

# **WhatNext Vision Motors CRM**

## **Project Documentation**

Submitted by:  
Ken Chico

## Table of Contents

<b>WhatNext Vision Motors CRM Project Documentation.....</b>	<b>1</b>
Project Overview.....	3
Objectives.....	3
<b>Phase 1: Requirement Analysis &amp; Planning.....</b>	<b>3</b>
Understanding Business Requirements.....	3
Defining Project Scope and Objectives.....	3
Design Data Model and Security Model.....	3
Stakeholders Mapping.....	7
Execution Roadmap.....	7
<b>Phase 2: Salesforce Development - Backend &amp; Configurations.....</b>	<b>7</b>
Setup Environment & DevOps Workflow.....	7
Customization of Objects, Fields, Validation Rules, Automation.....	8
Apex Classes, Triggers, Asynchronous Apex.....	9
<b>Phase 3: UI/UX Development &amp; Customization.....</b>	<b>11</b>
Lightning App Setup.....	11
Page Layouts, Dynamic Forms.....	11
User Management.....	11
Reports and Dashboards.....	11
Lightning Pages.....	11
<b>Phase 4: Data Migration, Testing &amp; Security.....</b>	<b>12</b>
Data Loading Process.....	12
Field History Tracking, Duplicate Rules, Matching Rules.....	12
Profiles, Roles, Permission Sets, Sharing Rules.....	12
Creation of Test Classes.....	12
Test Cases with Screenshots.....	12
<b>Phase 5: Deployment, Documentation &amp; Maintenance.....</b>	<b>14</b>
Deployment Strategy.....	14
System Maintenance and Monitoring.....	15
Troubleshooting Approach.....	15
<b>Conclusion.....</b>	<b>15</b>

## Project Overview

WhatNext Vision Motors CRM automates vehicle order processing in Salesforce, assigning orders to nearest dealers based on customer location, preventing orders for out-of-stock vehicles, and sending test drive email reminders one day prior to scheduled dates. Key features include custom objects for vehicles, customers, dealers, orders, test drives, and service requests with automation via Flows and Apex triggers.

## Objectives

The primary goals are to streamline vehicle order assignment to geographically closest dealers, enforce stock availability validation to prevent overselling, and automate test drive reminders for improved customer experience. These objectives deliver business value through efficient order processing, reduced manual intervention, inventory accuracy, and enhanced customer satisfaction via timely communications.

## Phase 1: Requirement Analysis & Planning

### Understanding Business Requirements

- Automate vehicle order assignment to nearest dealer based on customer address matching dealer location.
- Prevent order creation for vehicles with zero stock quantity.
- Send automated email reminders to customers one day before scheduled test drives.
- Track vehicle orders, test drives, and service requests with status management.

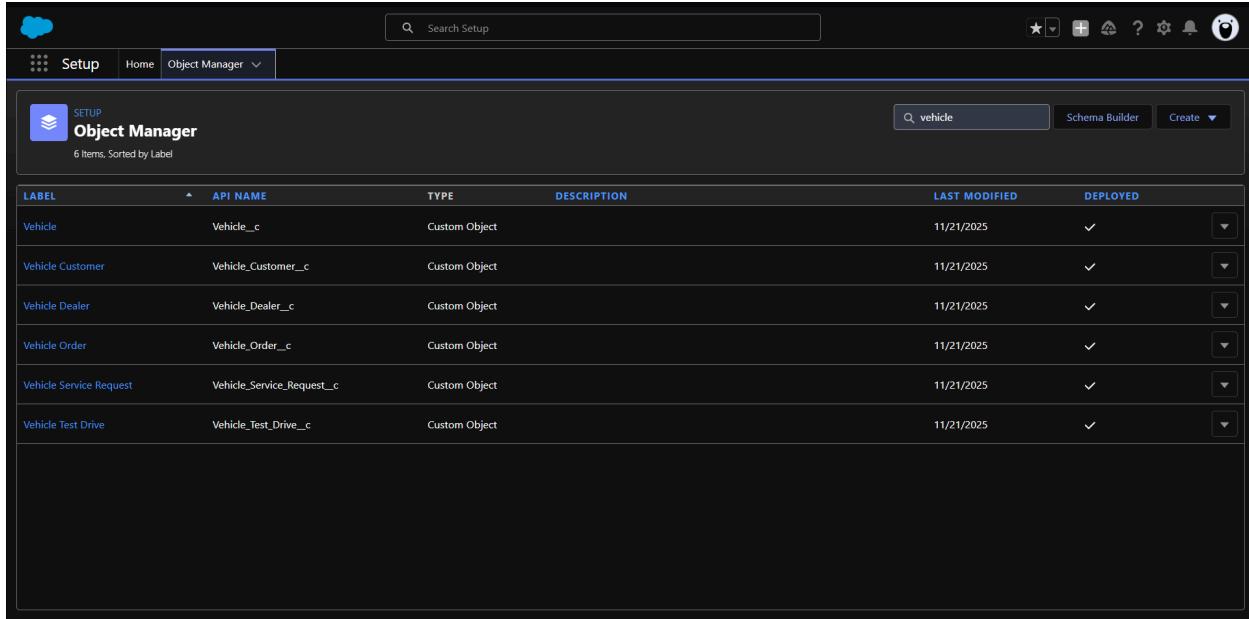
### Defining Project Scope and Objectives

- In Scope: Custom objects (Vehicle, Vehicle Customer, Vehicle Dealer, Vehicle Order, Vehicle Test Drive, Vehicle Service Request), Flows for auto-assignment and email alerts, Apex triggers for stock management.
- Out of Scope: Payment processing, full inventory forecasting, mobile app integration.
- Objectives tied to operational efficiency and customer service improvement.

