
```

The bottom navigation bar includes tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Problems tab is currently selected, indicated by an orange background.

◆ VehicleOrderTriggerHandler

- This class contains the **business logic** related to the `VehicleOrderTrigger`.
- It checks vehicle availability, updates order status, and manages relationships with vehicles and dealers.



The screenshot shows the Salesforce Apex editor interface. The title bar indicates the URL is `orgfarm-f9b55bd4b7-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. The tabs at the top show `VehicleOrderTrigger.apxc`, `VehicleOrderTriggerHandler.apxc` (which is the active tab), `VehicleOrderBatch.apxc`, and `VehicleOrderBatchScheduler.apxc`. The code editor contains the following Apex class:

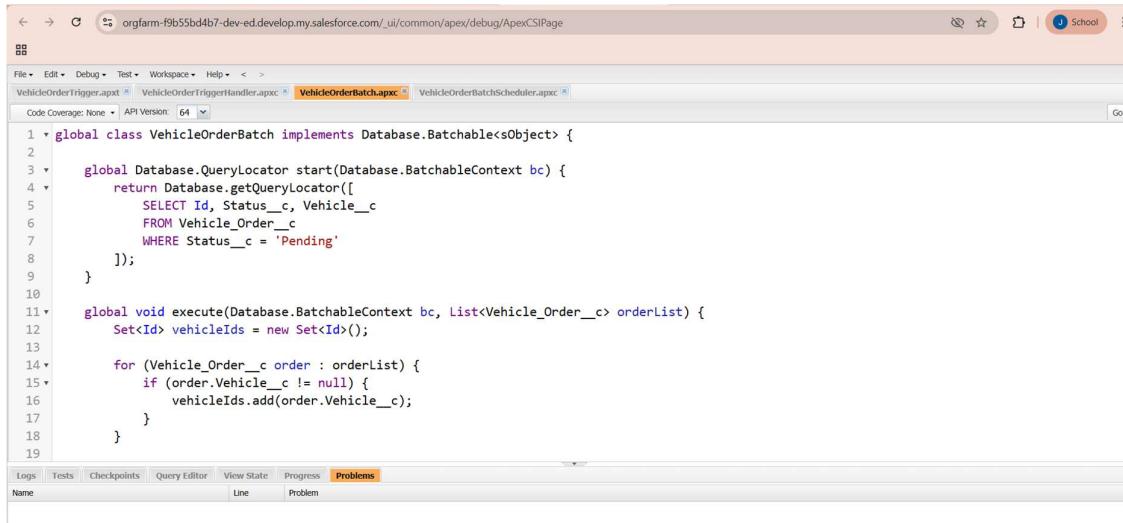
```
1 * public class VehicleOrderTriggerHandler {
2
3     public static void handleTrigger(List<Vehicle_Order__c> newOrders, Map<Id, Vehicle_Order__c> oldOrders, Boolean isBefore, Boolean isAfter, Boolean isInsert, Boolean isUpdate) {
4
5         if (isBefore) {
6
7             if (isInsert || isUpdate) {
8
9                 preventOrderIfOutOfStock(newOrders);
10            }
11        }
12    }
13
14
15    if (isAfter) {
16
17        if (isInsert || isUpdate) {
18
19            preventOrderIfOutOfStock(newOrders);
20        }
21    }
22 }
```

The bottom navigation bar includes tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Problems tab is currently selected, indicated by an orange background.

- Keeping logic in this separate handler promotes **clean code architecture** and **reusability**.

◆ VehicleOrderBatch

- This is a **batch class** that processes large numbers of vehicle orders in bulk.
- It queries all pending vehicle orders and performs actions such as assigning vehicles, updating availability, or marking orders as processed.
- Batch processing is used here to handle **large data volumes efficiently** and without hitting governor limits.



