

Salesforce Capstone Project On

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

1. Project Overview

WhatNext Vision Motors, an emerging leader in the automotive sector, aimed to modernize its customer interaction and operational processes through the implementation of a customized Salesforce CRM solution. The primary focus of the project was to streamline vehicle order management, ensure accurate dealer assignment, and enhance customer engagement via automation.

The previous manual processes often led to delays, stock mismanagement, and customer dissatisfaction. To overcome these challenges, the new CRM system was designed with features such as real-time stock validation, automatic dealer assignment based on customer location, test drive reminders via email, and backend automation through Apex triggers and batch classes.

The CRM platform also provides a user-friendly interface using Lightning Apps and Dynamic Forms, ensuring an efficient experience for internal users. Overall, the solution improves efficiency, reduces errors, and lays a scalable foundation for future enhancements like AI-based vehicle recommendations or chatbot support.

2. Objectives

The key objectives of the Salesforce implementation are:

- Enhance customer experience during the vehicle ordering process
- Automatically suggest the nearest dealer based on customer address
- Prevent orders for out-of-stock vehicles
- Automate order status updates based on stock availability
- Improve operational efficiency through automation

3. Scope of the Project

In Scope:

- Customer order creation in Salesforce
- Dealer recommendation based on customer location
- Stock availability validation
- Automated order status updates
- Scheduled batch processing for bulk orders

Out of Scope:

- Payment gateway integration
- Third-party logistics systems
- Vehicle delivery tracking

4. Functional Requirements

4.1 Dealer Recommendation

- The system automatically identifies and suggests the nearest dealer location based on the customer's address.
- This reduces customer effort and speeds up the ordering process.

4.2 Stock Availability Validation

- Customers are restricted from placing orders for vehicles that are out of stock.
- Only vehicles with available inventory can be ordered.

4.3 Order Status Management

- Orders are assigned statuses based on stock availability:
 - **Confirmed** – Vehicle is in stock
 - **Pending** – Vehicle is out of stock

5. Proposed Salesforce Solution

The Salesforce solution provides an automated and intelligent ordering system that:

- Identifies the nearest dealer using customer location data
- Validates vehicle stock availability before order creation
- Uses scheduled automation to update order statuses
- Provides real-time visibility into order fulfillment status

6. Salesforce Platform Overview

Salesforce is a cloud-based Customer Relationship Management (CRM) platform that provides tools for sales automation, customer service, analytics, and application development.

Salesforce Features Used:

- Standard Objects (Account, Contact, Order)
- Custom Objects
- Salesforce Flow
- Apex Programming
- Validation Rules
- Scheduled Automation
- Role Hierarchy & Security