### Design Data Model and Security Model

Figure 1 shows the full list of all the custom objects that were created for the development of WhatNext Vision Motors CRM.

Figure 1  
Object Manager Overview



The screenshot shows the Salesforce Object Manager interface. At the top, there's a search bar with 'Search Setup' and a 'Setup' button. Below the header, the title 'Object Manager' is displayed with a blue icon. A sub-header indicates '6 Items, Sorted by Label'. The main area is a table with columns: 'LABEL', 'API NAME', 'TYPE', 'DESCRIPTION', 'LAST MODIFIED', and 'DEPLOYED'. The data rows are:

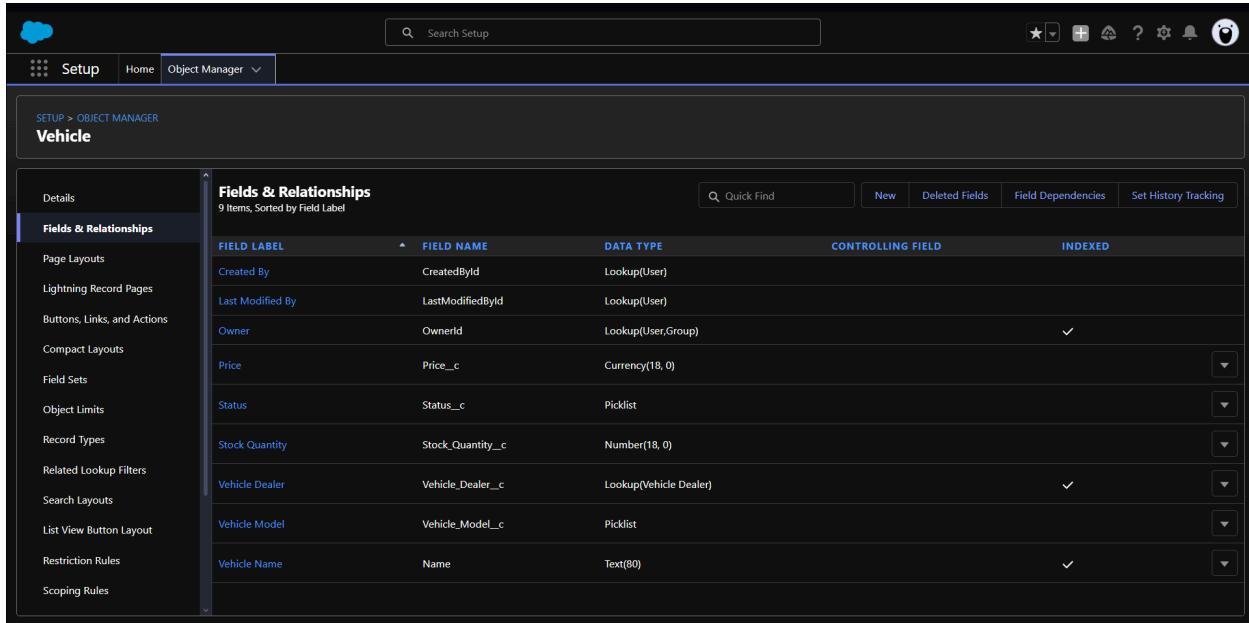
Label	API Name	Type	Description	Last Modified	Deployed
Vehicle	Vehicle_c	Custom Object		11/21/2025	✓
Vehicle Customer	Vehicle_Customer_c	Custom Object		11/21/2025	✓
Vehicle Dealer	Vehicle_Dealer_c	Custom Object		11/21/2025	✓
Vehicle Order	Vehicle_Order_c	Custom Object		11/21/2025	✓
Vehicle Service Request	Vehicle_Service_Request_c	Custom Object		11/21/2025	✓
Vehicle Test Drive	Vehicle_Test_Drive_c	Custom Object		11/21/2025	✓

### Data Model:

Figure 2 shows the screenshot of the fields and relationships of the vehicle object in the object manager.

