

WhatNext Vision Motors – Vehicle Management System

July 15, 2025

Project Overview

The WhatNext Vision Motors Vehicle Management System is a custom Salesforce CRM solution designed to streamline the process of vehicle orders, test drives, and service requests. It centralizes vehicle stock tracking, automates dealer assignment, and ensures seamless interaction between customers and vehicle dealers. The solution enhances operational efficiency through automated flows, triggers, and dashboards.

Key Features:

- Custom Objects for Vehicles, Dealers, Customers, Orders, Test Drives, and Service Requests
- Automated Dealer Assignment
- Reminder Emails for Test Drives
- Order Stock Validation and Update via Apex
- Dynamic Reports & Dashboards

Objectives

The primary goal of this CRM is to automate and optimize the end-to-end vehicle management process for WhatNext Vision Motors.

Objectives include:

- Improve customer experience via timely service and test drive notifications.
- Automate order assignment and prevent overbooking.
- Maintain accurate stock levels and streamline dealer-customer interaction.
- Enable real-time reporting for better decision-making

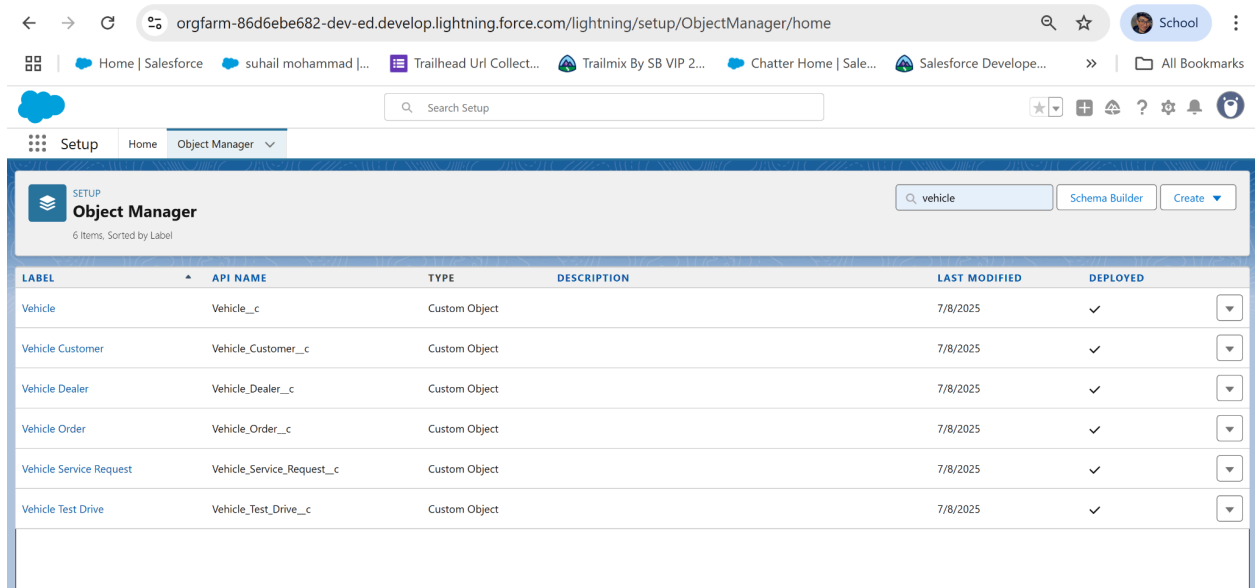
Phase 1: Requirement Analysis & Planning

Understanding Business Requirements:

- Customers need to request test drives, services, and place orders via CRM.
- Dealers need visibility into stock and customer requests.
- Admins need reports and automation to streamline operations.

Defining Project Scope & Objectives:

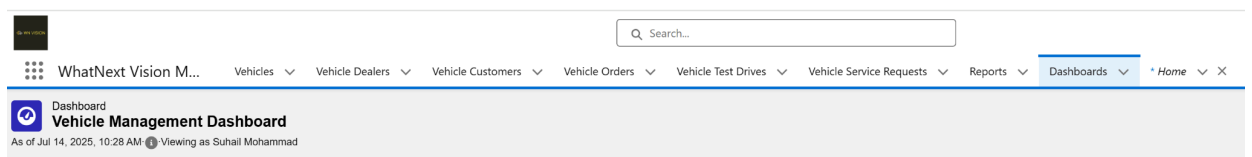
- Design a system with six custom objects (Vehicle, Dealer, Customer, etc.)



- Enable automation via flows and Apex for dealer assignment and stock validation.
- Provide dashboards for performance tracking.

Design Data Model & Security Model:

- Lookup relationships between objects to link dealers, customers, and vehicles.
- Field-level security enforced through profiles.
- Tabs created for each custom object for navigation and control.