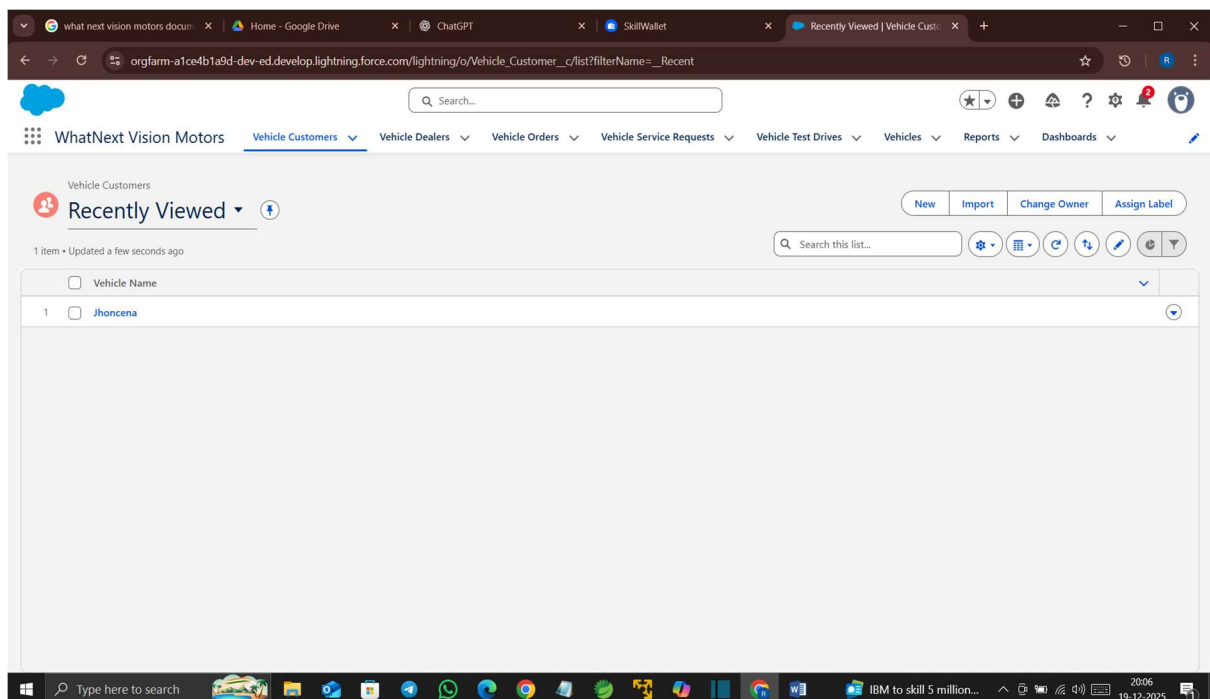
7. System Architecture

7.1 Architecture Description

The Salesforce system architecture consists of:

- Salesforce Org as the central platform
- Customer interface for order creation
- Dealer and vehicle inventory management modules
- Automation layer using Flows and Apex
- Scheduled jobs for bulk processing

This layered architecture ensures scalability, performance, and maintainability.