Figure 2

### Vehicle Object



The screenshot shows the 'Fields & Relationships' section for the 'Vehicle' object in the Object Manager. On the left, a sidebar lists various setup categories like Page Layouts, Lightning Record Pages, etc. The main area has a title 'Fields & Relationships' with a subtitle '9 items, Sorted by Field Label'. It includes a 'Quick Find' search bar and buttons for 'New', 'Deleted Fields', 'Field Dependencies', and 'Set History Tracking'. The table has columns: 'FIELD LABEL', 'FIELD NAME', 'DATA TYPE', 'CONTROLLING FIELD', and 'INDEXED'. The data rows are:

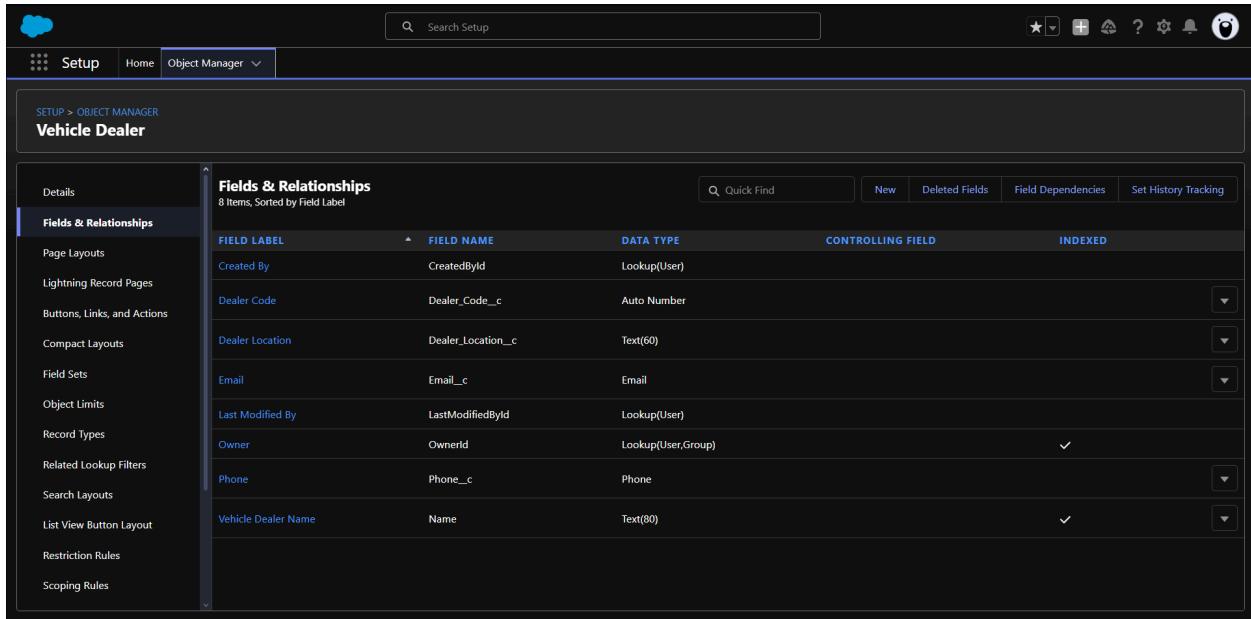
Field Label	Field Name	Data Type	Controlling Field	Indexed
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Price	Price_c	Currency(18, 0)		
Status	Status_c	Picklist		
Stock Quantity	Stock_Quantity_c	Number(18, 0)		
Vehicle Dealer	Vehicle_Dealer_c	Lookup(Vehicle Dealer)		✓
Vehicle Model	Vehicle_Model_c	Picklist		
Vehicle Name	Name	Text(80)		✓

Vehicle: Name, Model (Picklist: Sedan, SUV, EV, etc.), Stock Quantity (Number), Price (Currency), Dealer (Lookup), Status (Picklist: Available, Out of Stock, Discontinued).

Figure 3 shows the screenshot of the fields and relationships of the vehicle dealer object in the object manager.

Figure 3

## Vehicle Dealer



The screenshot shows the Salesforce Object Manager interface for the 'Vehicle Dealer' object. The left sidebar lists various setup options like Details, Fields & Relationships, Page Layouts, and Lightning Record Pages. The main content area is titled 'Fields & Relationships' and displays eight fields. The fields are:

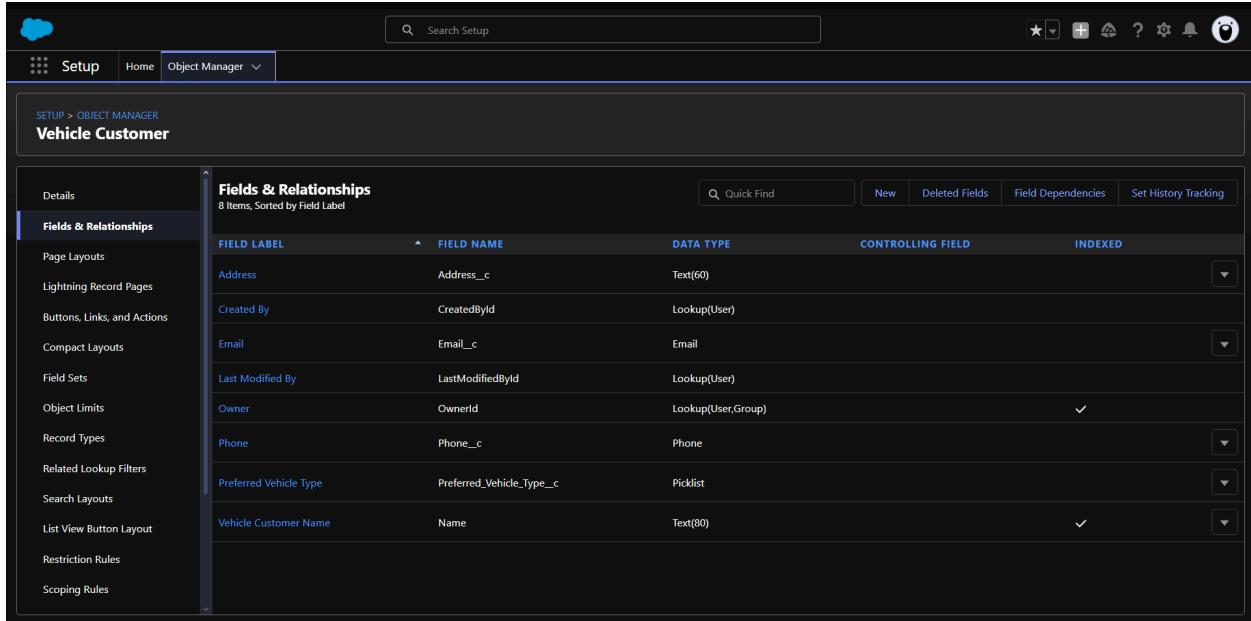
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Dealer Code	Dealer_Code__c	Auto Number		
Dealer Location	Dealer_Location__c	Text(60)		
Email	Email__c	Email		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)	✓	
Phone	Phone__c	Phone		
Vehicle Dealer Name	Name	Text(80)	✓	