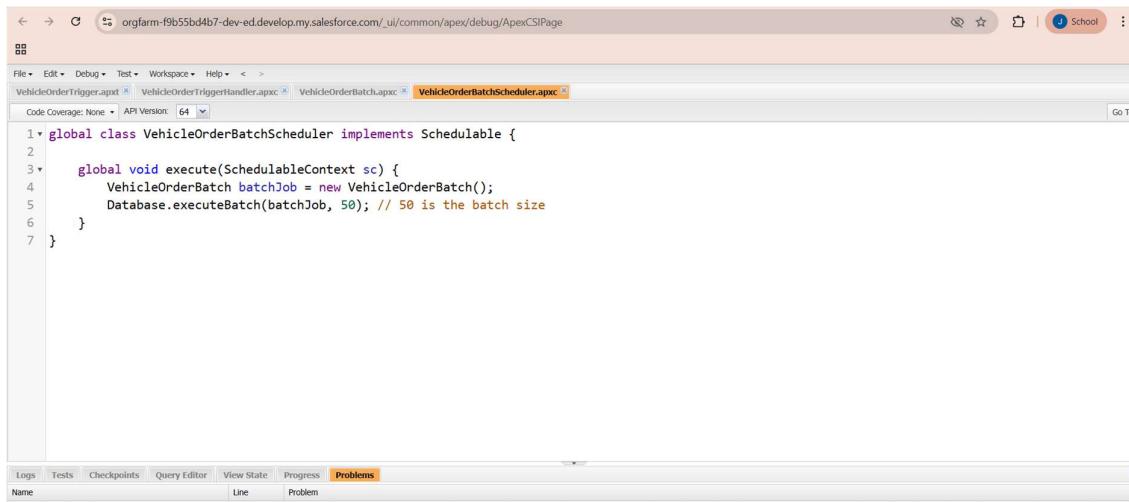
```

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
14        for (Vehicle_Order__c order : orderList) {
15            if (order.Vehicle__c != null) {
16                vehicleIds.add(order.Vehicle__c);
17            }
18        }
19    }

```

◆ VehicleOrderBatchScheduler

- This class is used to **schedule the batch job to run automatically** at a specific time.
- It leverages the System.schedule method to ensure the VehicleOrderBatch job runs every day without manual triggering.



```

1 * global class VehicleOrderBatchScheduler implements Schedulable {
2
3     global void execute(SchedulableContext sc) {
4         VehicleOrderBatch batchJob = new VehicleOrderBatch();
5         Database.executeBatch(batchJob, 50); // 50 is the batch size
6     }
7 }

```

◆ Scheduled Job Monitoring

- Once the scheduler is deployed, you can monitor the scheduled job from **Setup > Scheduled Jobs** in Salesforce.
- This allows tracking of batch job execution, checking for errors, and ensuring it runs as intended.

The screenshot shows the Salesforce Setup interface with the search bar set to 'schedul'. The left sidebar is expanded to show 'Feature Settings' and 'Jobs'. Under 'Jobs', 'Scheduled Jobs' is selected. The main content area is titled 'All Scheduled Jobs' and displays a table of scheduled jobs. The table includes columns for Action, Job Name, Submitted By, Started, Next Scheduled Run, Type, and Cron Trigger ID. The table shows four scheduled Apex jobs:

Action	Job Name	Submitted By	Started	Next Scheduled Run	Type	Cron Trigger ID
Manage Del Pause Job	Daily Vehicle Order Processing	Jayavil Deva Harshvardhan	7/27/2025, 3:19 AM	7/28/2025, 12:00 PM	Scheduled Apex	08egK000000858m0