8. Data Model & Object Design

8.1 Standard Objects

Account

- Stores customer account details

Contact

- Stores individual customer information

Order

- Represents customer vehicle orders

Order Item

- Stores vehicle details associated with orders

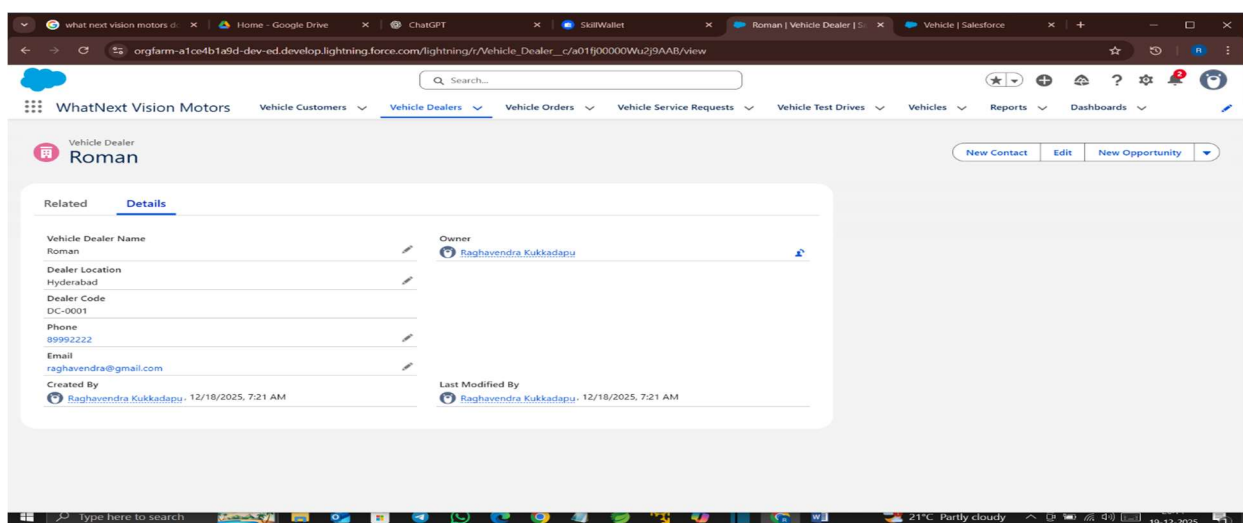
8.2 Custom Objects

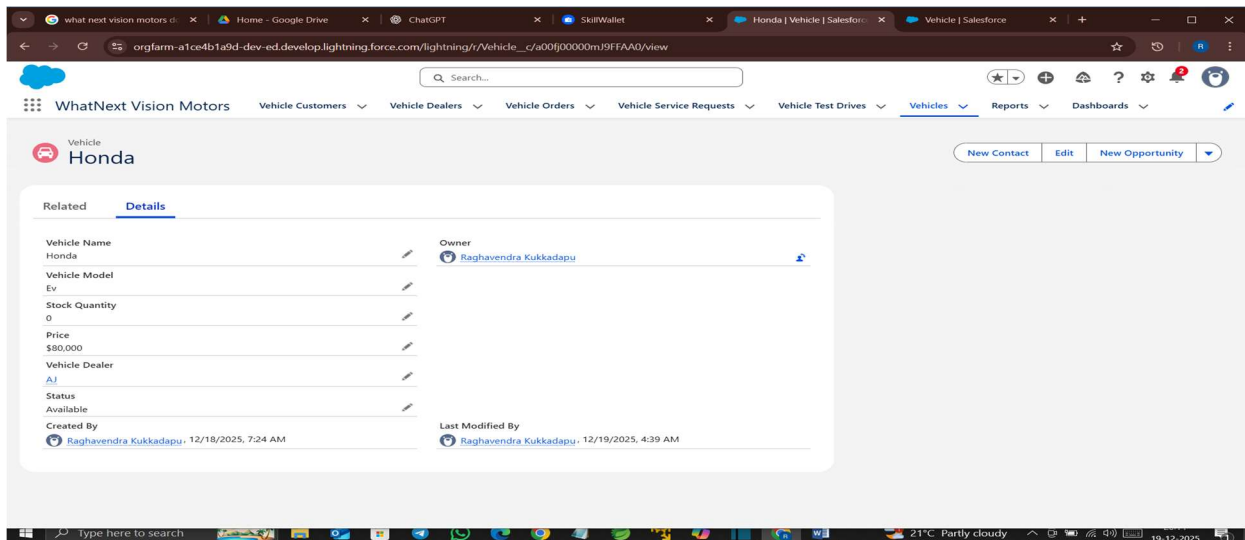
Vehicle__c

- Vehicle Name
- Model
- Stock Quantity
- Stock Status

Dealer__c

- Dealer Name
- Location
- Distance
- Available Vehicles





8.3 Object Relationships

- Dealer__c → Vehicle__c (One-to-Many)
- Order → Order Item (One-to-Many)
- Order Item → Vehicle__c (Lookup)

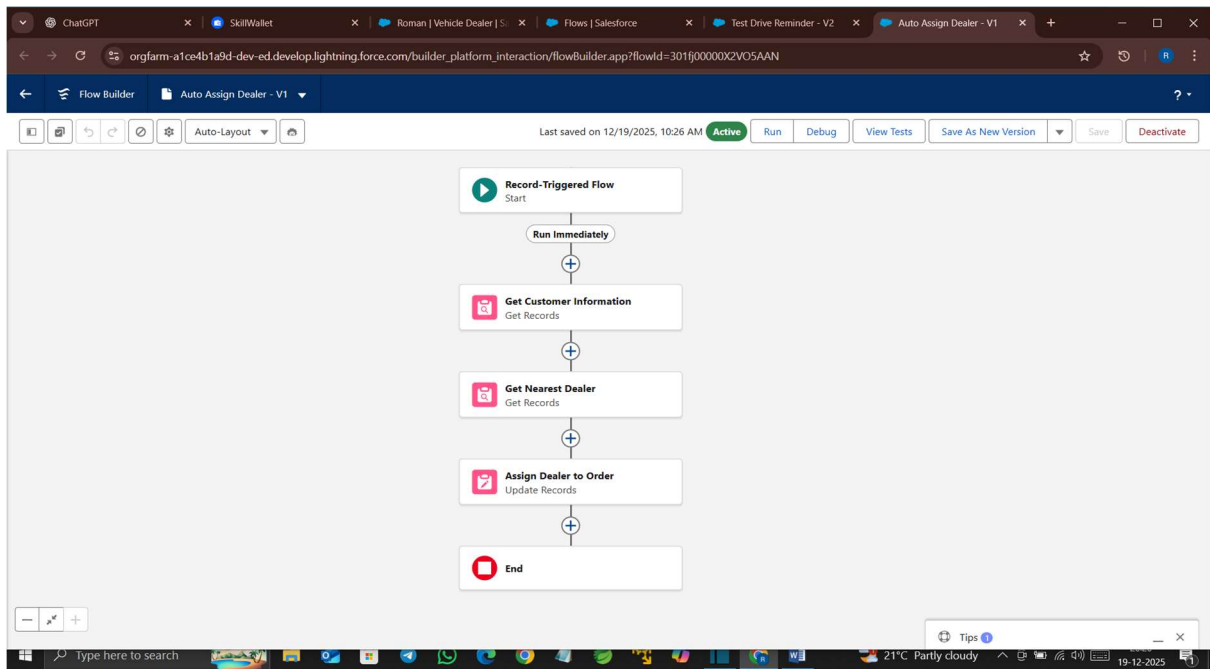
9. Business Process Overview

The Salesforce system supports the end-to-end ordering lifecycle:

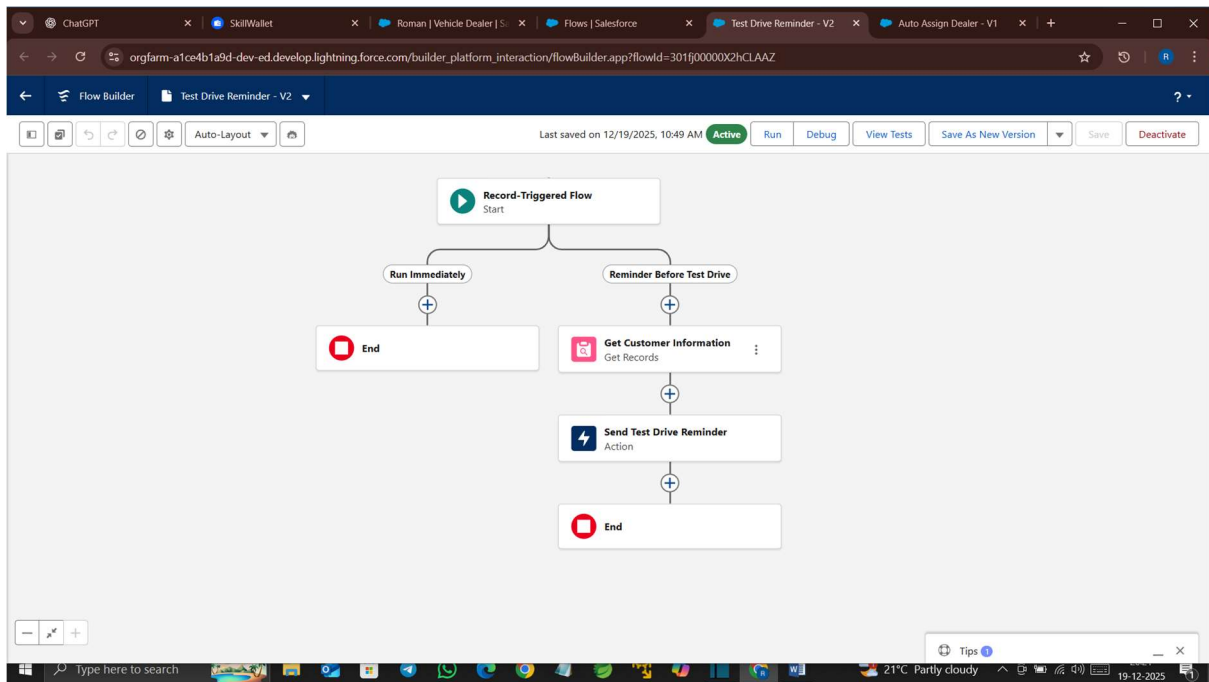
1. Customer initiates order
2. Dealer is recommended automatically
3. Vehicle availability is checked
4. Order is created or blocked
5. Order status is updated automatically
6. Customer receives accurate order information

10. Process Flow Diagrams

Auto Assign Dealer



Test Drive Remainder



10.1 Order Creation Flow

Steps:

1. Customer enters address
2. Nearest dealer identified
3. Vehicle list filtered by availability
4. Stock validation occurs
5. Order is created only if stock exists

10.2 Order Status Update Flow

Steps:

1. Scheduled process runs
2. Stock availability is checked
3. Order status updated automatically
4. Bulk orders handled efficiently

11. Automation Design

11.1 Salesforce Flow

- Record-triggered Flow for order validation
- Scheduled Flow for periodic updates
- Reduces manual intervention

11.2 Apex Automation

- Apex Scheduler for bulk processing
- Batch logic ensures performance

- Handles large datasets efficiently

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 && (isInsert || isUpdate)) {
            preventOrderIfOutOfStock(newOrders);
        }

        if (isAfter && (isInsert || isUpdate)) {
            updateStockOnOrderPlacement(newOrders);
        }
    }
}
```

```
// ✗ Prevent placing an order if stock is zero
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>(
            [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id
            IN :vehicleIds]
        );

        for (Vehicle_Order__c order : orders) {
            Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
            if (vehicle != null && vehicle.Stock_Quantity__c <= 0) {
                order.addError('This vehicle is out of stock. Order cannot be
placed.');
```

```
            }
        }
    }
}

// ✔ Decrease stock when an order is confirmed
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>(
        [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id
IN :vehicleIds]
    );

    List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();
    for (Vehicle_Order__c order : orders) {
        Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
        if (vehicle != null && 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>(
            [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.');
```

VehicleOrderBatchScheduler:-