Vehicle Dealer: Name, Location (Text 60), Code (Auto Number: DC-0001), Phone, Email.

Figure 4 depicts the fields and relationships of the vehicle dealer object.

Figure 4

## Vehicle Customer



The screenshot shows the Salesforce Object Manager interface for the 'Vehicle Customer' object. The left sidebar lists various setup options like Details, Fields & Relationships, Page Layouts, and Lightning Record Pages. The main content area is titled 'Fields & Relationships' and displays eight fields. The fields are:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Text(60)		
Created By	CreatedById	Lookup(User)		
Email	Email__c	Email		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)	✓	
Phone	Phone__c	Phone		
Preferred Vehicle Type	Preferred_Vehicle_Type__c	Picklist		
Vehicle Customer Name	Name	Text(80)	✓	

Vehicle Customer: Name, Email, Phone, Address (Text 60), Preferred Vehicle Type (Picklist).

Figure 5 showcases the fields and relationships of the vehicle order in the object manager.

**Figure 5**  
Vehicle Order

Fields & Relationships				
9 Items, Sorted by Field Label				
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Assigned Dealer	Assigned_Dealer_c	Lookup(Vehicle Dealer)		✓
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		✓

Vehicle Order: Order Number (Auto Number: O-0001), Customer (Lookup), Vehicle (Lookup), Order Date (Date), Status (Picklist: Pending, Confirmed, Delivered, Cancel), Assigned Dealer (Lookup).

Figure 6 presents the vehicle test drive custom object along with its fields and relationships in the object manager.

**Figure 6**  
Vehicle Test Drive

Fields & Relationships				
8 Items, Sorted by Field Label				
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
Last Modified By	LastModifiedBy	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)		✓

Vehicle Test Drive: Customer (Lookup), Vehicle (Lookup), Test Drive Date (Date), Status (Picklist: Scheduled, Completed, Cancel).

Figure 7 depicts the vehicle service request object with its fields and relationships.

Figure 7

### Vehicle Service Request

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Issue Description	Issue_Description__c	Text(60)		
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 Service Request: Customer (Lookup), Vehicle (Lookup), Service Date (Date), Issue Description (Text 60), Status (Picklist: Requested, In Progress, Completed).  
 Security Model: System Administrator profile access; standard Salesforce sharing rules apply.

### Stakeholders Mapping

- Primary: Dealers (order assignment), Customers (test drives/services), Admins (maintenance).
- Secondary: Salesforce Admins for monitoring Flows/Apex.

### Execution Roadmap

1. Developer Org Setup → Object/Field Creation → App/Tabs → Flows → Apex → Testing.

## Phase 2: Salesforce Development - Backend & Configurations

### Setup Environment & DevOps Workflow

- Salesforce Developer Edition Org created via signup link.
- WhatNext Vision Motors Lightning App created via App Manager, including all custom objects + Reports/Dashboards.

## Customization of Objects, Fields, Validation Rules, Automation

- 6 Custom Objects created with "Allow Reports" and "Allow Search" enabled.
- Fields as detailed in Data Model above.
- Flows:

Flow Name	Trigger	Logic
Auto Dealer Assign	Vehicle Order (Status = Pending)	Get customer address → Match nearest Vehicle Dealer → Update Assigned Dealer field.
Test Drive Reminder	Vehicle Test Drive (Status = Scheduled, 1 day before Test Drive Date)	Get customer email → Send reminder email with test drive details.

Figures 8 and 9 show the actual flow via flow builder.

