Title: – CRM-Based Vehicle Ordering and Dealer Automation for WhatsNext Vision Motors

Project Overview:

WhatsNext Vision Motors has modernized its customer experience and operational workflow by implementing a Salesforce-based CRM system for end-to-end vehicle ordering and dealership management. The system allows customers to place vehicle orders, schedule test drives, and raise service requests while ensuring stock accuracy and automatic dealer assignment based on customer location.

Advanced automation features such as Record-Triggered Flows, Apex Triggers, Batch Apex, and Scheduled Apex ensure seamless order management and real-time updates. The CRM enhances business productivity by validating orders against vehicle stock, calculating order status dynamically, auto-assigning orders to the nearest dealer, and triggering timely reminders for scheduled test drives.

This automation improves customer satisfaction, reduces processing time, prevents manual errors, and delivers a unified customer-centric buying experience.

Objectives:

- Automation eliminates manual data entry and reduces order processing time. This ensures faster and seamless customer experience during vehicle purchase.
- The system intelligently detects customer address and maps the order to the closest dealership.
  This reduces delivery delays and improves customer convenience.
- Real-time stock validation blocks orders for unavailable vehicles during booking. This prevents customer dissatisfaction and avoids incorrect order commitments.
- Scheduled Apex automatically sends reminder emails before a test drive appointment.
  This increases customer engagement and reduces test drive no-show cases.
- Batch jobs periodically update stock levels and confirm pending orders automatically.
  This ensures accurate inventory tracking and smooth order flow without manual effort.
- The CRM system delivers an error-free order cycle with fast communication and transparency.
  This results in happier customers and more efficient business operations.

Student Outcomes:

- Hands-on Salesforce CRM development experience using Apex, Flows, and Lightning UI
- Expertise in building automated order and stock management systems
- Understanding of data modeling for real-world business processes
- Experience in designing approval-free automation and dealer distribution workflows
- Knowledge of Batch Apex and Scheduled Apex for bulk order updates
- Ability to integrate CRM with customer interactions like test drives and servicing

System Requirements:

Hardware Requirements:

- Computer with minimum 4 GB RAM, Dual-core processor
- Stable internet connection

Software Requirements:

- Salesforce Developer Edition Org
- Modern Web Browser (Chrome / Firefox)

Skills Required:

- Salesforce Data Modeling & Configuration
- Lightning App Builder
- Security & Access Management
- Apex Classes & Trigger Handlers
- Batch Apex & Scheduled Apex
- Record Triggered Flows

Phases Overview:

Phase No.	Phase Name	Description	Page Numbers
1	Requirement Analysis & Planning	Identify business requirements for CRM vehicle ordering and workflow automation	4-8
2	Salesforce Development – Backend & Configuration	Create objects, fields, automations, triggers and batch logic	8-38
3	UI/UX Development & Customization	Build an intuitive CRM app with Lightning pages and layouts	39-42
4	Data Migration, Testing & Security	Load data, test system reliability, and configure system-level security	42-43
5	Deployment, Documentation & Maintenance	Deploy CRM app, monitor performance and maintain system health	43-46

Project Main Overview:

WhatsNext Vision Motors has built a vehicle management portal on Salesforce CRM to provide a seamless customer experience for ordering, test drives, and post-purchase services. The CRM ensures each customer order is automatically assigned to the nearest dealer based on address, while preventing order creation for vehicles that are out of stock. The data model includes Vehicle__c, Vehicle_Order__c, Vehicle_Dealer__c, Vehicle_Customer__c, Vehicle_Test_Drive__c, and Vehicle_Service_Request__c objects for storing vehicle information, stock availability, dealerships, orders, test drives, and servicing.

Through Apex Triggers and Batch Apex, the system validates stock before order confirmation and updates bulk pending orders automatically when stock replenishes. Record-Triggered Flows trigger dealer assignment and test drive reminders, while Scheduled Apex processes pending orders daily.

Overall, the CRM improves customer satisfaction, minimizes manual operational tasks, and delivers an automated, intelligent, and scalable solution for the Automotive industry

Main Objectives

- Automatic dealer assignment based on customer location
- Prevent order placement for vehicles with zero stock
- Scheduled order status update through batch processing
- Automated email reminders for scheduled test drives
- Maintain accurate test drive, service request, and order life-cycle history
- Provide real-time dashboards for orders, stock, and dealer activity
- Role-based access control for admin, dealers, and customers
- Deliver a unified digital vehicle ordering and servicing experience

Phase 1: Requirement Analysis & Planning:

1. Understanding Business Requirements:

Objective:

Understand how automotive companies manage the complete vehicle purchase cycle including vehicle stock availability, dealer assignment, customer coordination, and scheduled services and identify operational challenges that reduce efficiency and customer satisfaction.

The goal is to build a Salesforce-based solution that automates order processing, stock validation, and dealer mapping while providing centralized visibility and analytics for the management of vehicle orders.

Approach:

- Gather and analyze requirements from Sales Managers, Dealer Partners, Service Coordinators, and Customers to understand the current vehicle ordering workflow and pain points.
- Study how customers select vehicles, how dealers confirm orders, and how stock availability is tracked across multiple branches.
- Identify challenges such as wrong dealer mapping, delayed order confirmation, manual stock updates, scheduling follow-ups, and lack of automated communication.
- Conduct requirement study using ChatGPT, Google, Salesforce Documentation, and Trailhead to design a scalable and secure CRM-based Vehicle Management System.

Key Business Requirements Identified:

- Provide a Salesforce CRM application for placing and managing vehicle orders.

- Auto-assign orders to the nearest dealer based on customer location.
- Restrict order booking when vehicle stock is unavailable.
- Enable customers to schedule test drives through CRM and receive reminders.
- Allow managers to track stock availability, pending orders, and test-drive schedules through dashboards.
- Ensure secure, role-based visibility for Admin, Dealer, and Customer logins.
- Generate analytical dashboards and reports for the sales team to monitor order trends and customer engagement.

2. Defining Project Scope & Objectives:

Project Scope:

- Build a Salesforce-based Vehicle Ordering System to automate customer orders, dealer assignment, stock validation, and scheduled test drive reminders.
- Integrate automation technologies such as Flows, Apex, and Batch Apex for smooth processing and real-time updates.
- Provide customers with a self-service interface to place orders and track test-drive bookings.
- Implement security and access controls for Admins, Dealers, and Customers.
- Allow sales managers to monitor vehicle stock, order confirmation rates, and conversion insights through dashboards.

Objectives Summary:

- Simplify the vehicle ordering workflow and remove manual follow-up processes.
- Improve customer experience through automated dealer assignment.
- Prevent stock-related errors by validating inventory before order creation.
- Increase productivity through automation of reminders and order-status updates.
- Ensure secure access and protected customer data using profiles and role-based visibility.
- Support management decision-making using CRM dashboards and analytics.

3. Gathering & Analyzing User Needs

Users Involved:

- Customers: Place vehicle orders, schedule test drives, check order status.
- Dealers: View assigned orders, update order progress, manage delivery.
- Sales Managers: Monitor stock availability, dealer performance, and order fulfillment.

- System Administrator: Configure user access, monitor workflows, and ensure data security.

Key Functional Needs:

- Intuitive customer ordering screen and test drive booking interface.
- Automatic dealer assignment based on customer address / location.
- Stock validation before confirming an order.
- Dealer dashboard to view assigned orders and update order status.
- Automated email alerts for test drive reminders and order status updates.
- Reports and dashboards to track orders, stock, and dealer performance.

Tools Used:

- Google Forms – To collect business and user requirements from sales and dealer teams.
- Miro Boards – To visualize customer ordering and dealer assignment workflows.
- User Personas – To design personalized experiences for customers, dealers, and sales managers.
  (Tools listed only for real-time project standard documentation.)