Phase 2: Salesforce Development – Backend & Configurations

Setup & DevOps:

- Developer Console used for Apex class & trigger creation.
- Flow Builder for automation.

Customization:

- Created objects, fields (e.g., **Vehicle Model**, **Stock Quantity**, **Order Date**)

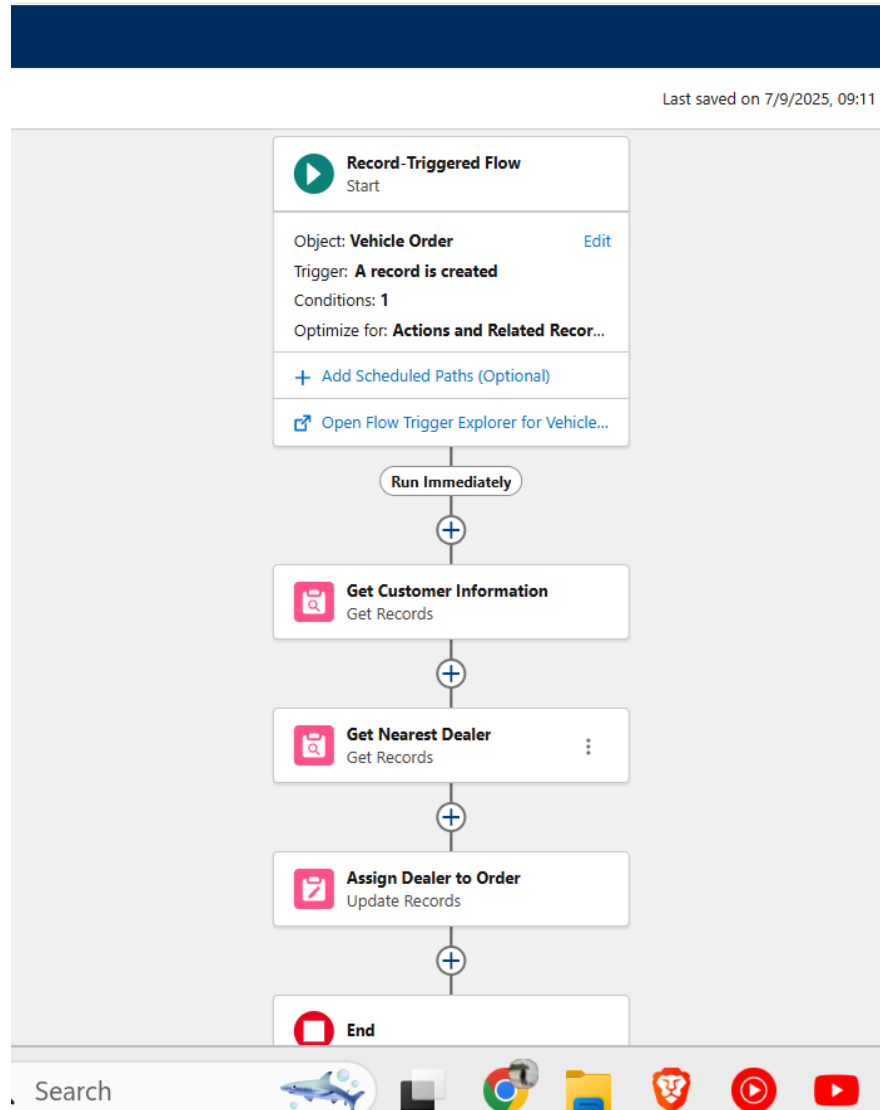
The screenshot shows the Salesforce Setup interface for the 'Vehicle' object. The 'Fields & Relationships' section is active, displaying a table of fields. The table has columns for Field Label, Field Name, Data Type, Controlling Field, and Indexed status. The fields listed are:

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(16, 2)		
Status	Status__c	Picklist		
Stock Quantity	Stock_Quantity__c	Number(18, 0)		
Vehicle Model	Vehicle_Model__c	Picklist		
Vehicle Name	Name	Text(80)		✓