Figure 8

Auto Assign Dealer Flow

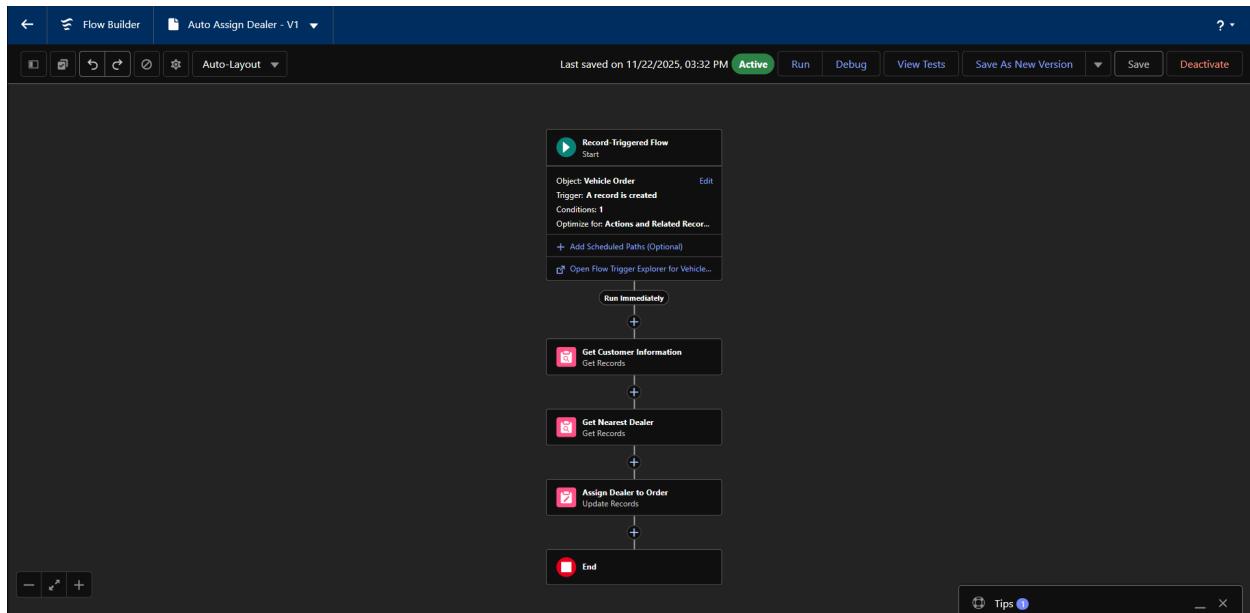
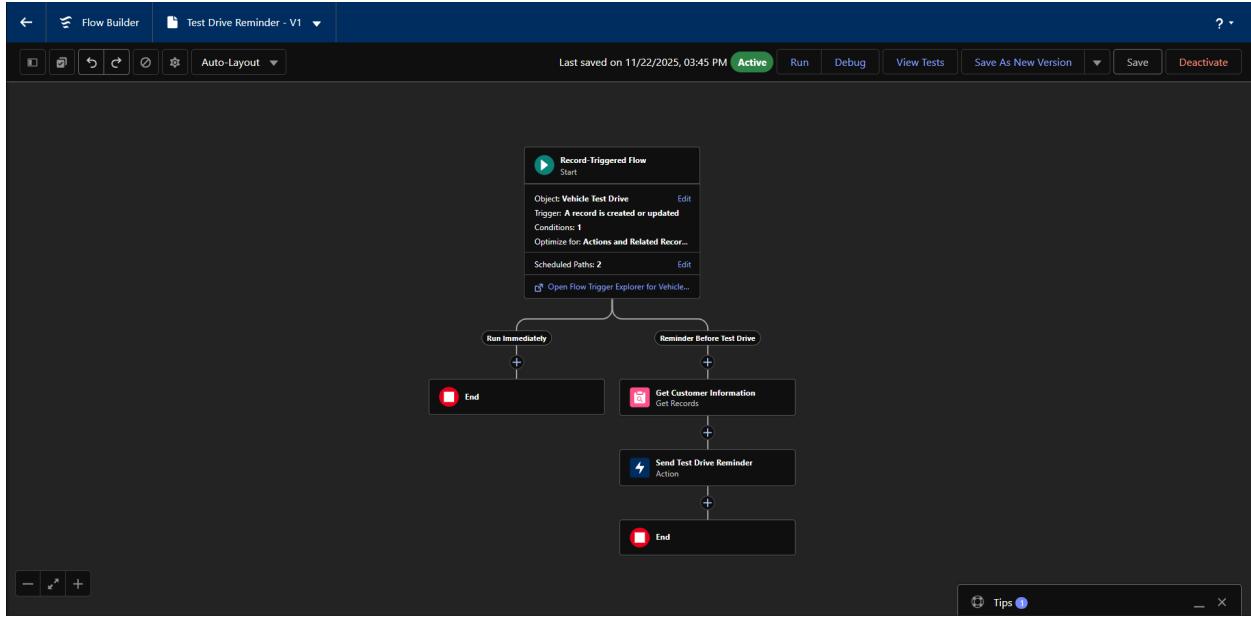


Figure 9

Test Drive Reminder Flow



## Apex Classes, Triggers, Asynchronous Apex

Figures 10 to 13 shows the different apex classes and triggers that were made to handle the automation of the WhatNext Vision Motors CRM gracefully.