4. Identifying Key Salesforce Features & Tools Required

Salesforce Features Planned:

● Custom Objects:

- Vehicle__c → Stores vehicle information and stock quantity.
- Vehicle_Dealer__c → Stores dealer details and location.
- Vehicle_Customer__c → Stores customer contact and address details.
- Vehicle_Order__c → Tracks orders placed by customers and dealer assignment.
- Vehicle_Test_Drive__c → Tracks test drive scheduling and completion.
- Vehicle_Service_Request__c → Tracks servicing and maintenance requests.

● Standard Object:

- User → Represents Admin, Dealer, and other users.

● Automations:

- Record-Triggered Flows for dealer auto-assignment and email reminders.
- Scheduled Flows for date-based notifications.

- **Apex:**
  - Trigger Handler for stock validation and stock reduction.
  - Batch Apex for auto-updating pending orders after stock refill.
  - Scheduled Apex for batch execution.
- **UI:**
  - Lightning App Pages and Dynamic Forms for vehicle, order, and test drive screens.
- **Email Services:**
  - Email templates and alerts for reminders and order status updates.
- **Security:**
  - Profiles, Permission Sets, Role Hierarchy, Field-Level Security, and component visibility.

5. Designing Data Model and Security Model:

Data Model Includes:

1. **Vehicle__c (Custom Object)**
   - Stores vehicle model and stock quantity.
   - Linked to Dealer for stock distribution.
   - Includes fields like Price, Model, Status, and Stock Quantity.
2. **Vehicle_Order__c (Custom Object)**
   - Tracks customer order details.
   - Linked to Customer and Vehicle.
   - Includes fields like Order Date, Dealer, Status.
3. **Vehicle_Customer__c (Custom Object)**
   - Stores customer identity and address.
   - Used for order and test drive records.
4. **Vehicle_Dealer__c (Custom Object)**
   - Stores dealer information and location.
   - Used to auto-assign orders based on proximity.
5. **Vehicle_Test_Drive__c (Custom Object)**
   - Tracks test drive bookings.
   - Linked to customer and vehicle.

6. **Security Model Design:**
    - Role Hierarchy: Admin → Dealer → Customer
    - Profiles: Admin Profile, Dealer Profile, Customer Profile
    - Record-Level Security: Sharing rules provide dealers access only to assigned orders
    - Field-Level Security: Protects sensitive customer and vehicle data
    - Component Visibility: Shows relevant pages dynamically based on login role

Summary:

This phase enabled a comprehensive understanding of the automotive ordering ecosystem and identified areas where automation can reduce manual effort.After analyzing the requirements of customers, dealers, sales teams, and management, a clear project scope and system architecture were defined. The data model and security framework ensure scalable automation, secure data-flow, and smooth coordination between customers, dealers, and administrative users.

Phase 2: Salesforce Development – Backend & Configurations:

This phase establishes the core system logic for CRM vehicle ordering.

Major Configurations:

- Custom objects and fields for vehicles, orders, test drives and services
- Validation rules to prevent invalid or duplicate entries
- Flows for automatic dealer assignment & test-drive reminders
- Apex Trigger Handlers for stock validation and stock reduction
- Batch Apex for updating pending orders when stock refills
- Scheduled Apex to execute batch job daily

Milestone 1: Salesforce Account:

Introduction:

Are you new to Salesforce? Not sure exactly what it is, or how to use it? Don't know where you should start on your learning journey? If you've answered yes to any of these questions, then you're in the right place. This module is for you. Welcome to Salesforce! Salesforce is game-changing technology, with a host of productivity-boosting features, that will help you sell smarter and faster.

What Is Salesforce?

Salesforce is your customer success platform, designed to help you sell, service, market, analyze, and connect with your customers.

Activity 1: Creating Developer Account:

Creating a developer org in salesforce.

1. Go to https://developer.salesforce.com/signup
2. On the sign-up form, enter the following details:

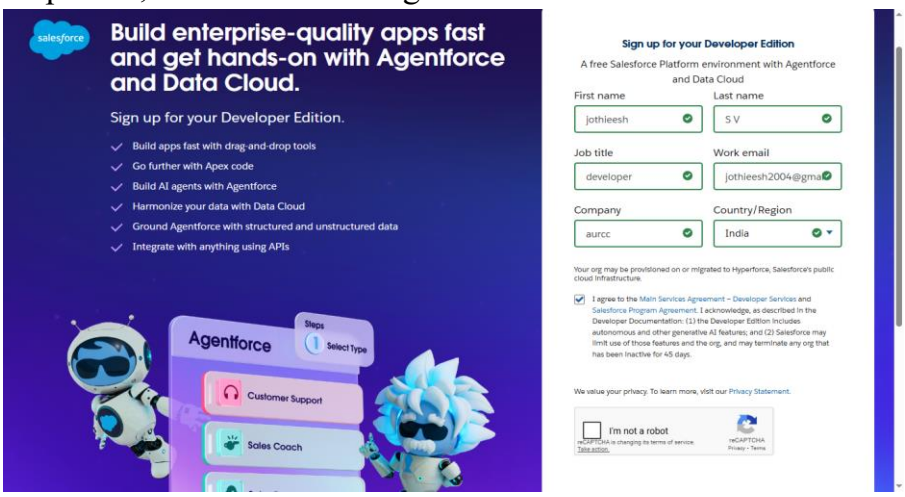


Fig: SignUp

1) First name & Last name
2) Email
3) Job Title: Developer
4) Company: College Name
5) Country: India

Activity 2: Account Activation

1. Go to the inbox of the email that you used while signing up. Click on the verify account to activate your account. The email may take 10-30mins and sometimes 2 hours.

