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

Project Overview

WhatNext Vision Motors, a rising innovator in the automotive industry, sought to modernize its customer interactions and operational workflows through a tailored Salesforce CRM implementation. The project's main objective was to optimize vehicle order management, enable accurate dealer assignments, and strengthen customer engagement through automation.

Previously, manual processes caused delays, inventory discrepancies, and reduced customer satisfaction. To address these challenges, the customized CRM solution introduced key features such as real-time stock validation, automatic dealer assignment based on customer location, automated test drive reminders via email, and backend automation using Apex triggers and batch classes.

Additionally, the system leverages Salesforce Lightning Apps and Dynamic Forms to deliver a streamlined and intuitive interface for internal users. Overall, this solution enhances operational efficiency, minimizes errors, and establishes a scalable foundation for future innovations such as AI-driven vehicle recommendations and chatbot-assisted customer support.

Objectives

The key objectives of this Salesforce CRM implementation are

1. Automate Order and Dealer Assignment

- Automatically assign the nearest dealer based on the customer's city at the time of order placement.

2. Prevent Out-of-Stock Orders

- Ensure customers can only place orders for vehicles currently in stock using validation rules and Apex triggers.

3. Send Test Drive Reminders

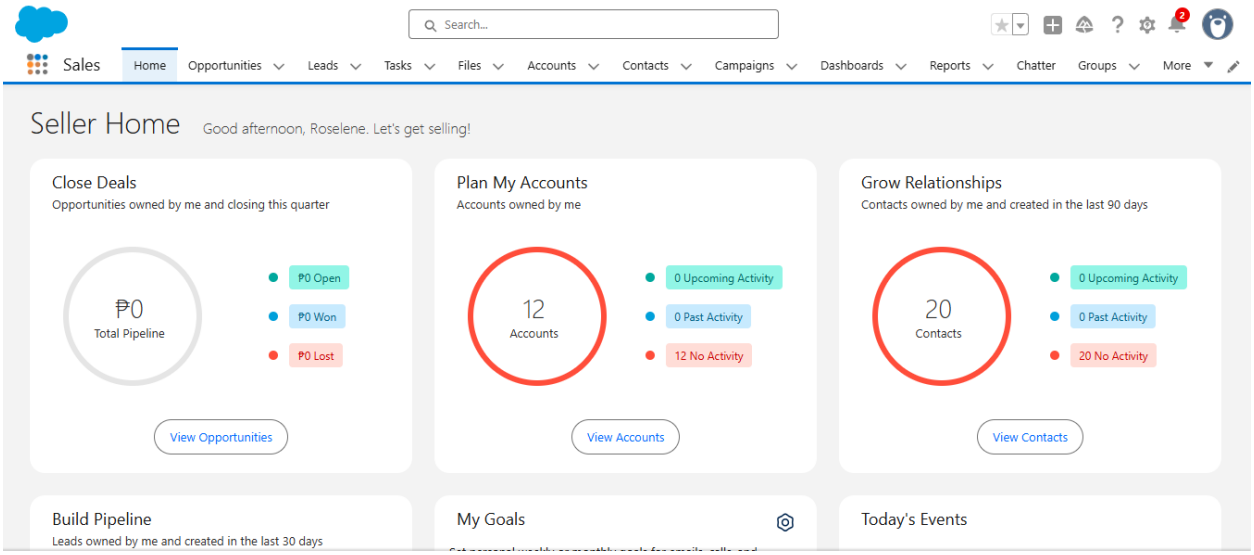
- Use scheduled email flows to remind customers of their upcoming test drives, reducing missed appointments.

4. Improve User Experience

- Implement Lightning Apps and Dynamic Forms to provide a clean, responsive interface for managing records

5. Maintain a Scalable Backend

- Use modular Apex classes and scheduled batch jobs to automate stock updates and order confirmations in bulk.



Phase 1: Requirement Analysis & Planning

The initial phase of the project focused on gaining a deep understanding of WhatNext Vision Motors' business goals and converting them into well-defined system requirements tailored for the Salesforce platform. The goal was to design a Customer Relationship Management (CRM) solution capable of supporting the entire vehicle management lifecycle — from inventory tracking and customer orders to post-sales services and customer engagement.

Business Requirements

The following core requirements were identified:

- Centralized management of vehicle, dealer, and customer information.
- Real-time validation of vehicle stock at the time of order placement.
- Automated assignment of the nearest dealer based on the customer's address.
- Tracking of test drives and vehicle service requests.
- Automation of key workflows to reduce manual intervention.

Defining Project Scope

To fulfill these business needs, the system was designed to include:

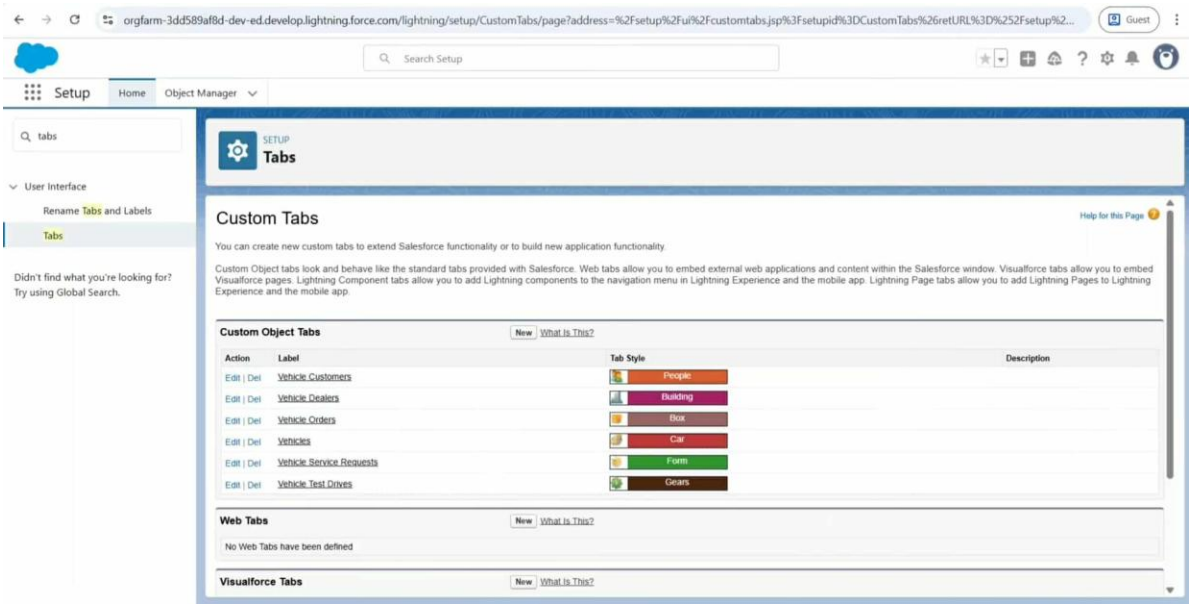
- Custom objects to manage vehicles, orders, dealers, customers, test drives, and service requests.