```

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

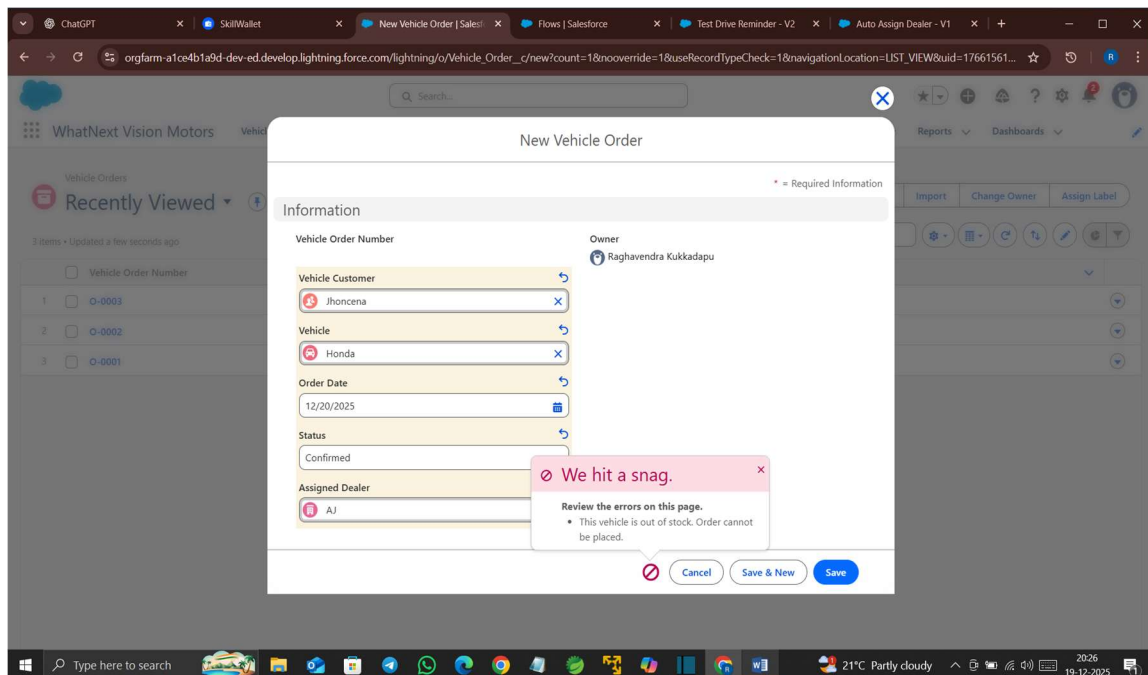
```

12. Validation & Business Rules

Key Validation Rules:

- Prevent order creation when stock = 0
- Mandatory dealer assignment
- Mandatory vehicle selection

These rules ensure data integrity and system reliability.



13. Security Model

13.1 User Roles

- System Administrator
- Sales Executive
- Inventory Manager

13.2 Security Features

- Role-based access
- Field-level security

- Object-level permissions

Sensitive inventory data is accessible only to authorized users.

14. Reporting & Dashboards

Salesforce reports provide insights into:

- Order volume
- Stock availability
- Dealer performance
- Pending vs confirmed orders

15. Testing Strategy

15.1 Types of Testing

- **Unit Testing:** Validates individual Apex classes, triggers, and validation rules.
- **Integration Testing:** Ensures seamless interaction between Orders, Vehicles, and Dealers.
- **Flow Testing:** Confirms correct execution of automated flows and scheduled processes.
- **User Acceptance Testing (UAT):** Business users validate functionality against requirements.

15.2 Sample Test Cases

- Order creation with available stock
- Order creation with unavailable stock
- Scheduled job execution
- Dealer recommendation accuracy

16. Deployment Strategy

- Development in Sandbox
- Testing completed successfully
- Change Sets used for deployment
- Post-deployment validation performed

17. Benefits of the System

Customer Benefits

- Faster vehicle ordering
- Clear order status visibility
- Reduced confusion and delays
- Enhanced overall experience

Business Benefits

- Reduced manual errors
- Improved operational efficiency
- Lower administrative costs
- Better reporting and control

18. Limitations

- No payment processing
- No real-time logistics tracking
- Dependent on data accuracy

19. Future Enhancements

- Payment gateway integration
- Mobile application
- AI-based recommendations
- Advanced analytics
- ERP integration

20. Conclusion

The Salesforce-based solution at WhatsNext Vision Motors modernizes vehicle ordering and inventory management. By leveraging Salesforce automation, validation rules, flows, and scheduled jobs, the organization achieves higher efficiency, transparency, and customer satisfaction.

This project demonstrates the practical application of Salesforce in solving real-world automotive challenges and provides a scalable foundation for future digital transformation.