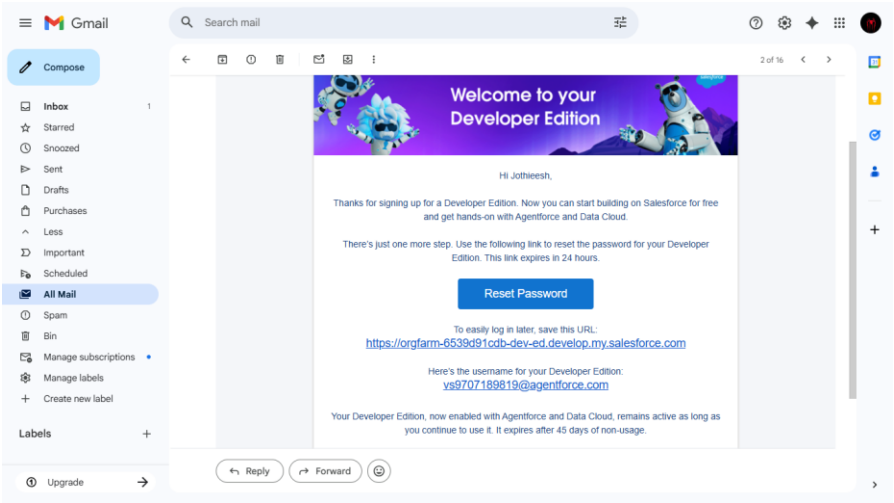


Fig: Reset Password

2. Click on Verify Account
3. Give a password and answer a security question and click on change password.

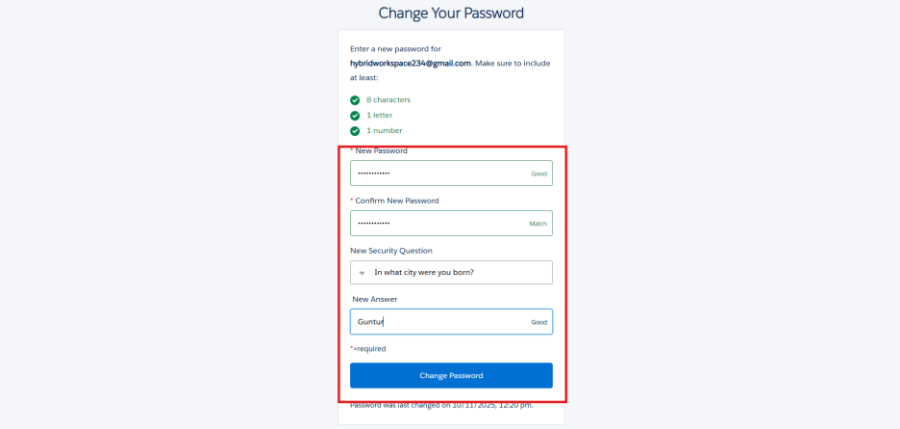


Fig: New Password

4. Then you will redirect to your salesforce setup page.

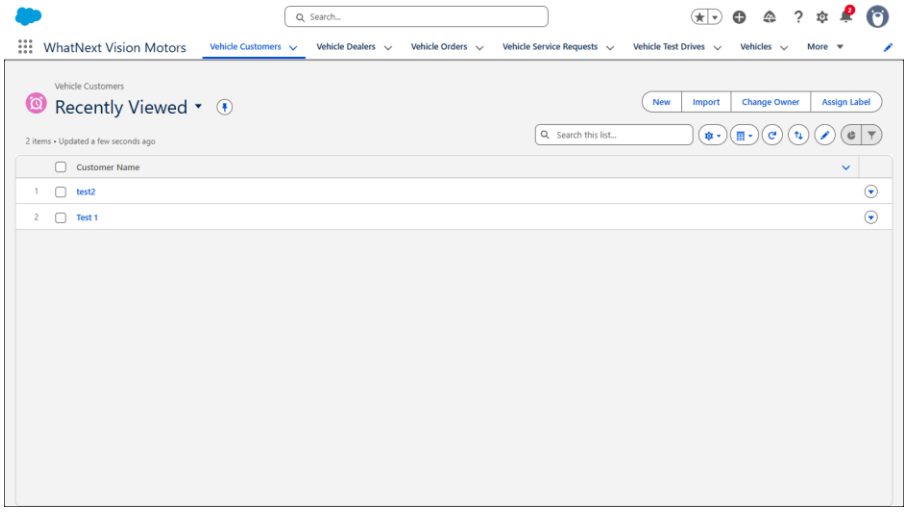


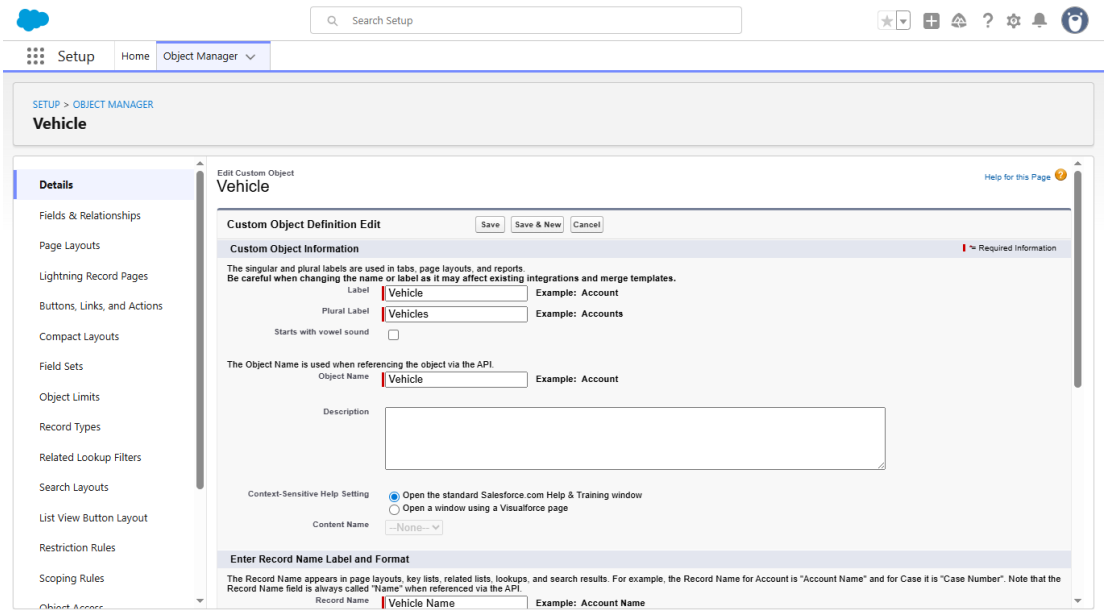
Fig: Org Page

Milestone 2: Objects Creation:

Activity 1: Creating Vehicle Object

To create an object:

1. From the setup page
2. Click on Object Manager
3. Click on Create >> Click on Custom Object.
4. Enter the label name as Vehicle
5. Enter Plural label name as Vehicles
6. Enter Record Name as Vehicle Name
7. Select Data Type as Text: Vehicle Name.
8. Select Allow reports.
9. Select Allow search**.**
10. Allow Track Field History
11. Click on Save and New



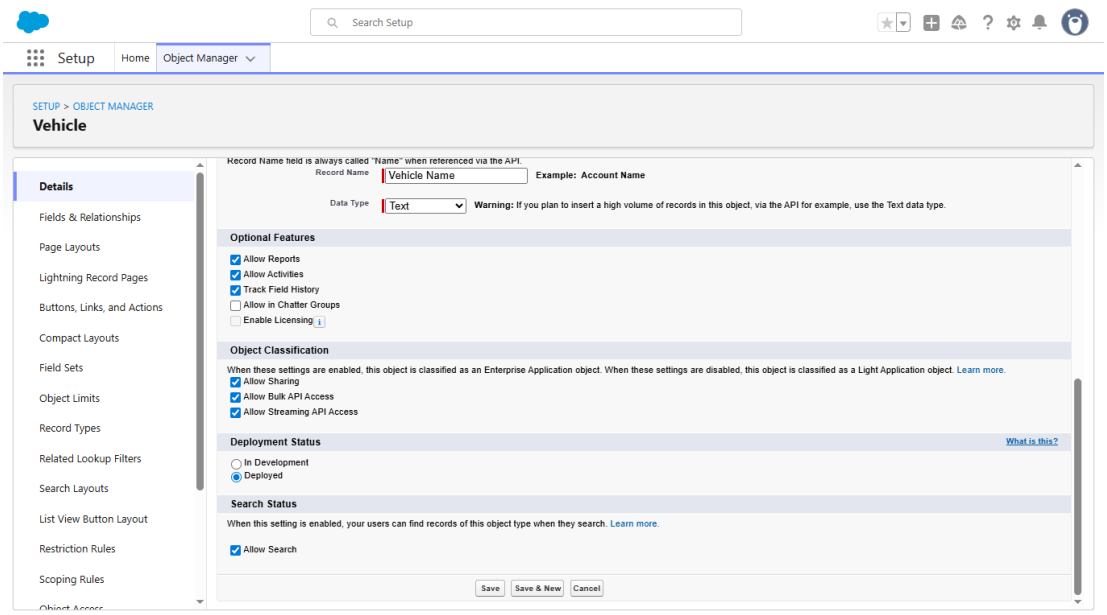


Fig: Vehicle Object

Activity 2: Creating Dealer Object

To create an object:

1. From the setup page
2. Click on Object Manager
3. Click on Create → Custom Object
4. Enter the label name as Vehicle Dealer
5. Enter Plural label name as Vehicle Dealers
6. Enter Record Name as Dealer Name
7. Select Data Type as Text: Dealer Name
8. Select Allow Reports
9. Select Allow Search
10. Allow Track Field History
11. Click on Save and New