- Record-triggered flows for dealer assignment and automated email notifications.
- Apex triggers for stock validation and automatic updates to inventory levels.
- Batch Apex processes to handle pending orders when stock becomes available.

Data Model

Six custom objects were created to reflect the business structure

Order Name	Purpose
Vehicle	Stores vehicle details and stock info.
Vehicle Dealer	Contains dealer information.
Vehicle Customer	Stores customer details.
Vehicle Order	Tracks order details, status updates.
Vehicle Test Drive	Schedules and monitors customer test drive requests.
Vehicle Service Request	Manages service issue and history.



Security Model

- Standard Salesforce profiles were used with additional Permission Sets to grant access to custom objects.

- Field-Level Security and a defined Role Hierarchy ensured users could only view or modify data appropriate to their roles and responsibilities.
- Field History Tracking was enabled for key fields—such as *Stock Quantity* on Vehicle and *Status* on Order—to support auditing and maintain data integrity.

Phase 2: Salesforce Development - Backend & Configurations

Setup Environment & DevOps Workflow

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

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

Customization of Objects, Fields, Validation Rules and Automation

Custom Objects and Fields

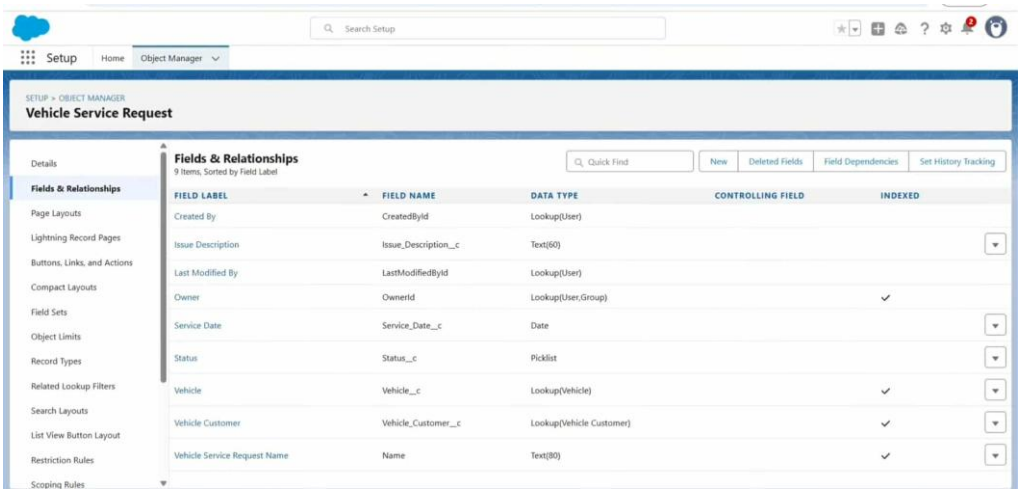
The following custom objects were created and configured to support the business flow:

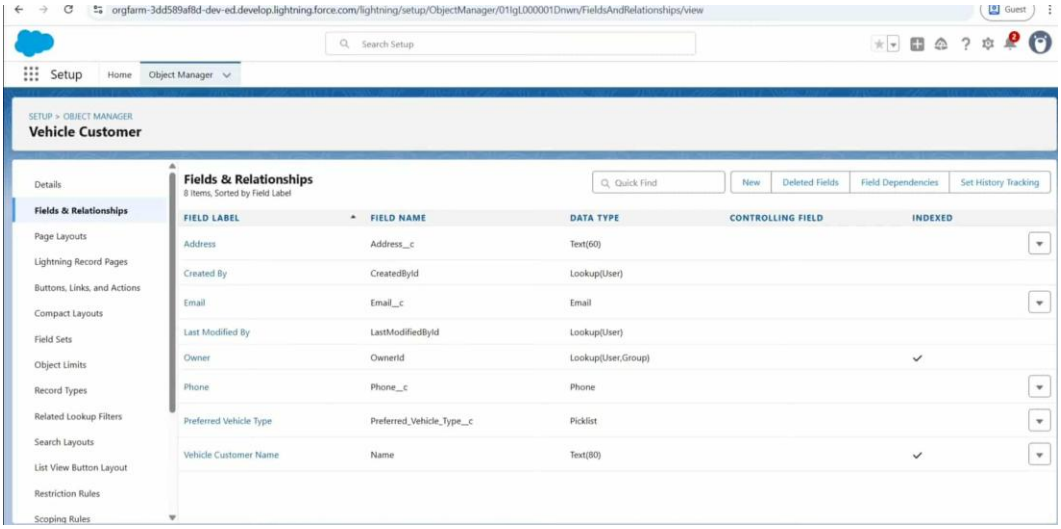
- Vehicle-Stores vehicle name, stock count, model, etc.
- Dealer - Stores dealer location and vehicle availability
- Customer-Stores customer details and address
- Order - Captures vehicle orders and order status

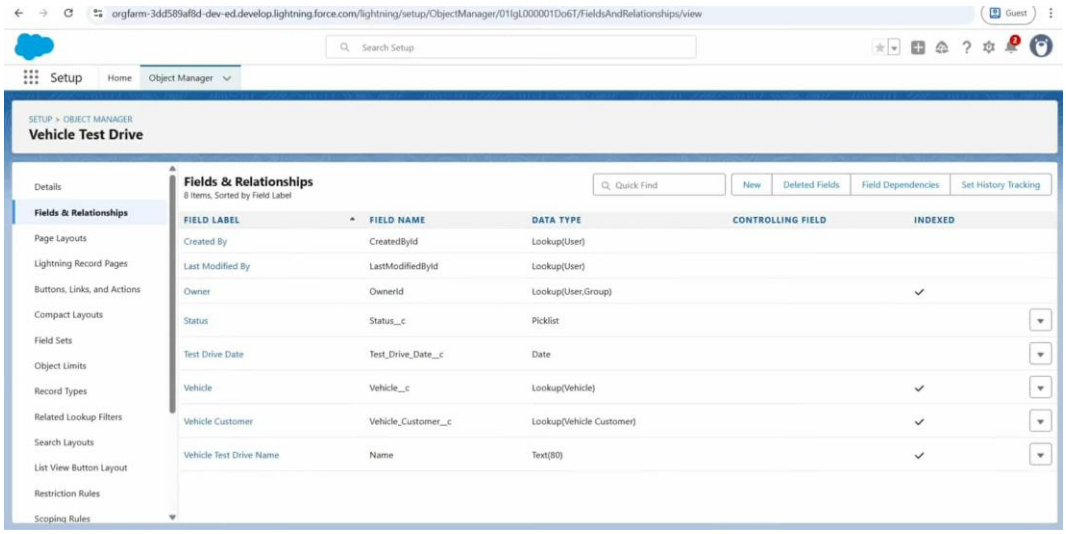
Relationships:

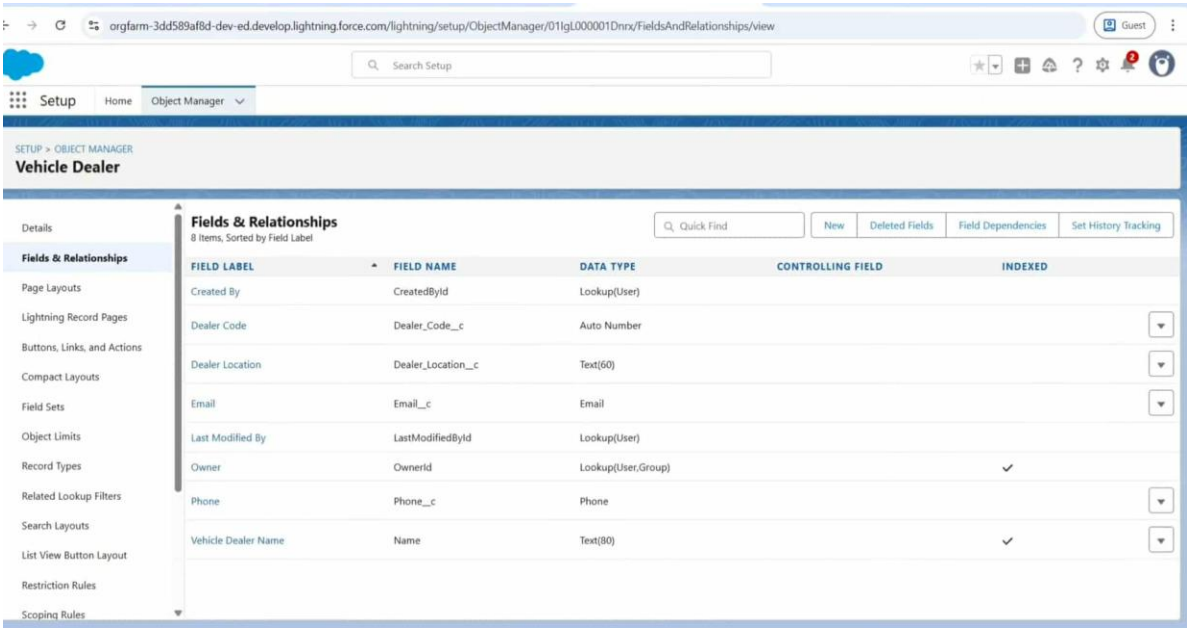
- Order → Vehicle: Lookup
- Order → Dealer: Lookup

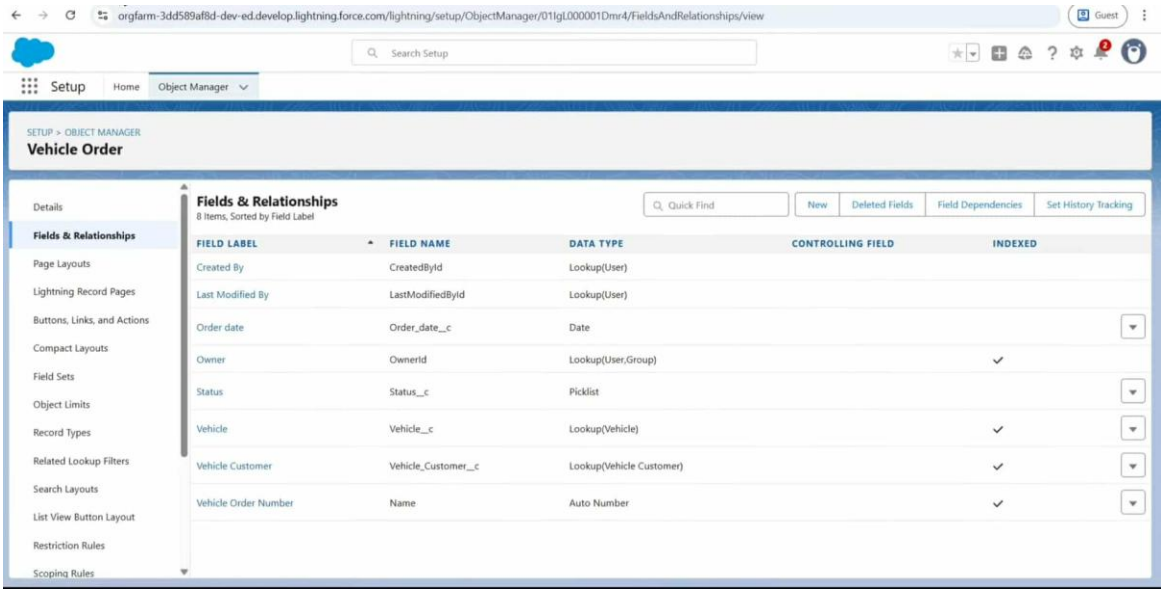
- Order → Customer: Master-Detail or Lookup (based on implementation)