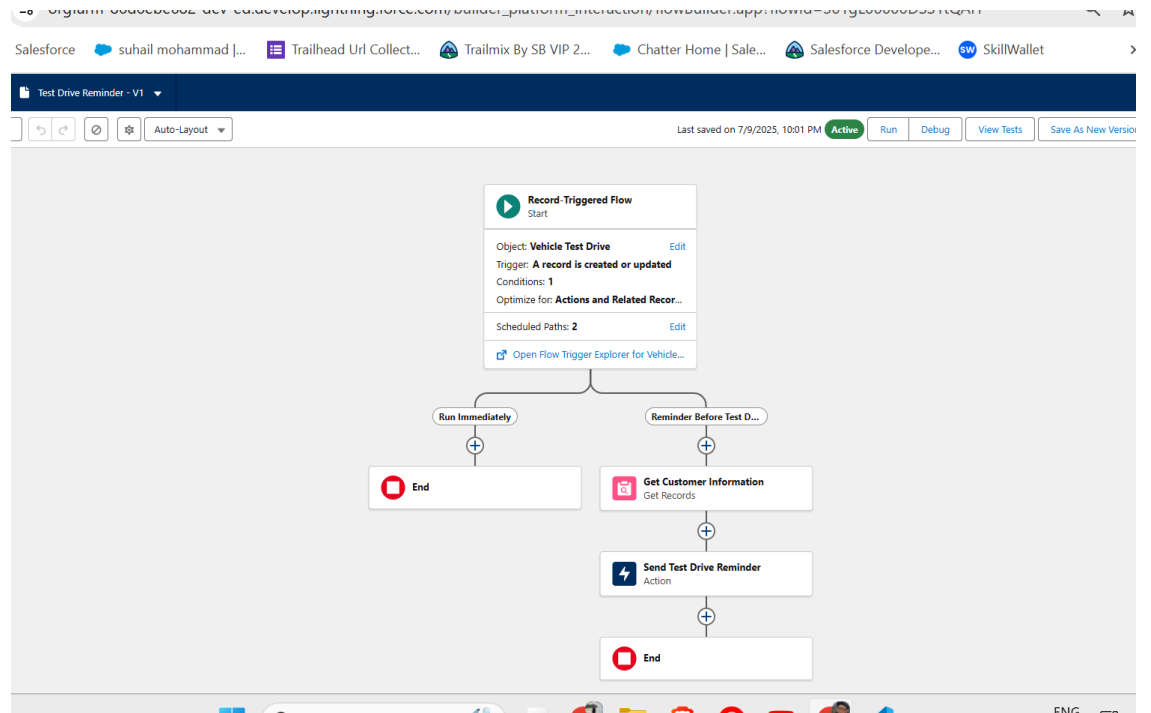
- Validation rule to prevent ordering if stock is unavailable.

5

- Flows:
 - Auto-assign dealer to orders



- Test drive reminder email



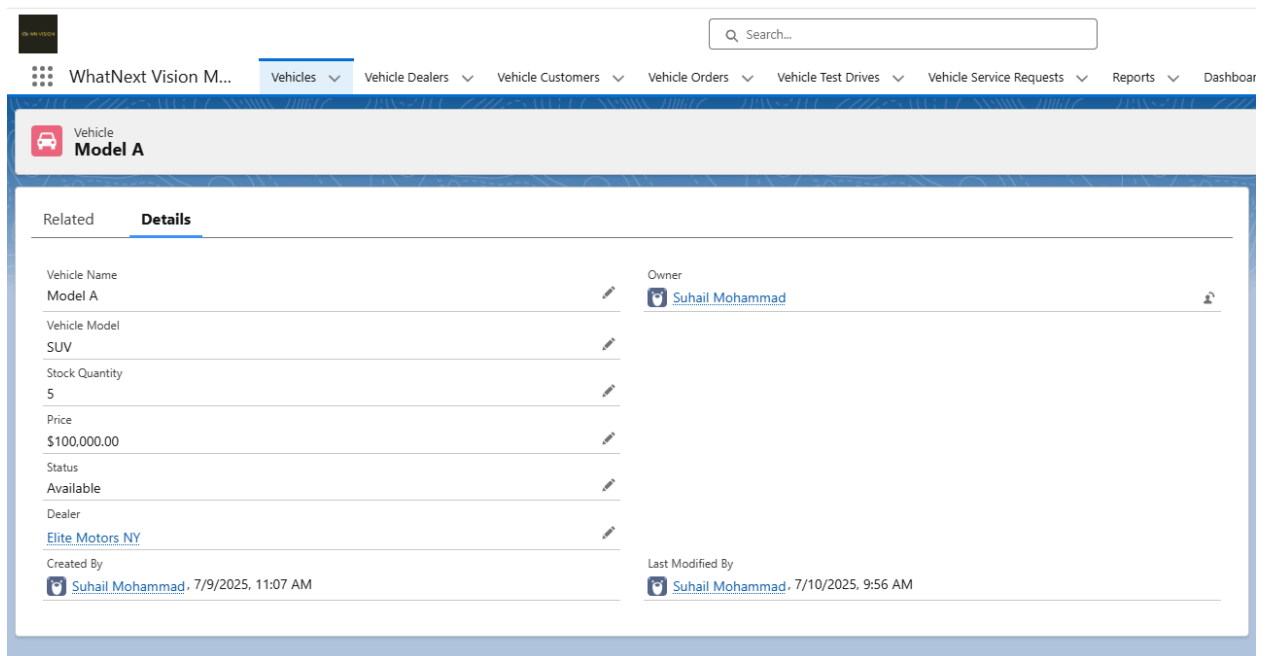
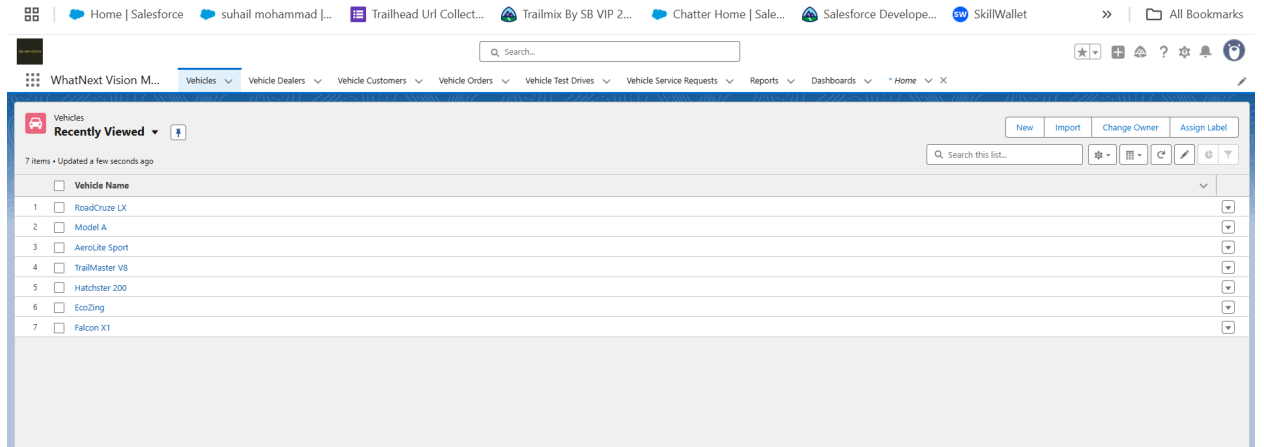
Apex:

- **VehicleOrderTriggerHandler**: Prevents orders when stock is 0 and updates stock.
- Trigger **VehicleOrderTrigger**: Calls handler methods on insert/update.
- Batch Class **VehicleOrderBatch**: Confirms orders and updates stock.
- Scheduled via **VehicleOrderBatchScheduler**.

Phase 3: UI/UX Development & Customization

Lightning App: WhatNext Vision Motors with all key tabs.

- Vehicle Tab



- Dealer Tab

WhatNext Vision M... Vehicles ▾ Vehicle Dealers ▾ Vehicle Customers ▾ Vehicle Orders ▾ Vehicle Test Drive

Vehicle Dealers
Recently Viewed ▾

4 items • Updated a few seconds ago

	<input type="checkbox"/> Dealer Name
1	<input type="checkbox"/> Elite Motors NY
2	<input type="checkbox"/> SpeedZone Dubai
3	<input type="checkbox"/> GreenDrive Mumbai
4	<input type="checkbox"/> Westside Autos

- Customer Tab

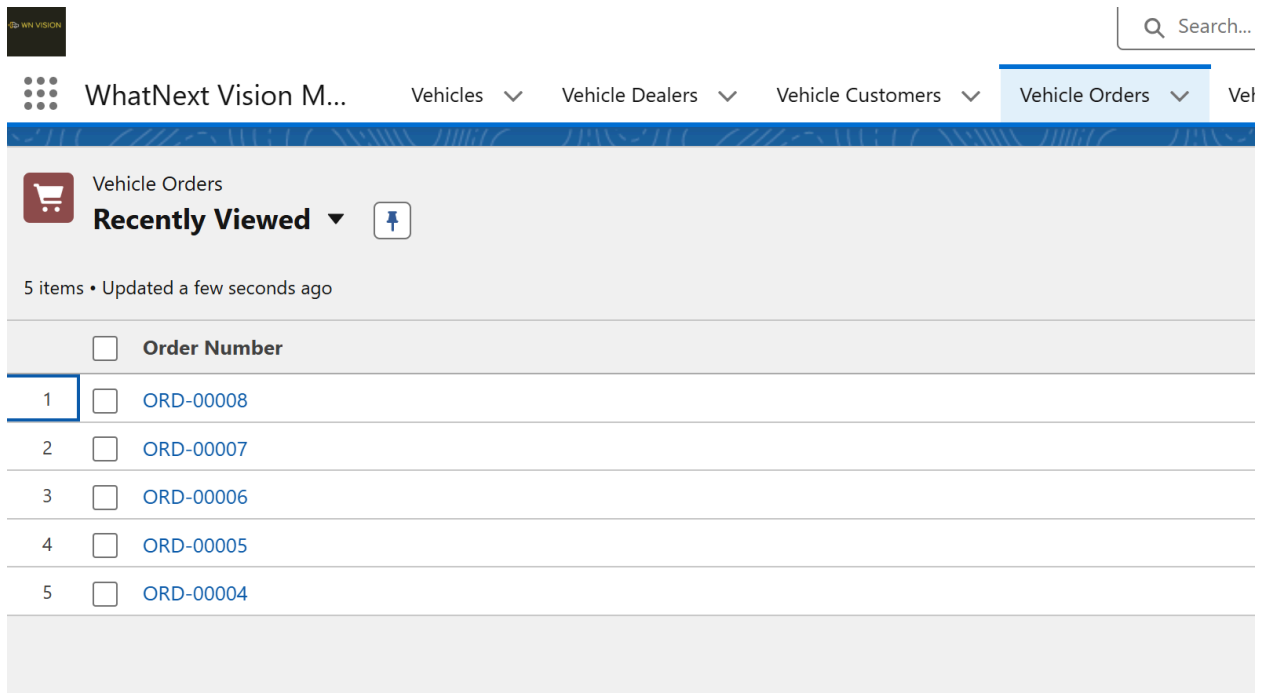
WhatNext Vision M... Vehicles ▾ Vehicle Dealers ▾ Vehicle Customers ▾ Vehicle Orders ▾ Vehicle Test Drive