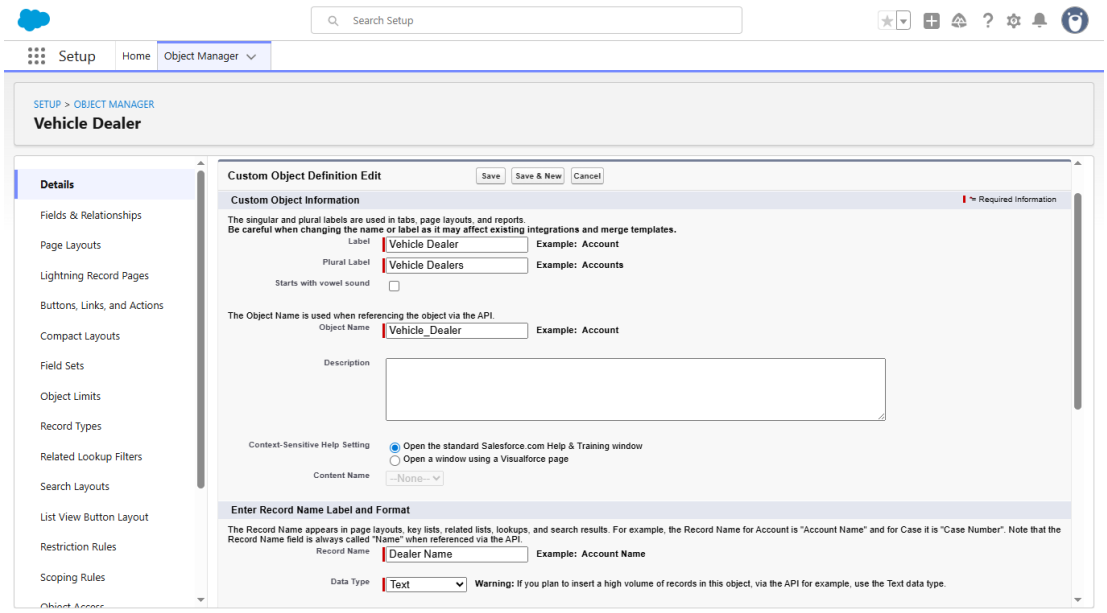


Fig: Dealer Object

Activity 3: Creating Customer Object

To create an object:

1. From the setup page
2. Click on Object Manager
3. Click on Create → Custom Object
4. Enter the label name as Vehicle Customer
5. Enter Plural label name as Vehicle Customers
6. Enter Record Name as Customer Name
7. Select Data Type as Text: Customer Name
8. Select Allow Reports
9. Select Allow Search
10. Allow Track Field History
11. Click on Save and New

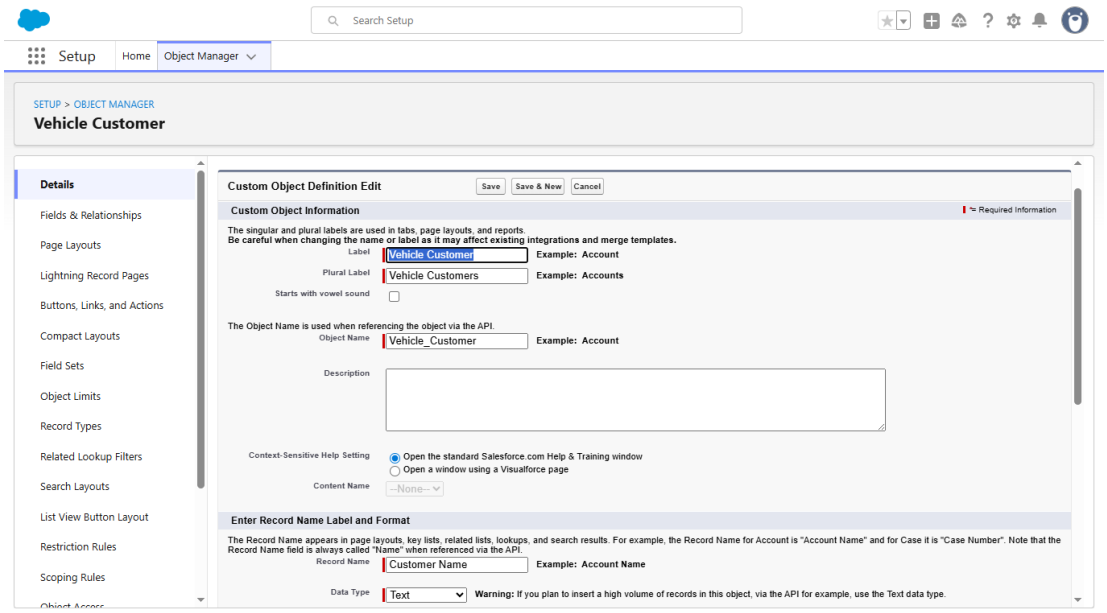


Fig: Customer Object

Activity 4: Creating Order Object

To create an object:

1. From the setup page
2. Click on Object Manager
3. Click on Create → Custom Object
4. Enter the label name as Vehicle Order
5. Enter Plural label name as Vehicle Orders
6. Enter Record Name as Order ID
7. Select Data Type as Auto Number: {0000}, starting from 1
8. Select Allow Reports
9. Select Allow Search
10. Allow Track Field History
11. Click on Save and New

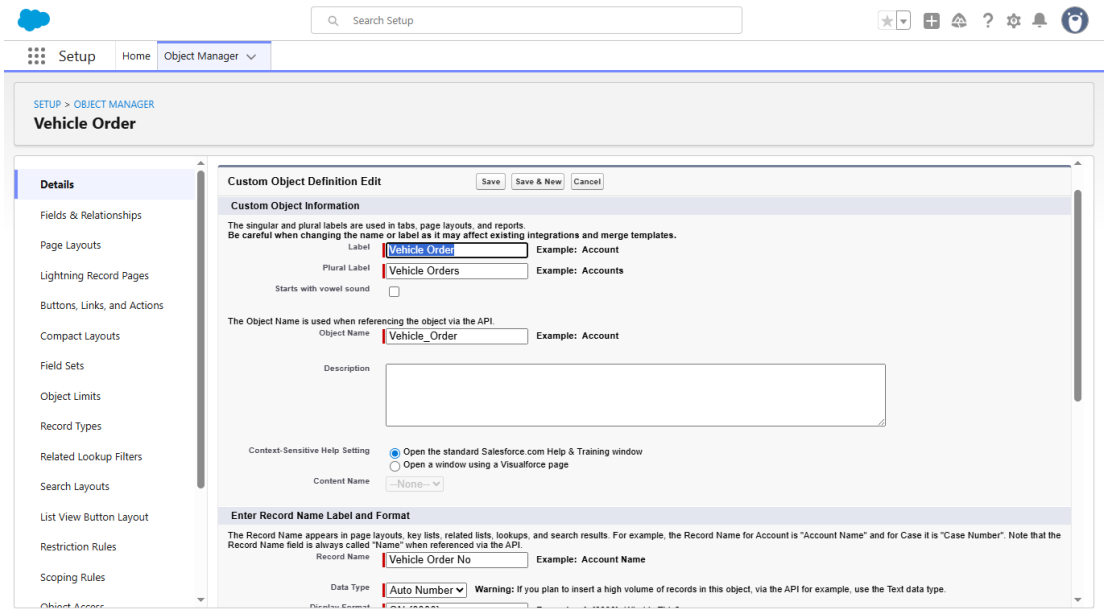


Fig: Order Object

Activity 5: Creating Test Drive Object

To create an object:

1. From the setup page
2. Click on Object Manager
3. Click on Create → Custom Object
4. Enter the label name as Vehicle Test Drive
5. Enter Plural label name as Vehicle Test Drives
6. Enter Record Name as Test Drive ID
7. Select Data Type as Auto Number: {0000}, starting from 1
8. Select Allow Reports
9. Select Allow Search
10. Allow Track Field History
11. Click on Save and New