Figure 10

### Vehicle Order Trigger Handler

```
public class VehicleOrderTriggerHandler {
    public static void handleTrigger(List<Vehicle_Order__c> newOrders, Map<Id, Vehicle_Order__c> oldOrders, Boolean isAfter, Boolean isInsert, Boolean isUpdate) {
        if (isAfter) {
            if (isInsert || isUpdate) {
                preventOrderIfStock(newOrders);
            }
        }

        if (isInsert) {
            if (isUpdate || isDelete) {
                updateVehicleDevelopment(newOrders);
            }
        }

        // Method to prevent orders when the vehicle is out of stock
        private static void preventOrderIfStock(List<Vehicle_Order__c> orders) {
            Set<Id> vehicleIds = new Set<Id>();
            for (Vehicle_Order__c order : orders) {
                if (order.order_Status__c == 'Confirmed') {
                    vehicleIds.add(order.vehicle__c);
                }
            }

            if (!vehicleIds.isEmpty()) {
                Map<Id, Vehicle__c> vehicleMap = new Map<Id, Vehicle__c>();
                for (Vehicle__c vehicle : [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]) {
                    vehicleMap.put(vehicle.Id, vehicle);
                }

                List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();
                for (Vehicle_Order__c order : orders) {
                    if (vehicleMap.containsKey(order.vehicle__c)) {
                        Vehicle__c vehicle = vehicleMap.get(order.vehicle__c);
                        if (vehicle.stock_Quantity__c <= 0) {
                            vehicle.stock_Quantity__c += 1;
                            vehiclesToUpdate.add(vehicle);
                        }
                    }
                }

                if (vehiclesToUpdate.size() > 0) {
                    updateVehicleStock(vehiclesToUpdate);
                }
            }
        }
    }
}
```

```
private static void updateVehicleStock(List<Vehicle__c> orders) {
    Set<Id> vehicleIds = new Set<Id>();
    for (Vehicle_Order__c order : orders) {
        if (order.order_Status__c == 'Confirmed') {
            vehicleIds.add(order.vehicle__c);
        }
    }

    if (!vehicleIds.isEmpty()) {
        Map<Id, Vehicle__c> vehicleMap = new Map<Id, Vehicle__c>();
        for (Vehicle__c vehicle : [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]) {
            vehicleMap.put(vehicle.Id, vehicle);
        }

        List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();
        for (Vehicle_Order__c order : orders) {
            if (vehicleMap.containsKey(order.vehicle__c)) {
                Vehicle__c vehicle = vehicleMap.get(order.vehicle__c);
                if (vehicle.stock_Quantity__c > 0) {
                    vehicle.stock_Quantity__c -= 1;
                    vehiclesToUpdate.add(vehicle);
                }
            }
        }

        if (vehiclesToUpdate.size() > 0) {
            updateVehicleStock(vehiclesToUpdate);
        }
    }
}
```

Vehicle Order Trigger Handler (Apex Class): Handles stock reduction on Confirmed status; prevents orders if stock = 0.

Figure 11

### Vehicle Order Trigger

```
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);

}
```

Vehicle Order Trigger: Fires on Vehicle Order insert/update.

Figure 12  
Vehicle Order Batch

```
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>(
                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
            );

            List<Vehicle_Order__c> ordersToUpdate = new List<Vehicle_Order__c>();
            List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();

            for (Vehicle_Order__c order : orderList) {
                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
                if (vehicle != null && 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.');
    }
}
```

Vehicle Order Batch (Batch Apex): Checks pending orders for out-of-stock vehicles and updates status.

Figure 13  
Vehicle Order Batch Scheduler

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

Vehicle Order Batch Scheduler (Schedulable Apex): Schedules batch execution.

## Phase 3: UI/UX Development & Customization

### Lightning App Setup

- WhatNext Vision Motors App: Contains tabs for all 6 custom objects + Reports/Dashboards; assigned to System Administrator.

### Page Layouts, Dynamic Forms

- Standard page layouts used with all custom fields.

### User Management

- System Administrator profile only.

### Reports and Dashboards

- Added to App; no custom reports developed in this phase.

### Lightning Pages

Figure 14 shows the custom tabs that were made for the WhatNext Vision Motors CRM Project to display custom object data in a user-friendly format.

Figure 14  
Custom Tabs

Action	Label	Tab Style	Description
Edit   Del	Vehicle Customers	People	
Edit   Del	Vehicle Dealers	Building	
Edit   Del	Vehicle Orders	Box	
Edit   Del	Vehicles	Car	
Edit   Del	Vehicle Service Requests	Form	
Edit   Del	Vehicle Test Drives	Gears	

Custom tabs with icons: Vehicle (Car), Customer (People), Dealer (Building), Order (Box), Service (Form), Test Drive (Gear).

## Phase 4: Data Migration, Testing & Security

### Data Loading Process

- Manual record creation via New buttons for testing (Customers, Dealers, Vehicles).

### Field History Tracking, Duplicate Rules, Matching Rules

- Not implemented in this project.

### Profiles, Roles, Permission Sets, Sharing Rules

- System Administrator profile with full access.
- Standard role hierarchy. No custom sharing rules.