Vehicle Customers
Recently Viewed ▾

5 items • Updated a few seconds ago

	<input type="checkbox"/> Customer Name
1	<input type="checkbox"/> Emma Johnson
2	<input type="checkbox"/> Omar Khalid
3	<input type="checkbox"/> Priya Mehta
4	<input type="checkbox"/> John Carter
5	<input type="checkbox"/> John Doe

- Order Tab



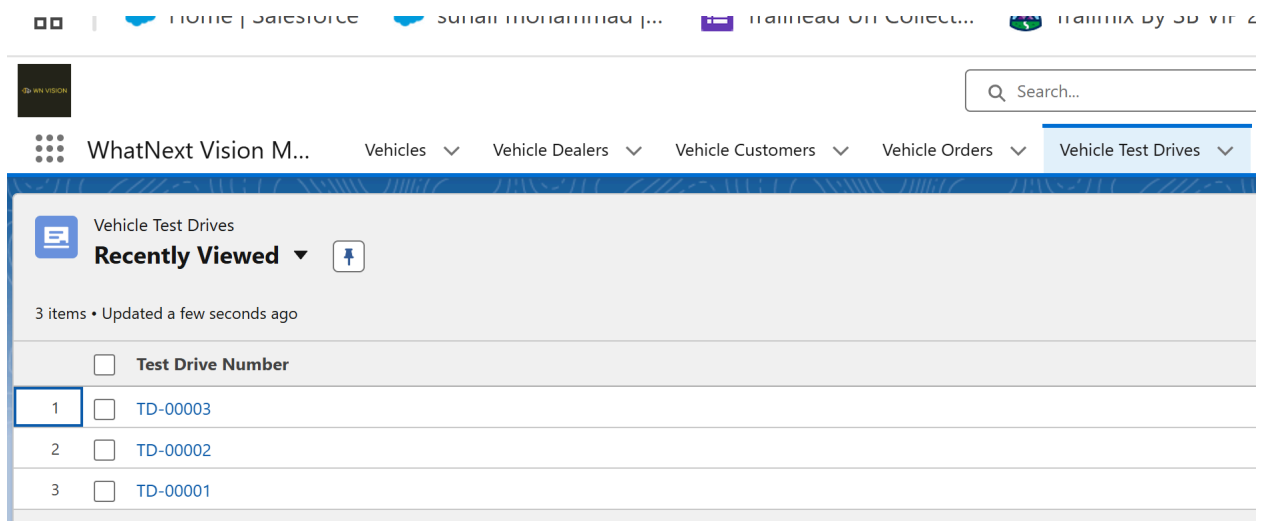
WhatNext Vision M... Vehicles Vehicle Dealers Vehicle Customers Vehicle Orders

Vehicle Orders
Recently Viewed ▼

5 items • Updated a few seconds ago

	<input type="checkbox"/> Order Number
1	<input type="checkbox"/> ORD-00008
2	<input type="checkbox"/> ORD-00007
3	<input type="checkbox"/> ORD-00006
4	<input type="checkbox"/> ORD-00005
5	<input type="checkbox"/> ORD-00004