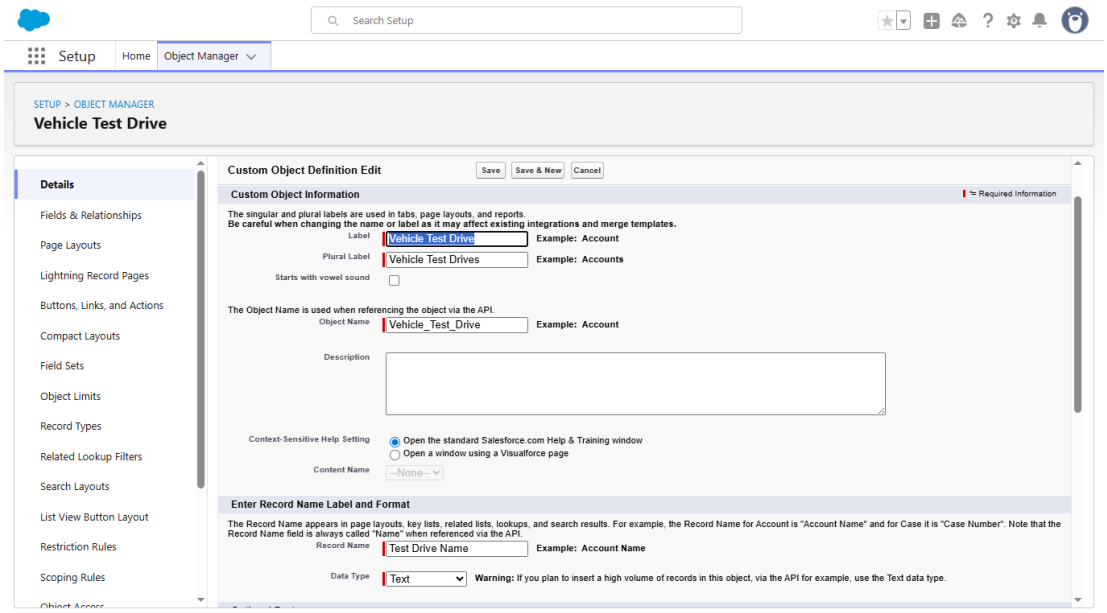


Fig: Test Drive Object

Activity 6: Creating Service Request Object

To create an object:

1. From the setup page
2. Click on Object Manager
3. Click on Create → Custom Object
4. Enter the label name as Vehicle Service Request
5. Enter Plural label name as Vehicle Service Requests
6. Enter Record Name as Service Request ID
7. Select Data Type as Auto Number: {0000}, starting from 1
8. Select Allow Reports
9. Select Allow Search
10. Allow Track Field History
11. Click on Save

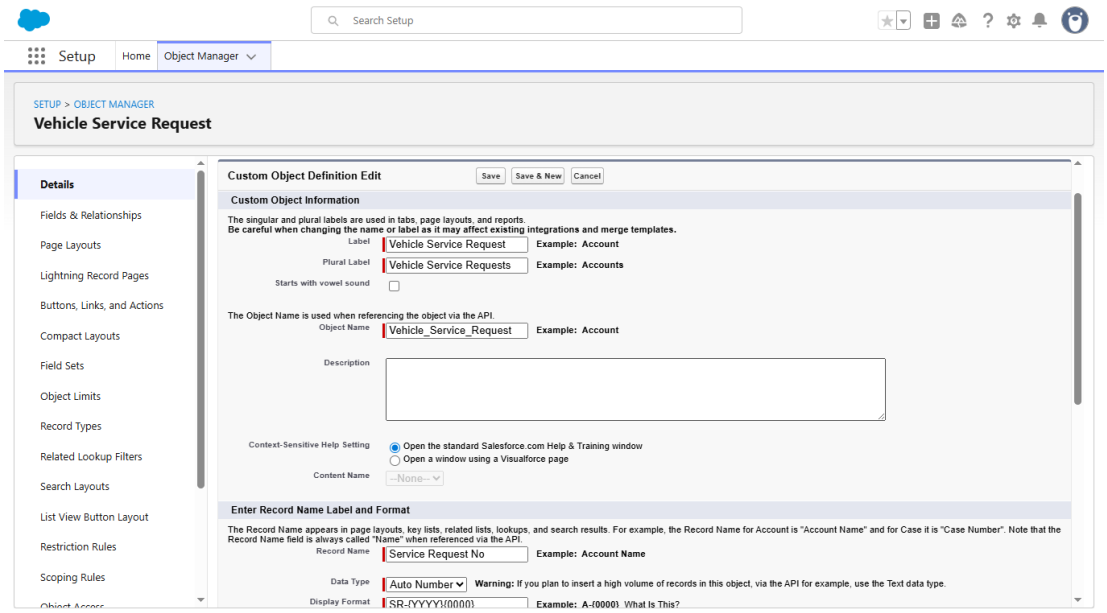


Fig: Service Request Object

Milestone 3- Tabs

Activity: Creating Tabs for All Objects

Follow the steps below to create tabs for every custom object in the project:

1. Go to the Setup page → type Tabs in the Quick Find bar
2. Click on Tabs
3. Click New (under Custom Object Tabs)
4. Select the object → Choose a Tab Style
5. Click Next → (Add to Profiles page) keep it as default
6. Click Next → (Add to Custom App) uncheck the Include Tab option
7. Make sure Append tab to the user's existing personal customizations is checked
8. Click Save
9. Repeat the same steps for each object created in Milestone-2.

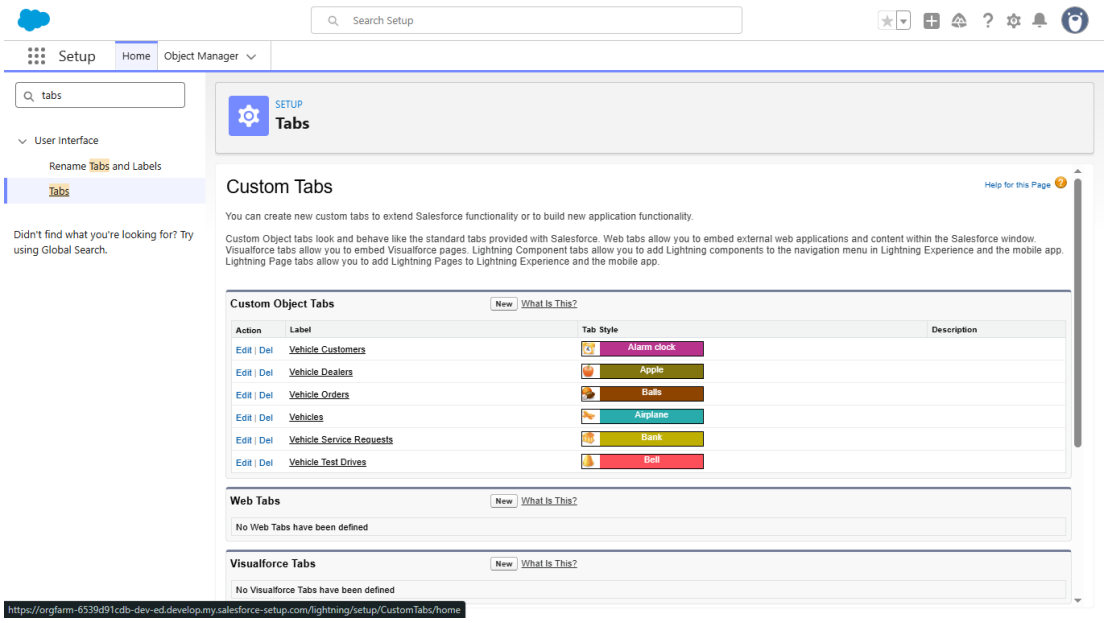


Fig: Tabs

Milestone 4: Fields & Relationships

Object Name	Field Name	Data Type	Notes / Relationship
Vehicle__c	Vehicle_Name__c	Text	Record Name
	Vehicle_Model__c	Picklist (Sedan, SUV, EV, etc.)	Model Type
	Stock_Quantity__c	Number	Stock Available
	Price__c	Currency	Vehicle Price
	Dealer__c	Lookup (Vehicle_Dealer__c)	Assigned Dealer