Del	Analytics Data Loader Job for Org: 000gk0000007PXRO	User_Integration	7/15/2025, 2:50 AM	7/28/2025, 12:31 AM	Autonomous Data Loader Job	08egK0000007Q2a7
	Program Milestone Computation Cron Job	Process, Automated	7/15/2025, 2:50 AM	7/28/2025, 6:59 AM	Program Milestone Computation Cron Job	08egK0000007Q2a5
	Program Status Update Cron Job	Process, Automated	7/15/2025, 2:50 AM	7/27/2025, 8:01 PM	Program Status Update Cron Job	08egK0000007Q2a6

Project Demonstration – Functional Walkthrough with Sample Data

Sample Data Setup:

Vehicle Customers:

- Customer-1
 - Customer Name : Deva Harshavardhan
 - Email : 322103383028@gvpce.ac.in
 - Phone : 9059090041
 - Address : Visakhapatnam
- Customer-2
 - Customer Name : Deva
 - Email : scanner@gmail.com
 - Phone : 8511224595
 - Address : Guntur

Vehicle Dealer:

- Dealer-1
 - Dealer Name : Harsha
 - Dealer Location : Visakhapatnam
 - Phone : 6300245951
 - Email : update@gmail.com
- Dealer-2
 - Dealer Name : poorna
 - Dealer Location : Hyderabad
 - Phone : 9059091141
 - Email : apex@gmail.com

Vehicles:

- Vehicle-1
 - Vehicle Name : RX100
 - Vehicle Model : SUV
 - Stock Quantity : 5
 - Vehicle Dealer : Deva
 - Price : 100000
 - Status : Available
- Vehicle-2
 - Vehicle Name : AUDI
 - Vehicle Model : Sedan
 - Stock Quantity : 0
 - Vehicle Dealer : Naga
 - Price : 100000
 - Status : Out of Stock

Vehicle Orders:

- Vehicle-1
 - Vehicle Order Name :car
 - Vehicle Customer : porrna
 - Vehicle : RX100
 - Order_Date :7/29/2025
 - Status :Pending
- Vehicle-1
 - Vehicle Order Name :Bike
 - Vehicle Customer : Naga
 - Vehicle : RX100
 - Order_Date :7/30/2025
 - Status :Pending

Flow Execution

After the above data is created, here's what happens automatically:

Dealer Assignment Flow:

When a new Vehicle Order is created with status Pending, the flow:

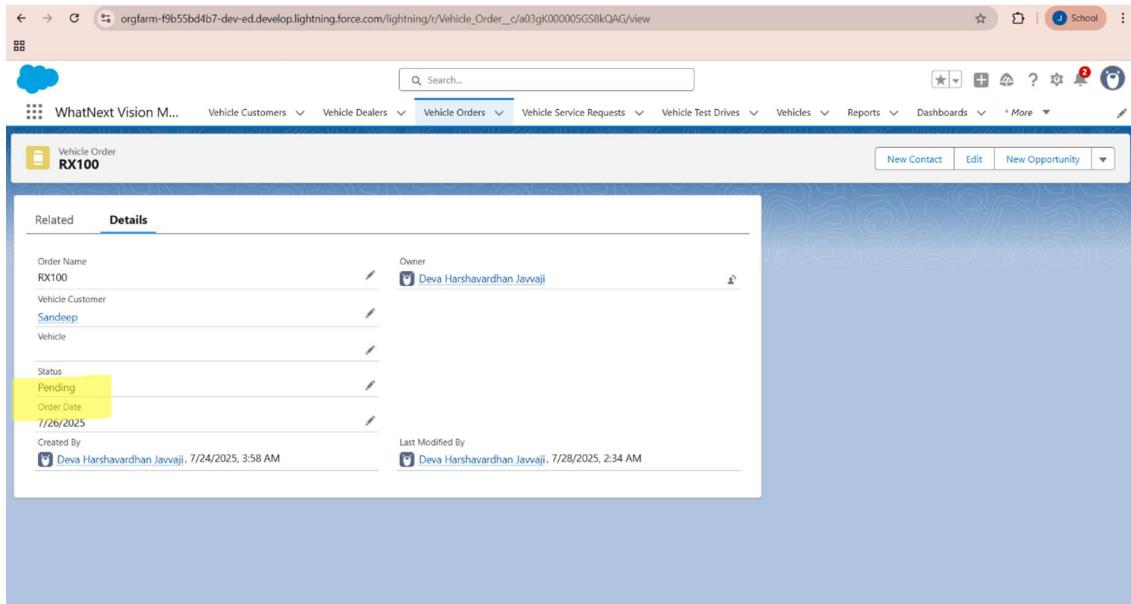
- Fetches the selected Vehicle__c
- Retrieves its associated Dealer__c
- Assigns that dealer to the Vehicle_Order__c (via lookup)
- Status remains Pending until confirmed

Trigger/Batch Apex:

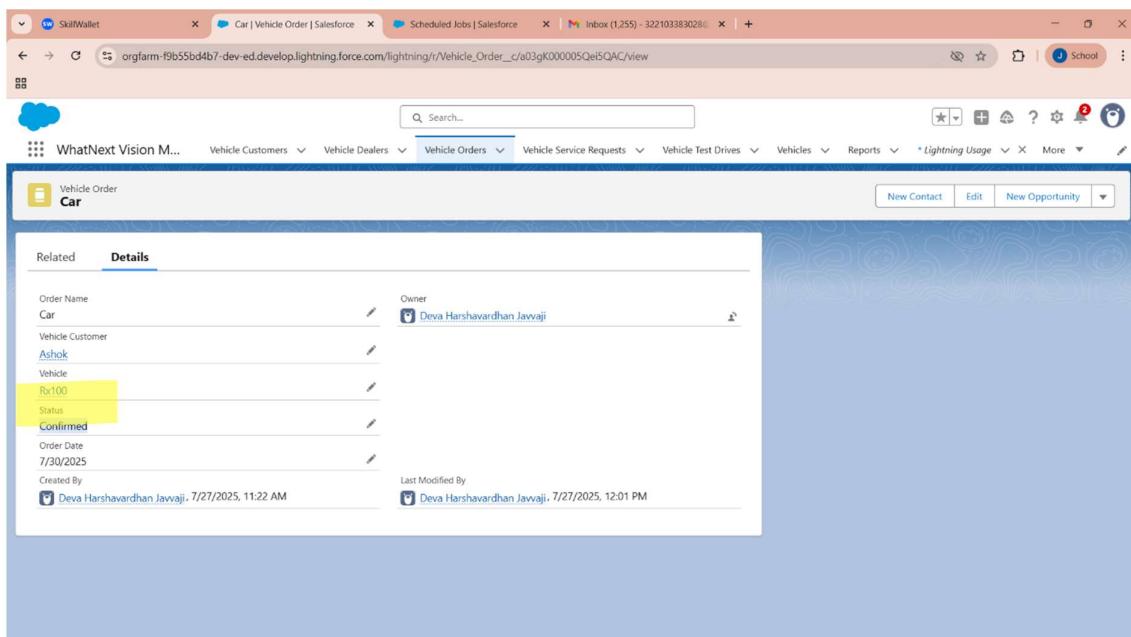
If a vehicle is out of stock, order cannot be placed (error shown via addError() in trigger)

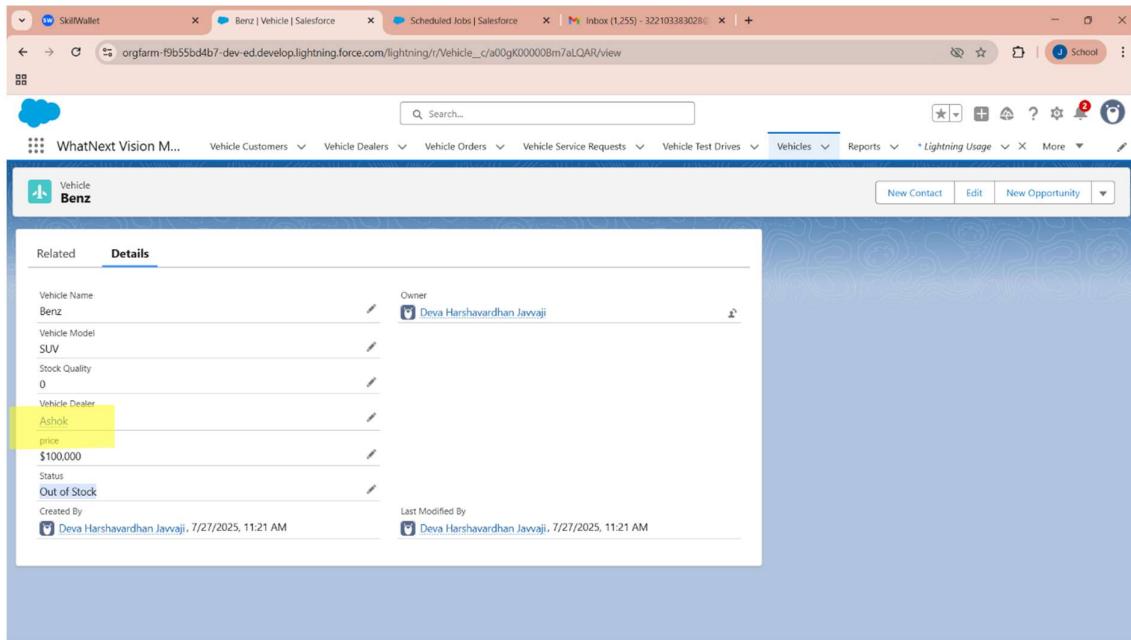
- When the Status__c of order changes to Confirmed, the Trigger or Batch Apex:
 - Reduces the vehicle stock by 1
 - Prevents over-ordering
 - Keeps vehicle stock synced

Before the Flow executed the Status is Pending:

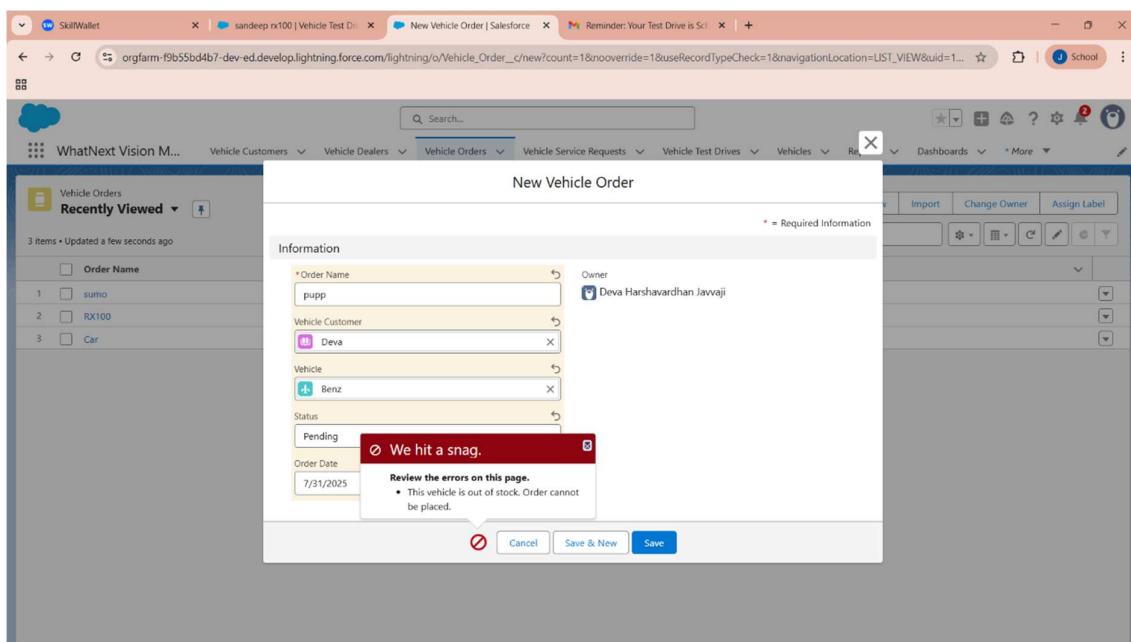


After the Execution of Flow:





Here Audi car is Out of stock.lets see the order is creating or not:



When a test drive is scheduled then email will be send Automatically to customer like this:

The screenshot shows a Gmail inbox with 1,255 messages. A yellow warning box highlights a message from 'Deva Harshavardhan Javvaji' about a scheduled test drive. The message content is as follows:

Reminder: Your Test Drive is Scheduled!

We care about your privacy. This message appears to be sent from your account but Gmail couldn't verify this. Someone might be impersonating your account. If you're not sure the message is from you, use caution when clicking links, downloading attachments, or replying with personal information.

Report spam Looks safe

Dear Deva,

This is a reminder that your test drive for the Sedan is scheduled on July 25, 2025.

We look forward to seeing you!

- Vision Motors Team

Reply Forward

This is how a customer can raise a vehicle service request in the system:

The screenshot shows a Salesforce Lightning Service Request detail page for 'sandeep rx100'. The 'Details' tab is selected, displaying the following fields:

Field	Value
Service Request Name	sandeep rx100
Vehicle Customer	Ashok
Vehicle	Benz
Service Date	7/29/2025
Issue Description	Battery problem
Status	Requested
Created By	Deva Harshavardhan Javvaji, 7/27/2025, 11:24 AM
Last Modified By	Deva Harshavardhan Javvaji, 7/27/2025, 11:24 AM

Conclusion:

Through the development of the "**WhatNext Vision Motors**" Salesforce application, I gained hands-on experience in building a complete, scalable business solution using Salesforce tools and automation features. I successfully designed and created custom objects, fields, and relationships to represent real-world entities like vehicles, customers, dealers, orders, test drives, and service requests. I enhanced user experience by setting up a custom **Lightning App**, creating **custom tabs**, and configuring **navigation items**.

On the automation side, I developed **record-triggered flows** to assign the nearest dealer to a customer and to send automated test drive reminders, improving customer engagement. I implemented Apex-based automation by creating a structured set of classes including **triggers**, **handler classes**, **batch classes**, and a **scheduler** to process vehicle orders automatically on a daily basis. This taught me how to handle real-time and scheduled processes in Salesforce effectively.

Overall, this project helped me understand and apply **Salesforce development principles**, automation best practices, and scalable design. It not only strengthened my technical skills but also gave me practical insights into building enterprise-level CRM solutions.