- Test Drive Tab



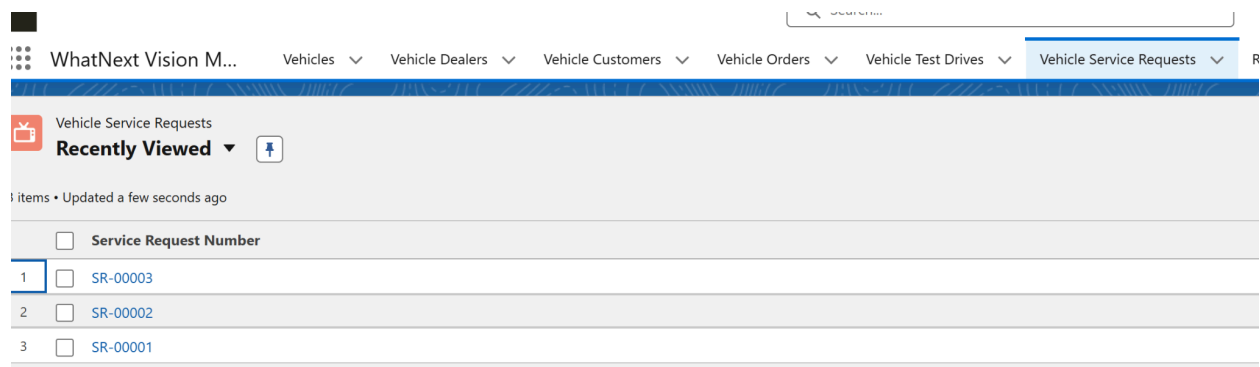
WhatNext Vision M... Vehicles Vehicle Dealers Vehicle Customers Vehicle Orders Vehicle Test Drives

Vehicle Test Drives
Recently Viewed ▼

3 items • Updated a few seconds ago

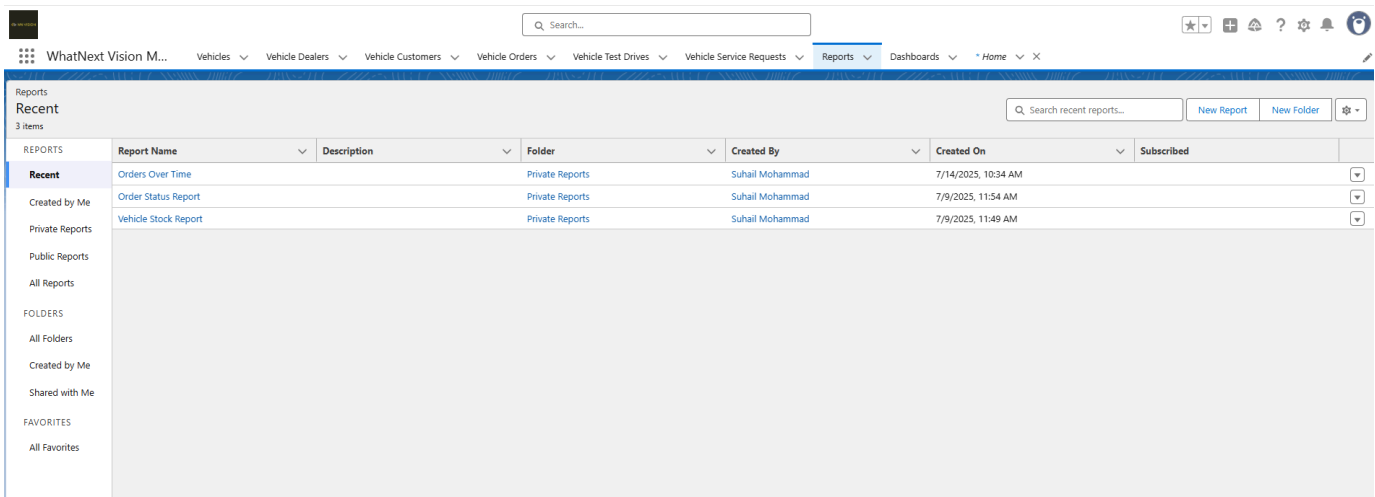
	<input type="checkbox"/> Test Drive Number
1	<input type="checkbox"/> TD-00003
2	<input type="checkbox"/> TD-00002
3	<input type="checkbox"/> TD-00001

- Service Request Tab




Page Layouts customized per object with relevant fields.


Reports & Dashboards:



- Vehicle Stock Report



WhatNext Vision M... Vehicles ▾ Vehicle Dealers ▾ Vehicle Customers ▾ Vehi

 Report: Vehicles

Vehicle Stock Report

Total Records
7

Total Stock Quantity
17

<input type="checkbox"/> Vehicle: Vehicle Name ↑ ▾	Stock Quantity ▾	Status ▾
<input type="checkbox"/> AeroLite Sport (1)	3	Available
Subtotal	3	
<input type="checkbox"/> EcoZing (1)	5	Available
Subtotal	5	
<input type="checkbox"/> Falcon X1 (1)	0	Available
Subtotal	0	
<input type="checkbox"/> Hatchster 200 (1)	2	Available
Subtotal	2	
<input type="checkbox"/> Model A (1)	5	Available
Subtotal	5	
<input type="checkbox"/> RoadCruze LX (1)	1	Out of Stock
Subtotal	1	
<input type="checkbox"/> TrailMaster V8 (1)	1	Available
Subtotal	1	
Total (7)	17	

- Order Status Report

WhatNext Vision M...