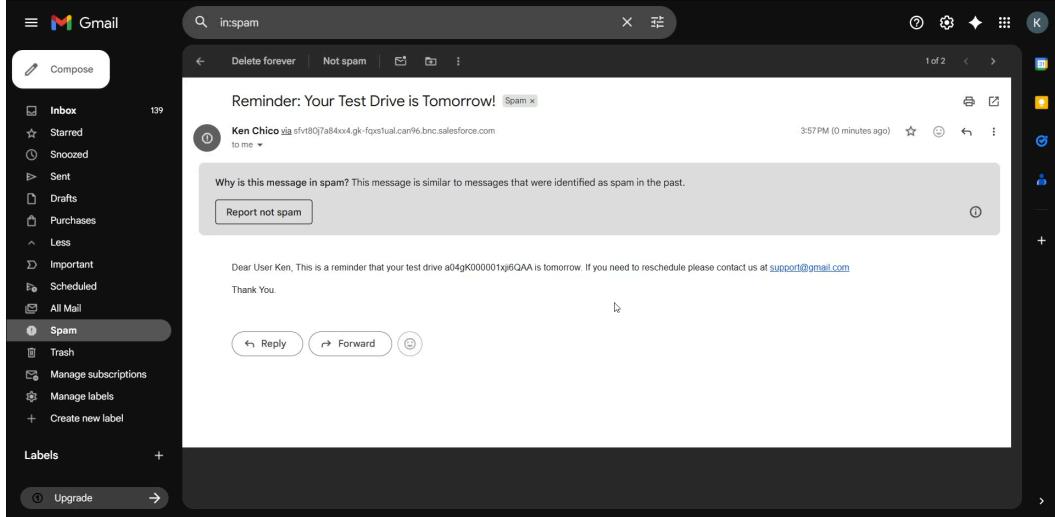
### Creation of Test Classes

- Not shown; Apex classes include necessary logic for unit tests.

### Test Cases with Screenshots

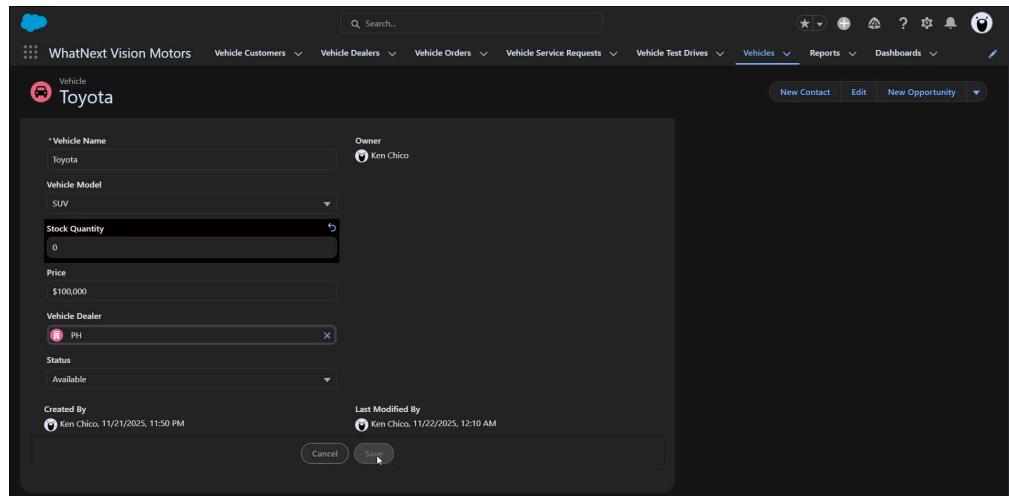
#### Test Case 1: Test Drive Reminder Flow

- Input: Create Vehicle Test Drive (Status=Scheduled, Date=Tomorrow, Customer=Ken).
- Expected Output: Email sent: "Reminder your test drive is tomorrow" to customer. Passed.

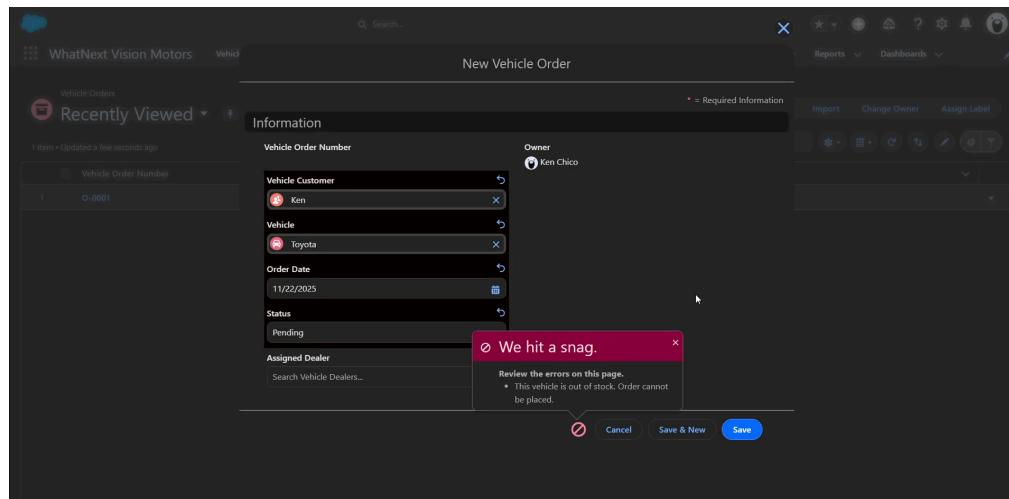


### Test Case 2: Apex Stock Prevention

- Input: Stock Quantity = 0, Create New Vehicle Order.