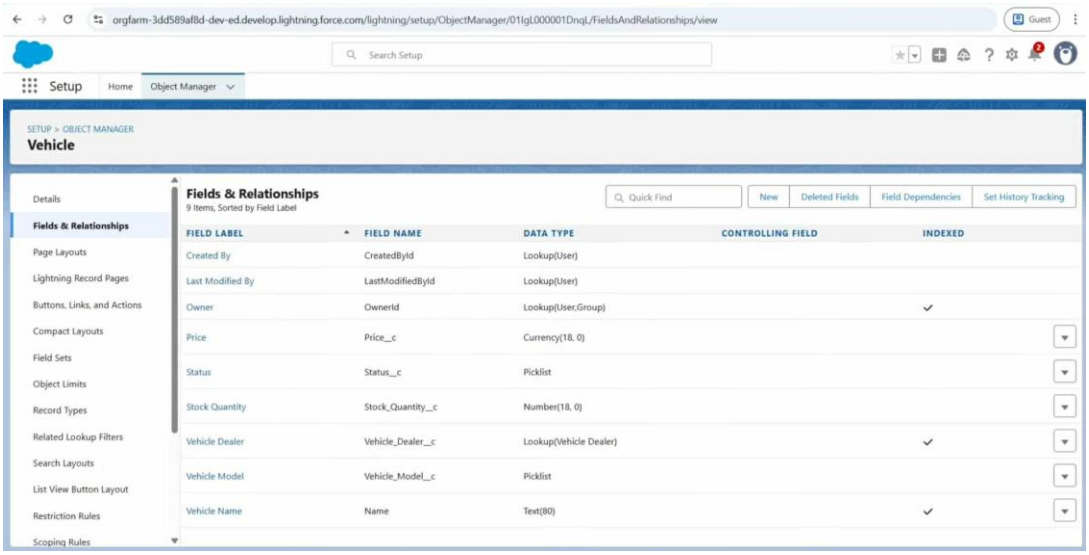







orgfarm-3dd589af8d-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01IgL000001Dnrx/FieldsAndRelationships/view
Guest
Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Vehicle Dealer
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules
Fields & Relationships
8 Items, Sorted by Field Label
Quick Find
New
Deleted Fields
Field Dependencies
Set History Tracking
FIELD LABEL
FIELD NAME
DATA TYPE
CONTROLLING FIELD
INDEXED
Created By
CreatedById
Lookup(User)
Dealer Code
Dealer_Code__c
Auto Number
Dealer Location
Dealer_Location__c
Text(60)
Email
Email__c
Email
Last Modified By
LastModifiedById
Lookup(User)
Owner
OwnerId
Lookup(User,Group)
Phone
Phone__c
Phone
Vehicle Dealer Name
Name
Text(80)


orgfarm-3dd589af8d-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01IgL000001Dmr4/FieldsAndRelationships/view
Guest
Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Vehicle Order
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules
Fields & Relationships
8 Items, Sorted by Field Label
Quick Find
New
Deleted Fields
Field Dependencies
Set History Tracking
FIELD LABEL
FIELD NAME
DATA TYPE
CONTROLLING FIELD
INDEXED
Created By
CreatedById
Lookup(User)
Last Modified By
LastModifiedById
Lookup(User)
Order date
Order_date__c
Date
Owner
OwnerId
Lookup(User,Group)
Status
Status__c
Picklist
Vehicle
Vehicle__c
Lookup(Vehicle)
Vehicle Customer
Vehicle_Customer__c
Lookup(Vehicle Customer)
Vehicle Order Number
Name
Auto Number



Validation Rules

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

  Prevents the creation of an order if the selected vehicle has zero stock.

Automation: Workflow Tools

Flows (Record-Triggered):

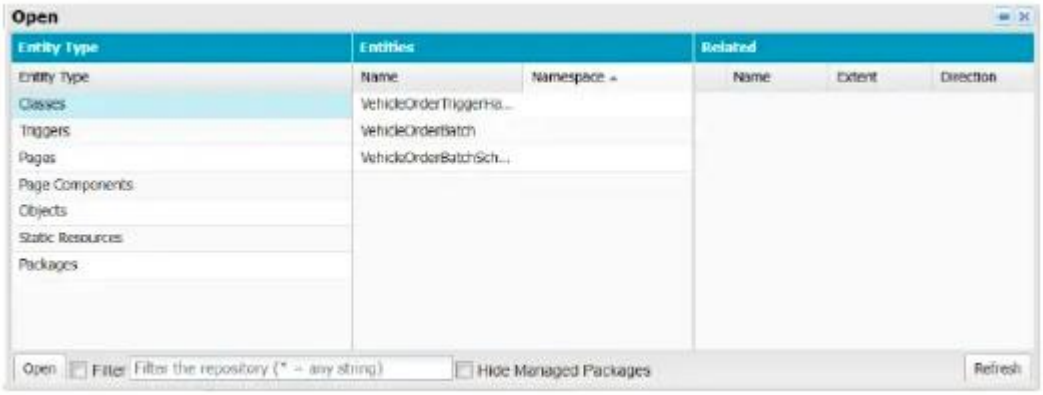
- Auto-assign the nearest dealer based on the customer's address using a Record-Triggered Flow on Order object.
- Send test drive reminders via Scheduled Flows

Apex Classes and Triggers

Apex Classes

- Apex Classes were written to modularize the trigger logic and support backend automation:
- VehideOrderTriggerHandler handles stock checks and updates in the trigger.
- VehicleOrderBatch processes pending orders and confirms them when sufficient stock becomes available.
- VehicleOrderBatchScheduler schedules the batch process to run daily at 12 PM.

All classes follow Salesforce best practices, including bulk-safe operations and reusable methods to ensure scalability and maintainability.

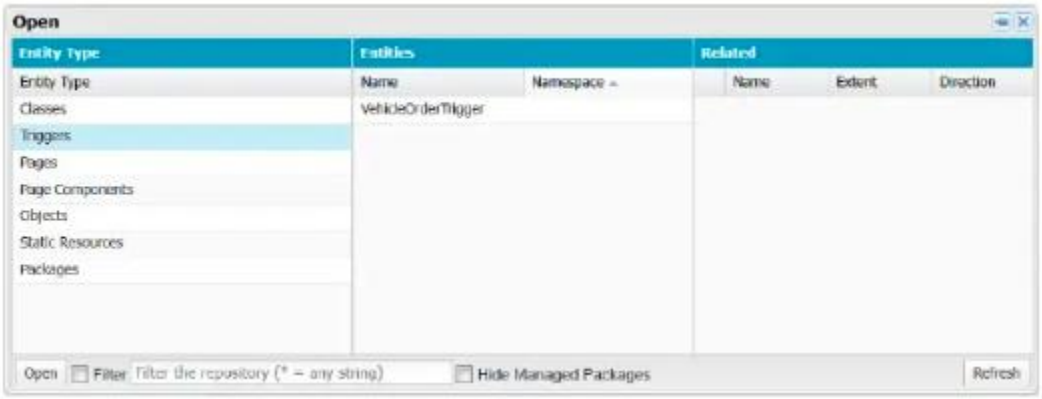


Apex Trigger:

Apex Trigger was written on the **Order** object to perform:

- Stock availability validation
- Auto-dealer assignment. (if not handled by Flow)
- Order status update logic (Pending or Confirmed)

Trigger follows best practices using a Trigger Handler pattern.



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 tahs like Vehicles, Dealers, Orders, Customers, Test Drives, and Service Requests for easy navigation.