Search...

Vehicles

Vehicle Dealers

Vehicle Customers

Vehicle Orders

Vehicle Test Driv

Report: Vehicle Orders

Order Status Report

Total Records

5

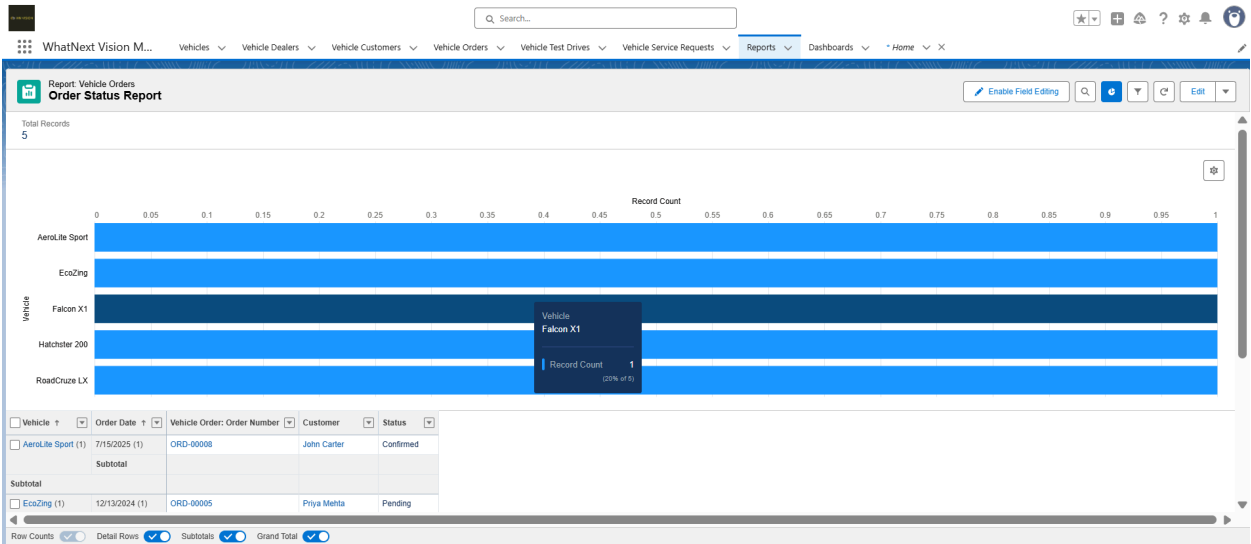
<input type="checkbox"/> Vehicle	<input type="checkbox"/> Order Date	<input type="checkbox"/> Vehicle Order: Order Number	<input type="checkbox"/> Customer	<input type="checkbox"/> Status
<input type="checkbox"/> AeroLite Sport (1)	7/15/2025 (1)	ORD-00008	John Carter	Confirmed
Subtotal				
Subtotal				
<input type="checkbox"/> EcoZing (1)	12/13/2024 (1)	ORD-00005	Priya Mehta	Pending
Subtotal				
Subtotal				
<input type="checkbox"/> Falcon X1 (1)	12/12/2024 (1)	ORD-00004	John Carter	Confirmed
Subtotal				
Subtotal				
<input type="checkbox"/> Hatchster 200 (1)	12/15/2024 (1)	ORD-00007	Emma Johnson	Delivered
Subtotal				
Subtotal				
<input type="checkbox"/> RoadCruze LX (1)	12/14/2024 (1)	ORD-00006	Omar Khalid	Canceled
Subtotal				
Subtotal				
Total (5)				