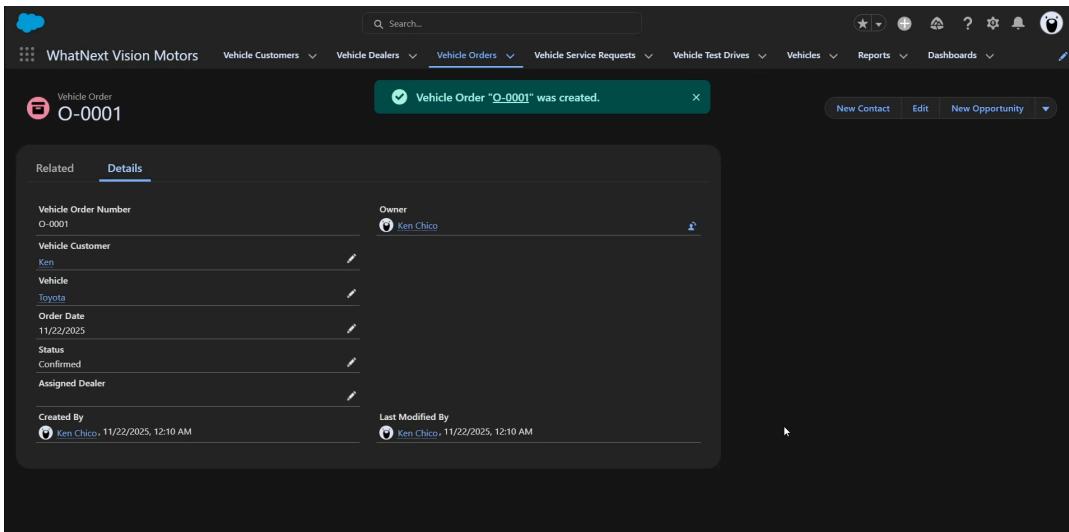


- Expected Output: Error "This vehicle is out of stock". Passed.

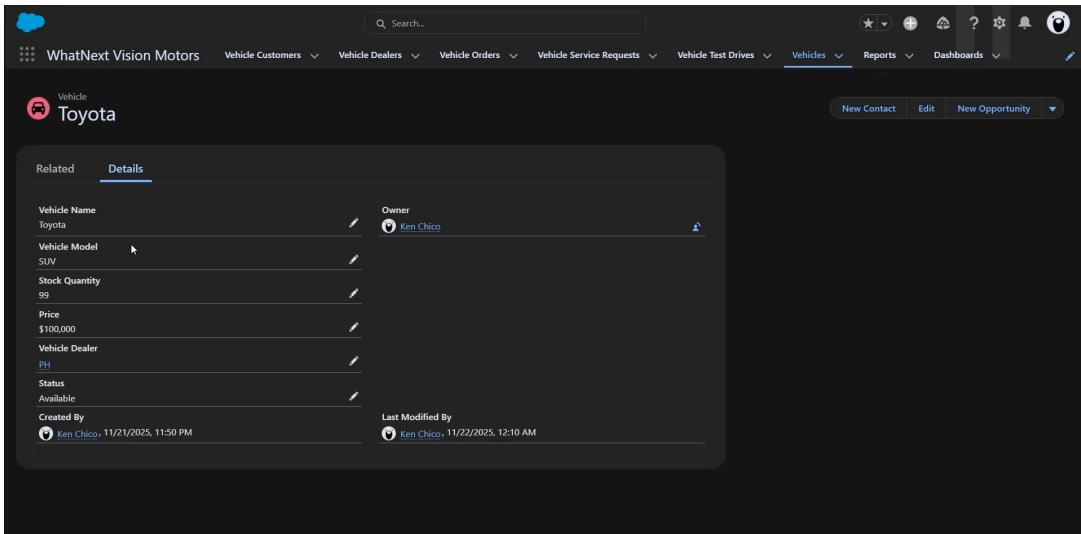


### Test Case 3: Stock Reduction

- Input: Vehicle stock=100; Create Order (Status=Confirmed).



- Expected Output: Stock reduced to 99. Passed.



Testing Approach: Manual testing via record creation; verify Flows via debug logs; Apex via record updates.

## Phase 5: Deployment, Documentation & Maintenance

### Deployment Strategy

- Use Change Sets from Developer Org to Production; includes objects, fields, Flows, Apex classes/triggers.

## System Maintenance and Monitoring

- Monitor Flows via Setup > Flows (activation status).
- Apex via Developer Console logs; Batch via scheduled jobs.

## Troubleshooting Approach

1. Check Flow debug logs for assignment/email failures.
2. Review Apex triggers for stock errors.
3. Validate data (addresses matching) for geo-assignment.

## Conclusion

The WhatNext Vision Motors CRM successfully automates core vehicle order processes, achieving efficient dealer assignment, stock management, and customer notifications through Salesforce native tools. All objectives met with Flows handling real-time automation and Apex ensuring data integrity.

### Future Enhancements:

- Integrate Einstein AI for dealer recommendation scoring.
- Add chatbot for test drive scheduling.
- Implement full Data Loader for bulk migrations.
- Mobile Lightning Web Components for customers.