	Status__c	Picklist (Available, Out of Stock, Discontinued)	Vehicle Availability
Vehicle_Dealer__c	Dealer_Name__c	Text	Record Name
	Dealer_Location__c	Text	Dealer Address
	Dealer_Code__c	Auto Number	Unique Dealer ID
	Phone__c	Phone	Dealer Contact
	Email__c	Email	Dealer Email
Vehicle_Order__c	Order ID	Auto Number	Record Name
	Customer__c	Lookup (Vehicle_Customer__c)	Ordered By Customer
	Vehicle__c	Lookup (Vehicle__c)	Ordered Vehicle
	Order_Date__c	Date	Order Date
	Status__c	Picklist (Pending, Confirmed, Delivered, Canceled)	Order Status
Vehicle_Customer__c	Customer_Name__c	Text	Record Name
	Email__c	Email	Customer

			Email
	Phone__c	Phone	Customer Phone
	Address__c	Text	Customer Address
	Preferred_Vehicle_Type__c	Picklist (Sedan, SUV, EV, etc.)	Vehicle Preference
Vehicle_Test_Drive__c	Test Drive ID	Auto Number	Record Name
	Customer__c	Lookup (Vehicle_Customer__c)	Test Drive Customer
	Vehicle__c	Lookup (Vehicle__c)	Test Drive Vehicle
	Test_Drive_Date__c	Date	Scheduled Date
	Status__c	Picklist (Scheduled, Completed, Canceled)	Test Drive Status
Vehicle_Service_Request__c	Service Request ID	Auto Number	Record Name
	Customer__c	Lookup (Vehicle_Customer__c)	Service Requested By
	Vehicle__c	Lookup (Vehicle__c)	Vehicle Under Service
	Service_Date__c	Date	Service

			Requested Date
	Issue_Description__c	Text	Problem Details
	Status__c	Picklist (Requested, In Progress, Completed)	Service Status