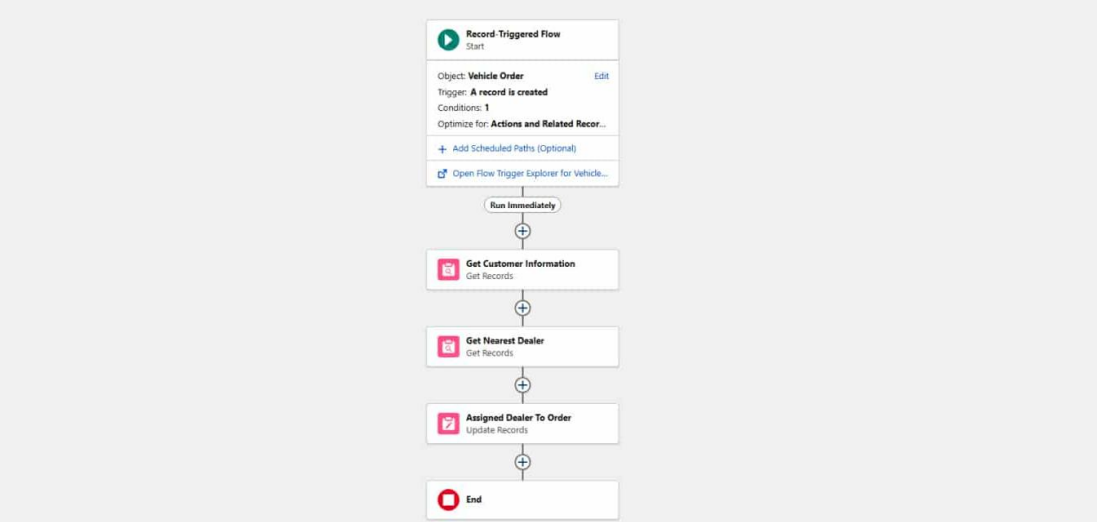
- Lightning App created: What Next Vision Motors
- Tabs: Vehicles, Dealers, Customers, Orders, Test Drives, Services
- Used Dynamic Forms for fields based on status & availability

- Highlight panels, related lists added to Lightning Pages

Page Layouts and Dynamic Forms:

Page layouts were customized for key objects such as Vehicle, Vehicle Order, and Vehicle Test Drive to create a clean and user-friendly interface that shows only the most relevant information to each user. Dynamic Forms were also implemented to place fields directly on the Lightning Record Page, allowing conditional visibility based on criteria like order status or vehicle availability. This approach improves clarity, reduces clutter, and ensures that users are guided through accurate and context-specific data entry.

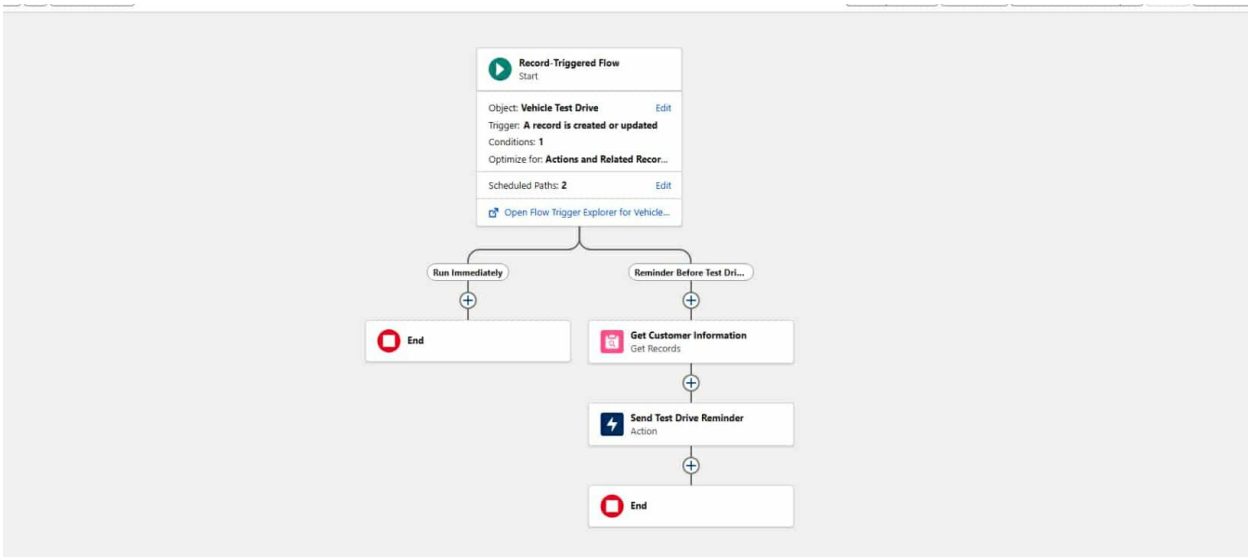
Flow 1: Auto Dealer Assignment



-

Flow 2: Test Drive Reminder

This Record-Triggered Flow:

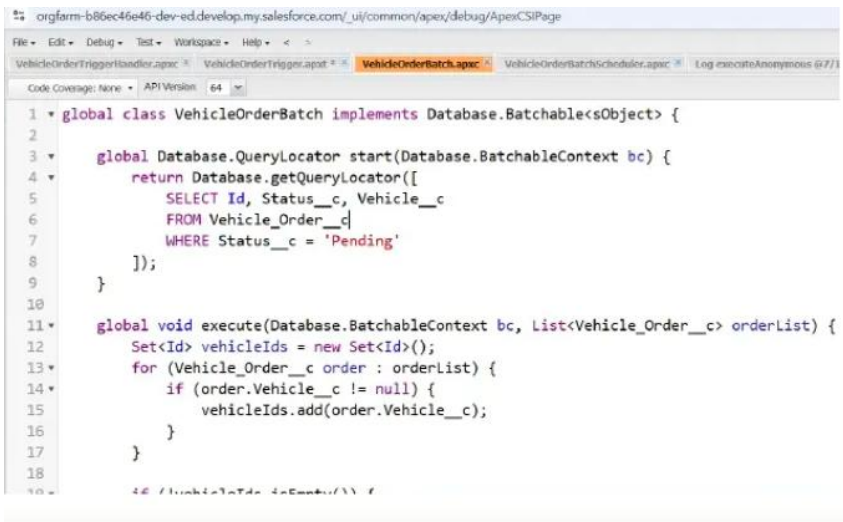


Apex Trigger & Handler

- Trigger: VehicleOrder Trigger
- Handler: VehicleOrder TriggerHandler
  - Prevents out-of-stock orders
  - Updates stock when order is confirmed