Row Counts

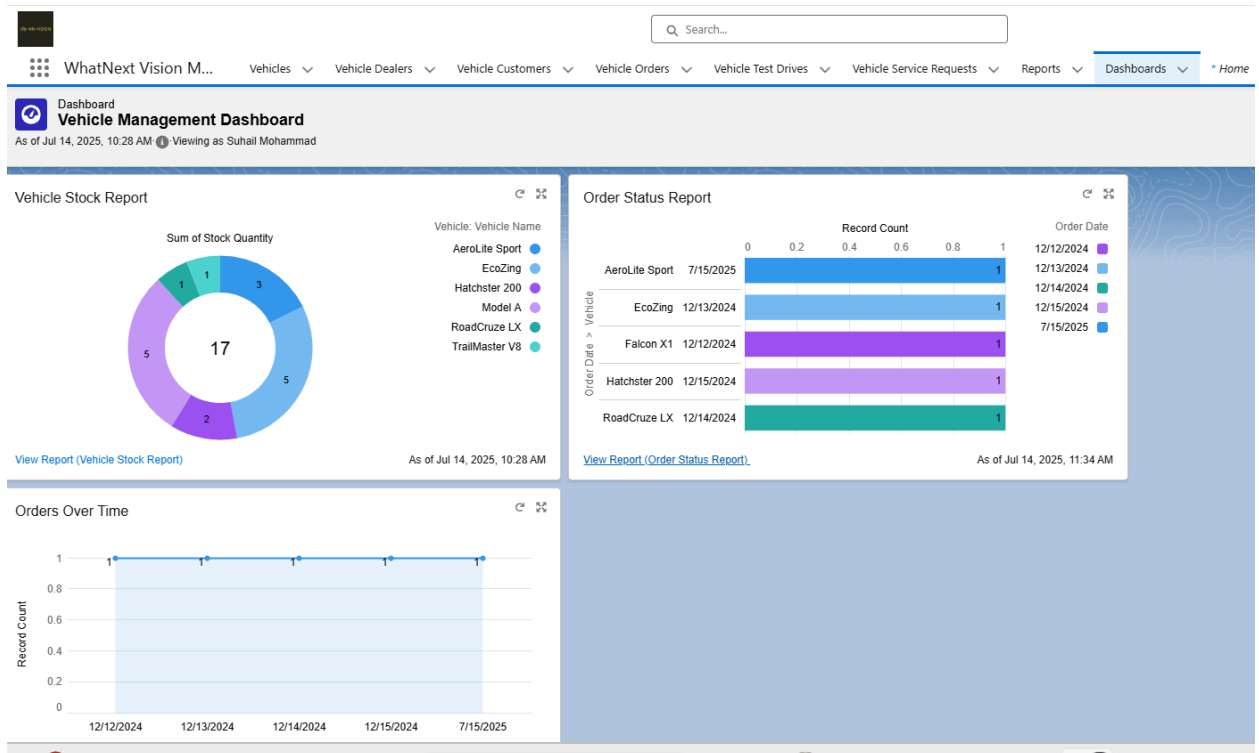
☒ Detail Rows

☒ Subtotals

☒ Grand Total



- Vehicle Management Dashboard



Phase 4: Data Migration, Testing & Security

Data Loading:

- Manual entry of test data via record pages.
- Could support Data Loader for bulk insert.

Field History, Matching & Duplicate Rules:

- Status fields used for tracking orders, services, and test drives.
- Matching Rules for dealers based on location.

Security Model:

- Roles: Admin, Dealer, Customer Service
- Profiles & Permission Sets created as needed.
- Sharing Rules applied for visibility control.

Testing:

- Created test cases:
 - **Order Creation:** Verified dealer auto-assignment.
 - **Test Drive Reminder:** Verified scheduled emails.
 - **Stock Validation:** Prevented order for out-of-stock vehicles.

The screenshot displays the 'New Vehicle Order' form. The form includes fields for Order Number, Customer (John Doe), Vehicle (Model A), Order Date (7/14/2025), Status (Pending), and Dealer (Elite Mot). An error message is shown: 'We hit a snag. Review the errors on this page. This vehicle is out of stock. Order cannot be placed.' The form also has buttons for Cancel, Save & New, and Save. A legend indicates that an asterisk (*) denotes required information.

New Vehicle Order

* = Required Information

Information

Order Number

Owner
Suhail Mohammad

Customer
John Doe

Vehicle
Model A

Order Date
7/14/2025

Status
Pending

Dealer
Elite Mot

We hit a snag.

Review the errors on this page.

- This vehicle is out of stock. Order cannot be placed.

Cancel Save & New Save

Phase 5: Deployment, Documentation & Maintenance

Deployment Strategy:

- Components moved using Change Sets.
- Batch jobs scheduled using Anonymous Apex.

Maintenance:

- Regular monitoring of Scheduled Jobs & Error Logs.
- Logs tracked via Debug Logs and Scheduled Job Monitoring.

Troubleshooting:

- Trigger issues handled via logs & debug points.
- Batch failures debugged using test runs

Future Enhancements

- Chatbot for vehicle selection assistance.
- AI recommendation engine for suggesting vehicles.
- Customer portal with self-service options.

Conclusion

The Vehicle Management System for WhatNext Vision Motors successfully integrates multiple Salesforce features to build a functional and scalable CRM. It automates business processes, improves user experience, and provides actionable insights through reports. The project demonstrates effective use of Flow, Apex, Data Modeling, and UI customization.

Annexure

Codes:

VehicleOrderTriggerHandler

```
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:

```

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);
}

```

VehicleOrderBatch:

```
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);

    }

    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:

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