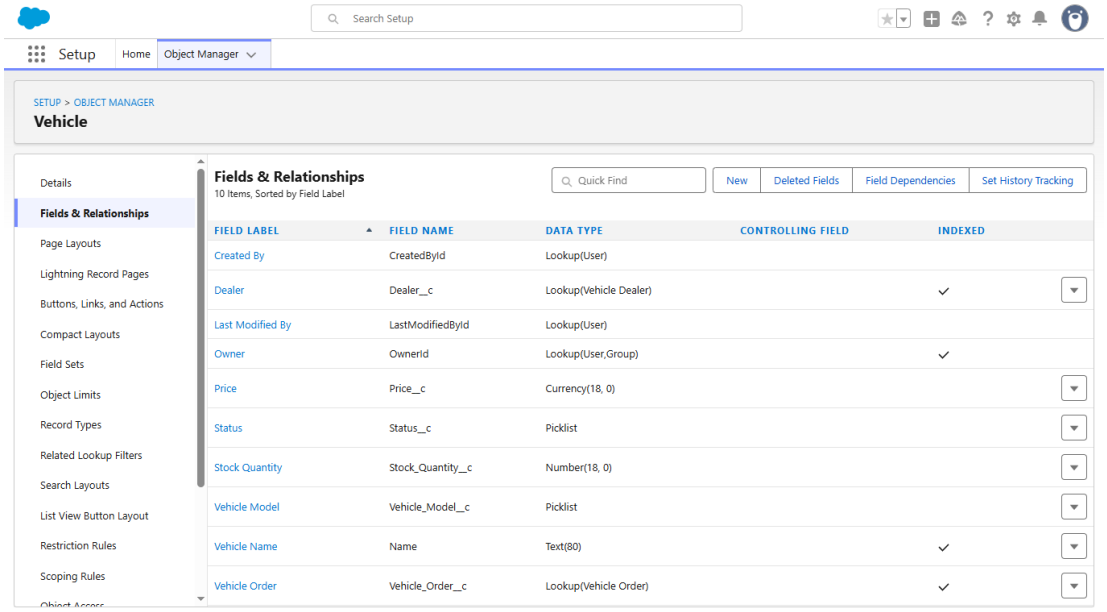


Fig: Vehical

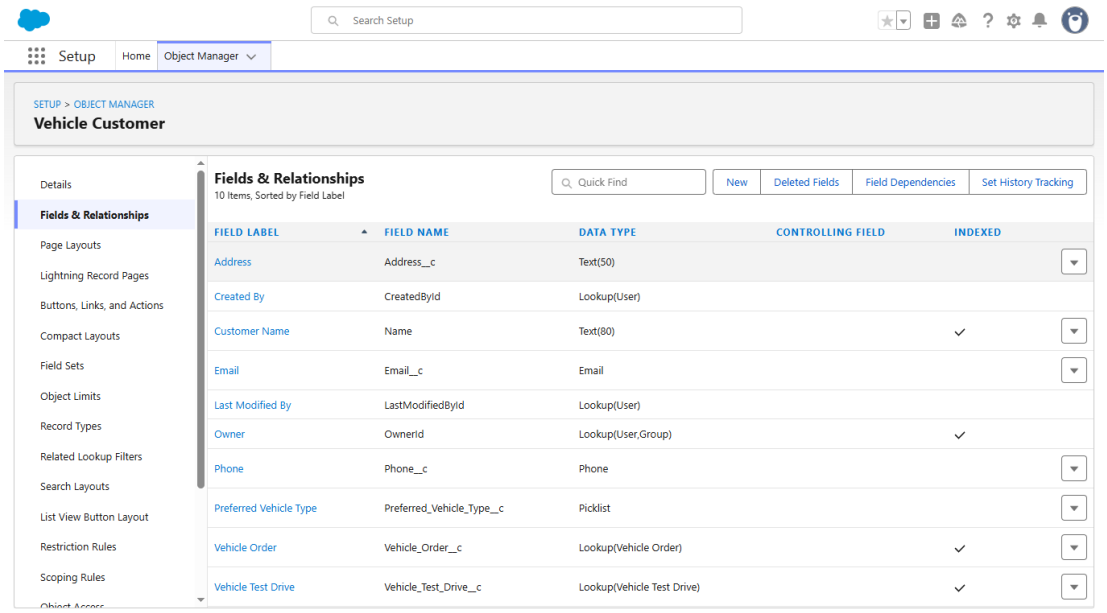


Fig: Vehicle Customer

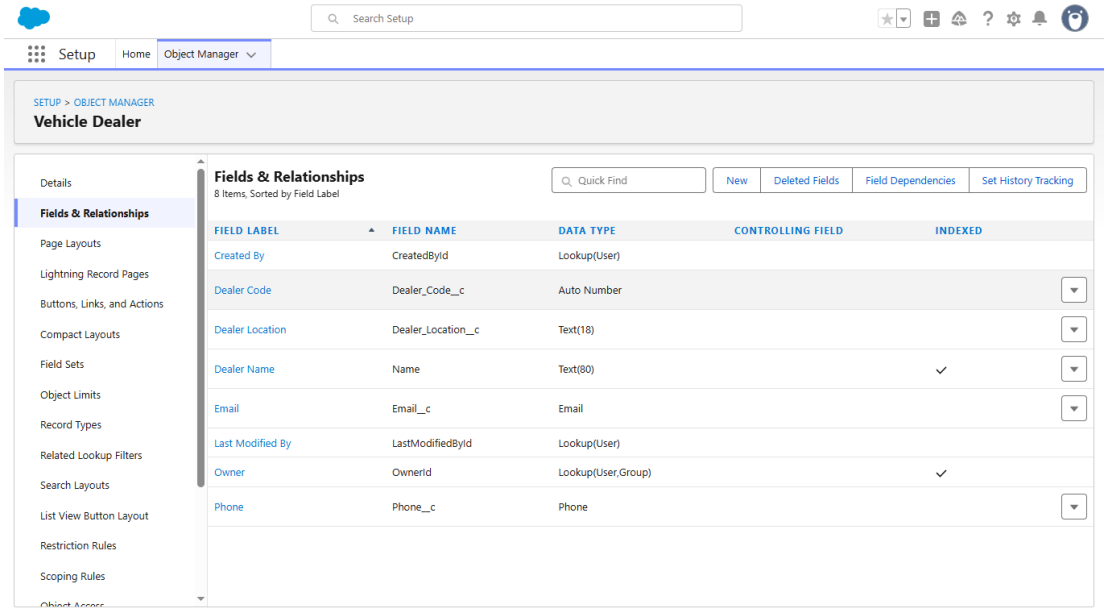


Fig: Vehicle Dealer

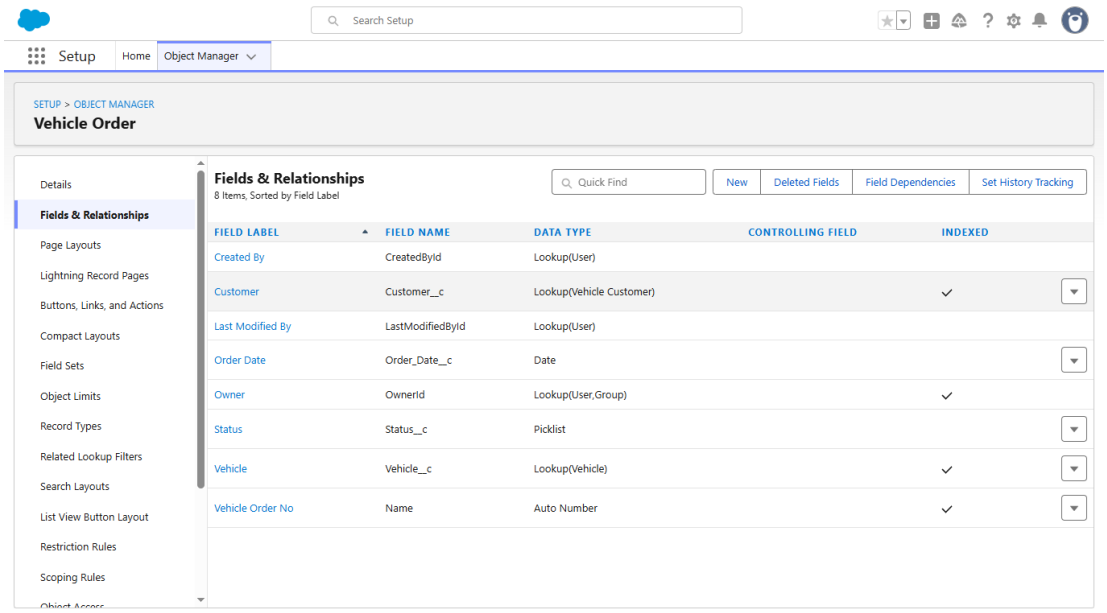


Fig: Vehicle Order

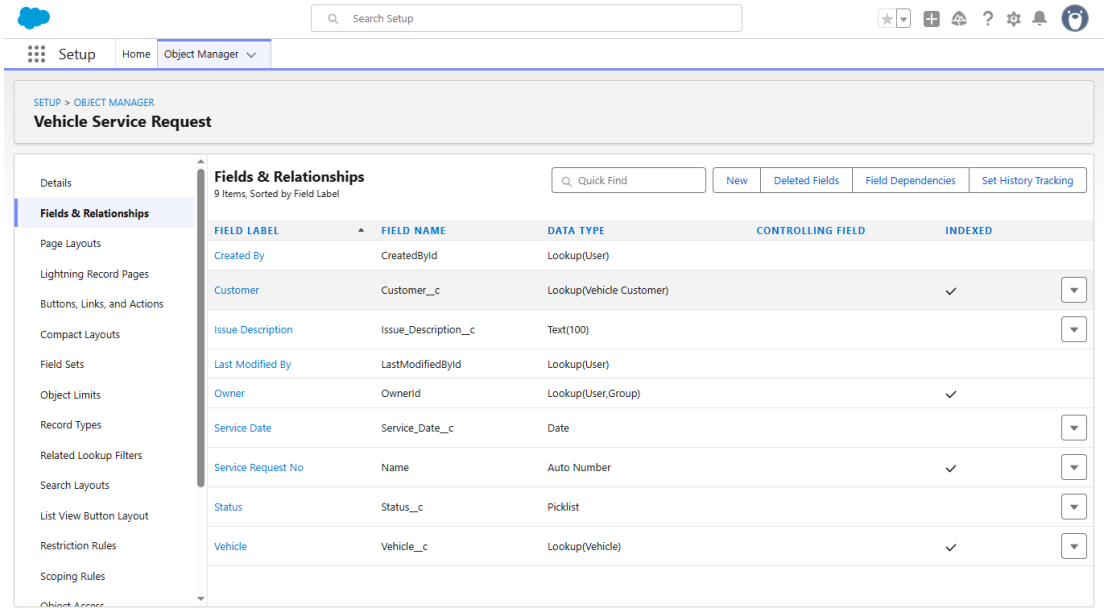


Fig: Service Request

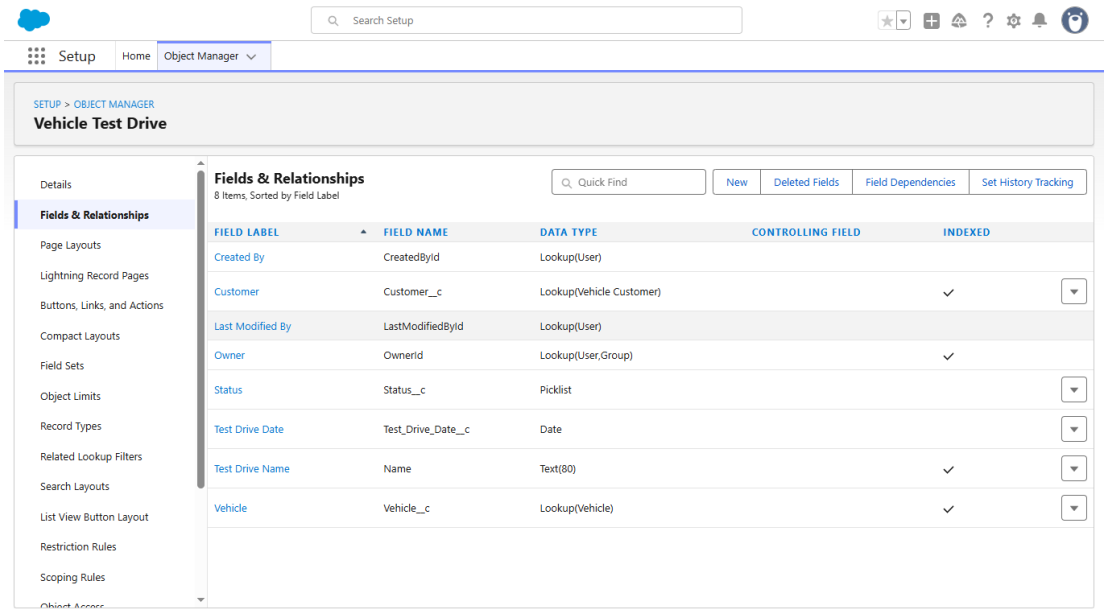


Fig: Vehicle Test Drive

Milestone 5 – Create a Lightning App :- WhatNext Vision Motors

1) Create the App

- Go to Setup → App Manager → New Lightning App
- Enter App Name: WhatNext Vision Motors
- Add Description (image optional)
- Click Next → Next → Next (keep all default settings)