Apex Batch Class

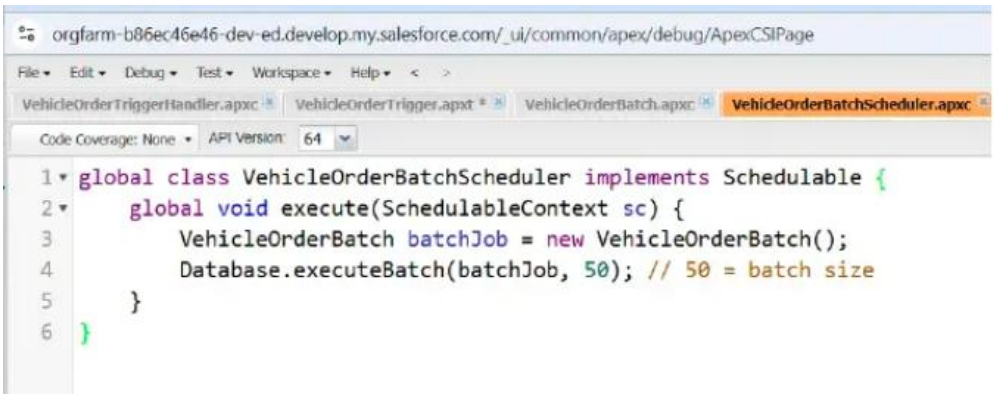
- Class: VehicleOrderBatch
- Runs daily
- Checks for pending orders and available stock
- Updates status to Confirmed and adjusts stock



Scheduled Apex

Class: VehicleOrderBatchScheduler

Executes batch class automatically



Phase 4: Data Migration, Testing & Security

Data Loading Process

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

Tools Used:

- Data Import Wizard:

  Used for importing standard object data (like Accounts, Contacts).

- Data Loader:

  Used for large volumes and for custom objects like Vehicle , Dealer , Order .

Steps:

1. Exported CSV files with sample records.

2. Mapped columns to corresponding Salesforce fields.

3. Used Data Loader to insert records for:

  - Vehicle
  - Dealer

- Customer
- Order (with valid relationships)

Field History Tracking, Duplicate Rules, and Matching Rules

Field History Tracking:

Enabled for the following objects to track changes:

- Vehicle c: Stock c field
- Order_c: Status cand Dealer c fields

Duplicate & Matching Rules:

- Matching Rule: A custom rule was defined on the Customer object to match records based on Email and Phone fields.
- Prevents the creation of duplicate customer records by enforcing the matching rule during data entry.

Profiles, Roles, Permission Sets, and Sharing Rules

Profiles and Roles

- Standard profiles like Standard User and System Administrator were used.
- Role Hierarchy established:

CEO

-Sales Manager

- Sales Rep

Permission Sets:

Created Order Management Access permission set

Assigned to users who need create/read access to Orders and Vehicles

Sharing Rules

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

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

1. Create a Vehicle:

INPUT:

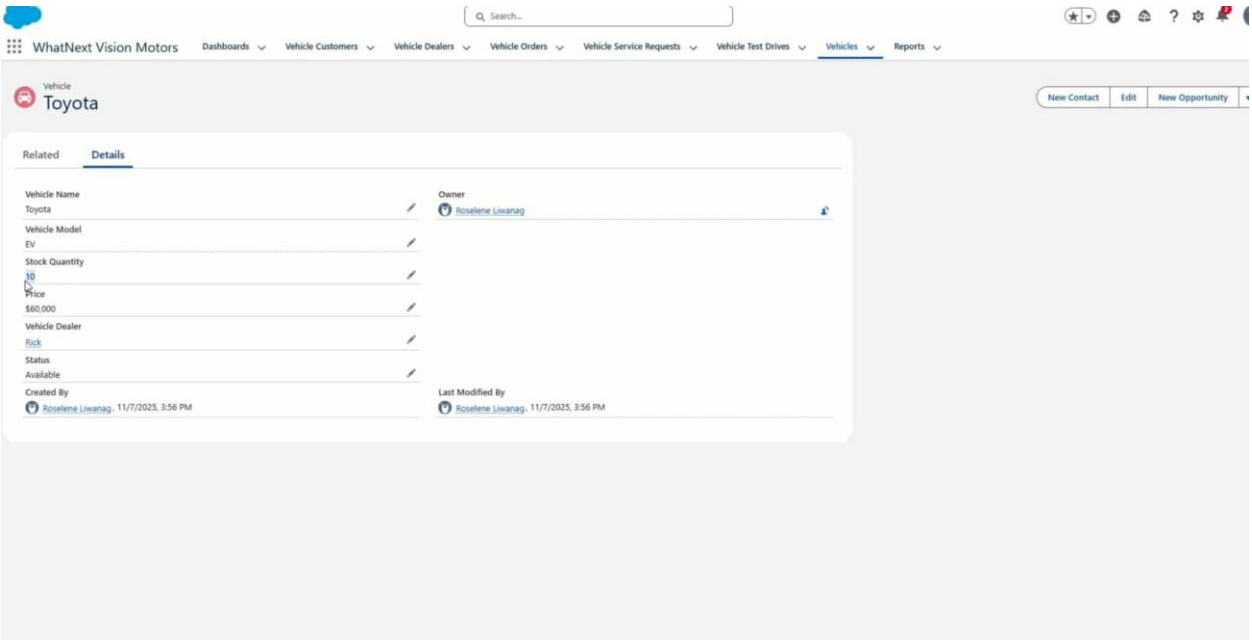
Vehicle Name: Toyota

Vehicle Model: EV

Stock Quantity: 10

Status: Available

Dealer: Select existing Vehicle Dealer



2. Test Stock=0 (Error Case):

INPUT:

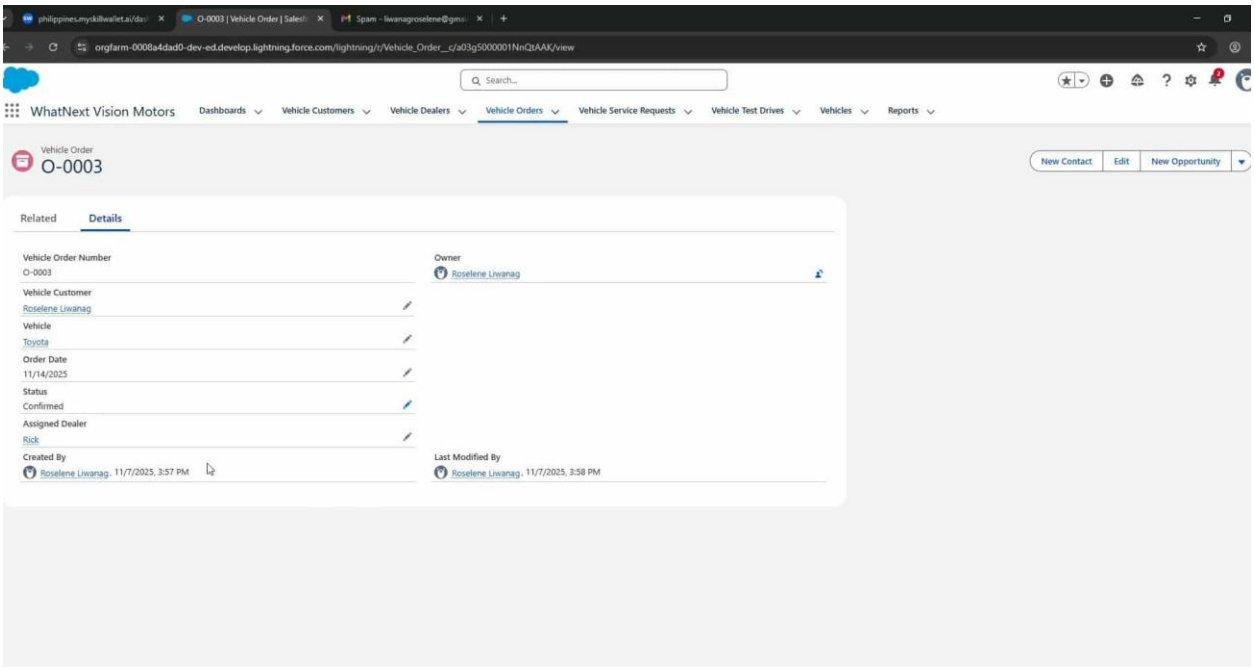
Edit the Stock Quantity of the above vehicle → Set it to 0.

Go to Vehicle Orders tab→ Click New.

Vehicle: Toyota

Status: Confirmed

Customer: Select any existing customer OUTPUT



3. Test Stock>0 (Confirmed Order)

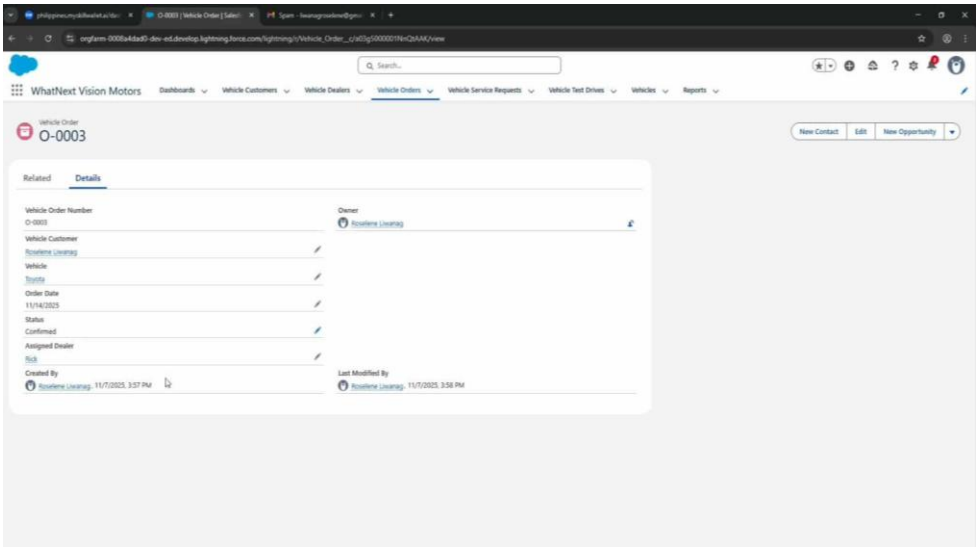
INPUT:

Steps:

1. Set vehicle Stock Quantity back to 10.

2. Create a Vehicle Order:

   - Status: Confirmed

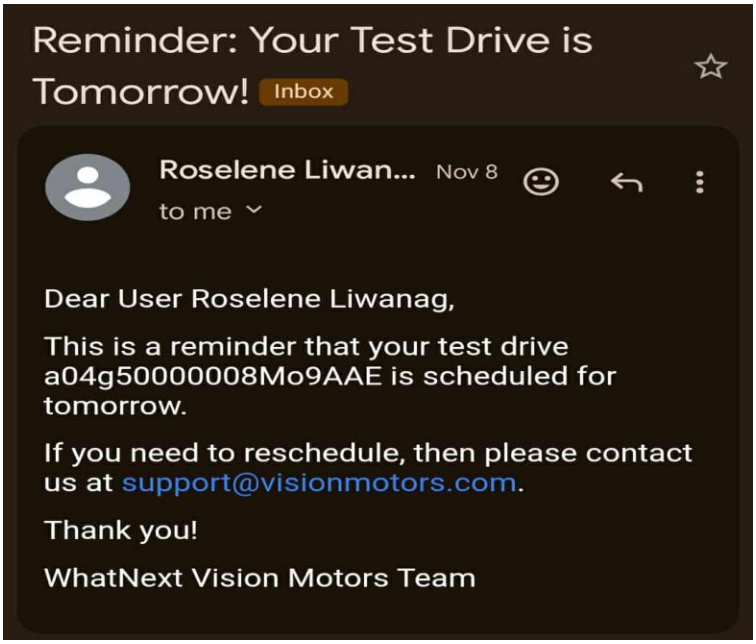


- Vehicle: Toyota

4. Test Drive Reminder Email:

Customer: Select any customer with email

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



Test Batch Job for Pending Orders:

INPUT:

Create a Pending Order when stock is 0:

1. Set Test Car stock to 0.

2. Create a Vehicle Order:

- Status: Pending

Update stock:

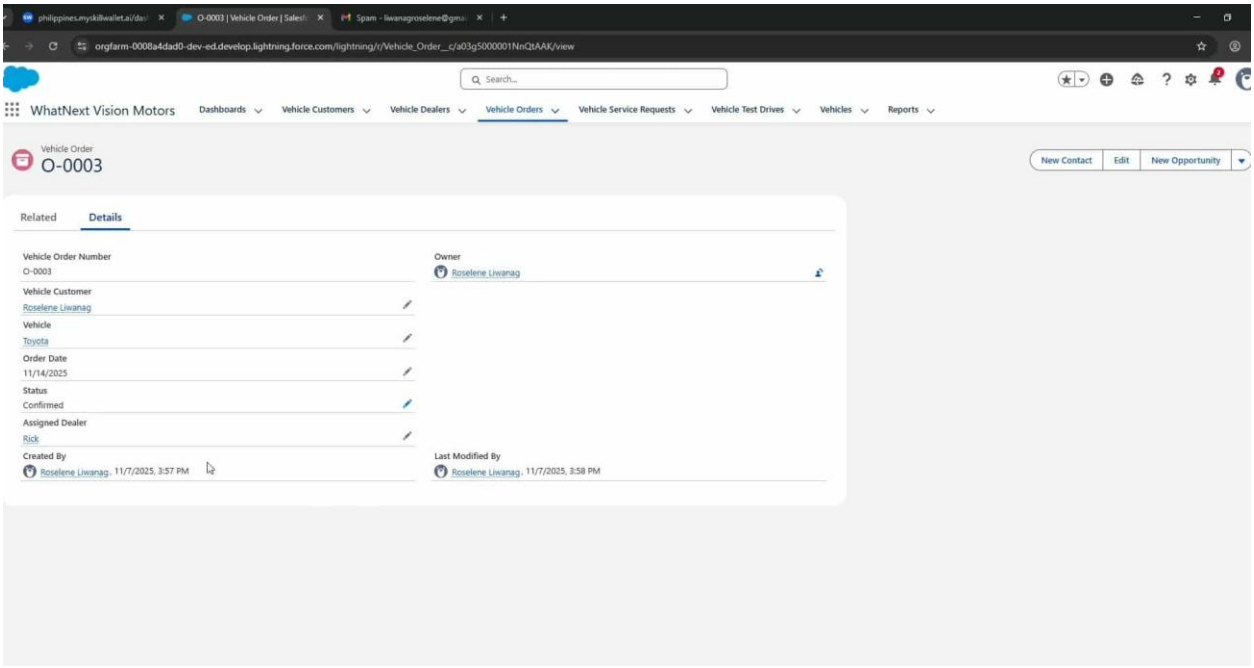
- Set Stock Quantity = 10

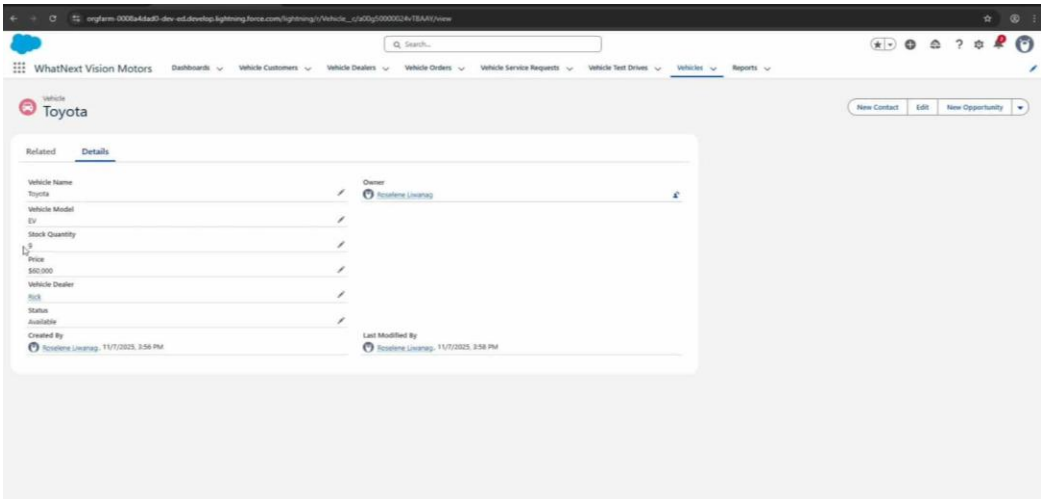
OUTPUT:

Expected Result:

Your Pending Order should become Confirmed.

Vehicle stock should reduce by 1





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

- Order TriggerHandler
- Dealer AssignmentService
- StockValidationTrigger

Test Class Features:

- Minimum 75% coverage

- Positive and negative test cases
- Used @isTest annotation with test data setup

Phase 5: Deployment, Documentation & Maintenance

Deployment Strategy

To deploy the developed features from the Developer Org to the live/production environment, the Change Set deployment method was used.

Deployment Steps:

1. Created an Outbound Change Set in the source org.

2. Added all custom components:

    - Custom objects, fields, flows, validation rules, triggers, and Apex classes.

3. Uploaded the Change Set to the Target Org (production/sandbox).

4. Validated and deployed it from Inbound Change Sets in the target org.

5. Post-deployment manual verification was done to ensure everything works as expected.

Testing & Sample Scenarios

- A test case was executed to ensure that creating a vehicle order with zero stock correctly triggered a stock error.
- Stock was updated to 2, and placing an order was tested to confirm that inventory decreased to 1.
- A pending order scenario was created and validated to confirm that the Batch Apex job automatically processed and confirmed the order once stock became available.

System Maintenance and Monitoring

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

1. Monitoring

- Apex Jobs were used to monitor scheduled jobs and batch class executions.
- Debug Logs were utilized to trace errors and diagnose unexpected behavior.
- Email Alerts were enabled to notify users about test drive reminders or failed processes.

2. User Feedback Loop

- Sales and operations teams were instructed to use the system for several days after deployment.
- Feedback was collected through manual walkthroughs to identify missing features, issues, or usability concerns.

3. Updates and Fixes

- Minor adjustments—such as updating field labels or adding help text—were made in the sandbox and redeployed using Change Sets.
- Quarterly system reviews were scheduled to plan enhancements, UI improvements, and future updates.

Troubleshooting Approach

In case any issues occur in the production environment, the following procedure will be followed:

- **Reproduce the Issue:** Attempt to replicate the problem in a sandbox or developer environment to understand its cause.
- **Enable Debug Logs:** Activate debug logs for the affected user and analyze the execution of Flows or Apex to identify errors.
- **Review Apex Jobs and Flows:** For issues related to background processes, check failed Apex Jobs or Flow error notifications.
- **Fix and Retest:** Adjust the logic in Flows or Apex as needed, retest the solution in the sandbox, and deploy the fix to production using Change Sets.

Conclusion

The Salesforce implementation at WhatsNext Vision Motors successfully achieved its objective of streamlining the customer ordering process and improving operational workflows.

Key configurations include:

- Automated nearest dealer assignment using Flows or Triggers
- Stock validation rules to prevent out-of-stock orders
- Scheduled logic to update order statuses (if Batch Apex was implemented)
- Enhanced customer experience through automation
- Reduced manual intervention for internal teams

Future Enhancements: Potential improvements include AI-driven suggestions, chatbot integration, and additional automation for scalable growth.

Overall, this project lays a strong foundation for **future Salesforce innovations** while supporting WhatsNext Vision Motors' commitment to operational excellence and customer satisfaction.