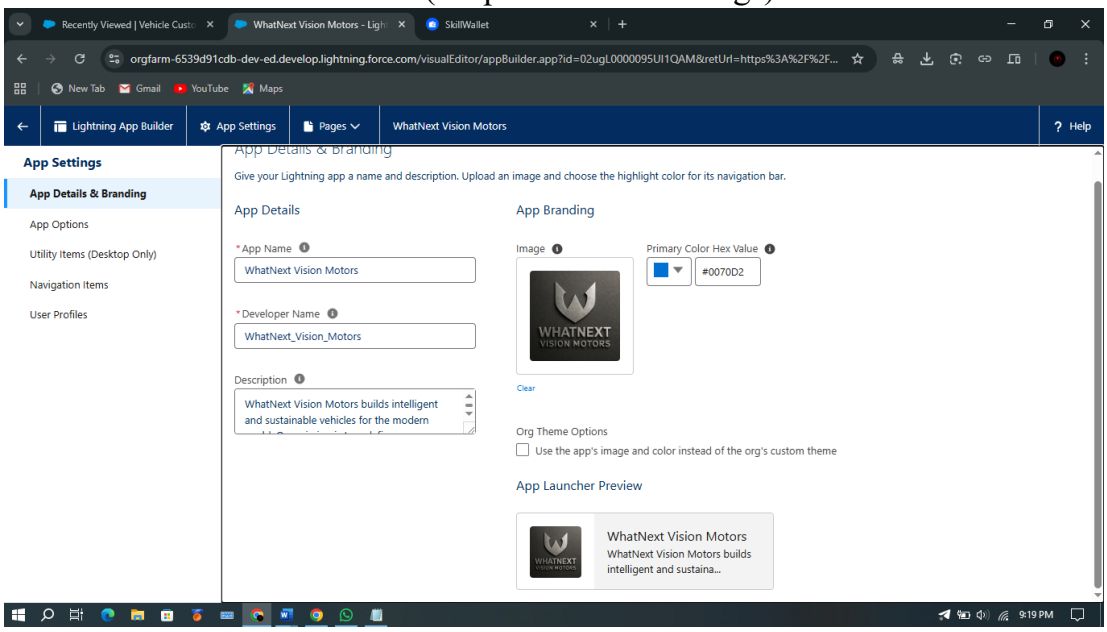


Fig: App Details

2) Add Navigation Items

- Search and add the following:
- Vehicle, Vehicle Dealer, Vehicle Customer, Vehicle Order, Vehicle Test Drive, Vehicle Service Request, Reports, Dashboards
- Click Next

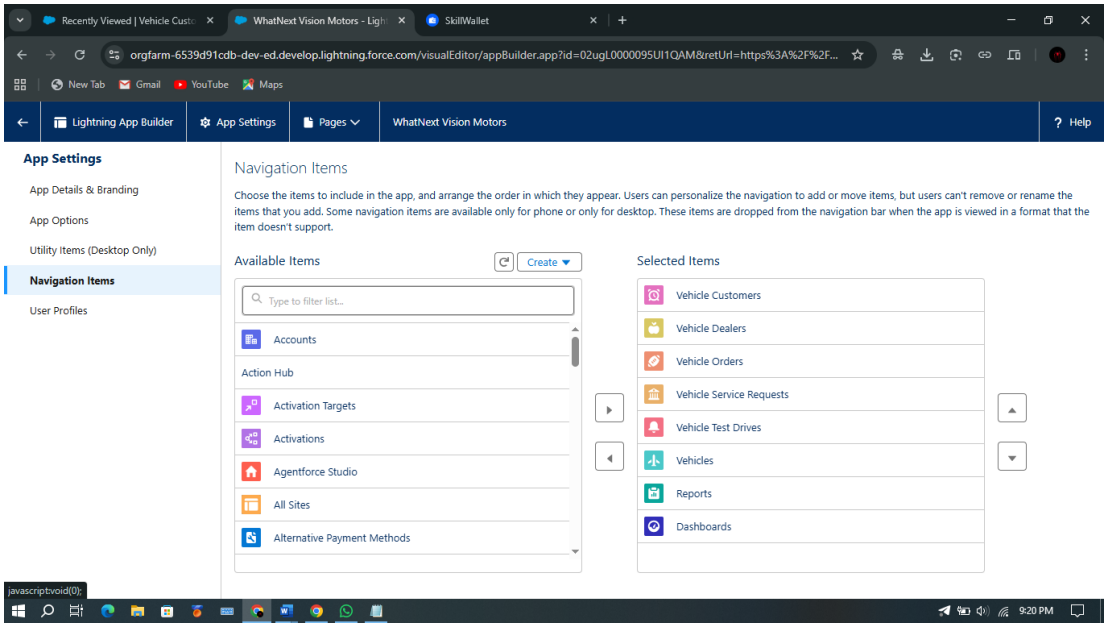


Fig: Navigation Item

3) Assign User Profiles

- Search System Administrator
- Move it to Selected Profiles
- Click Save & Fin

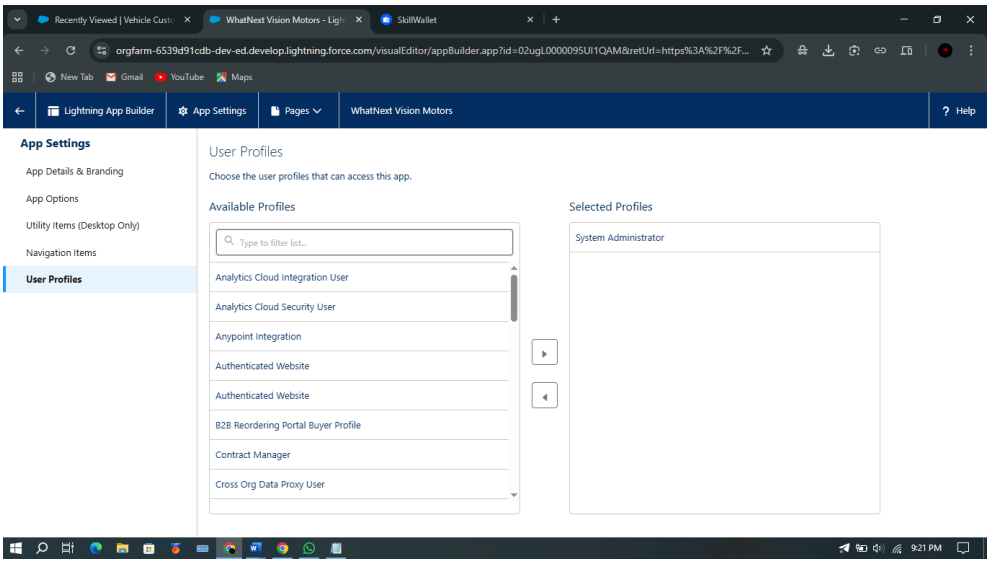


Fig: User Profile

Milestone 6 – Record-Triggered Flow to Auto-Assign Nearest Dealer

1)Create the Flow

Go to Setup → Quick Find → Flows → New Flow
Select Start from Scratch → Next
Choose Record-Triggered Flow → Create

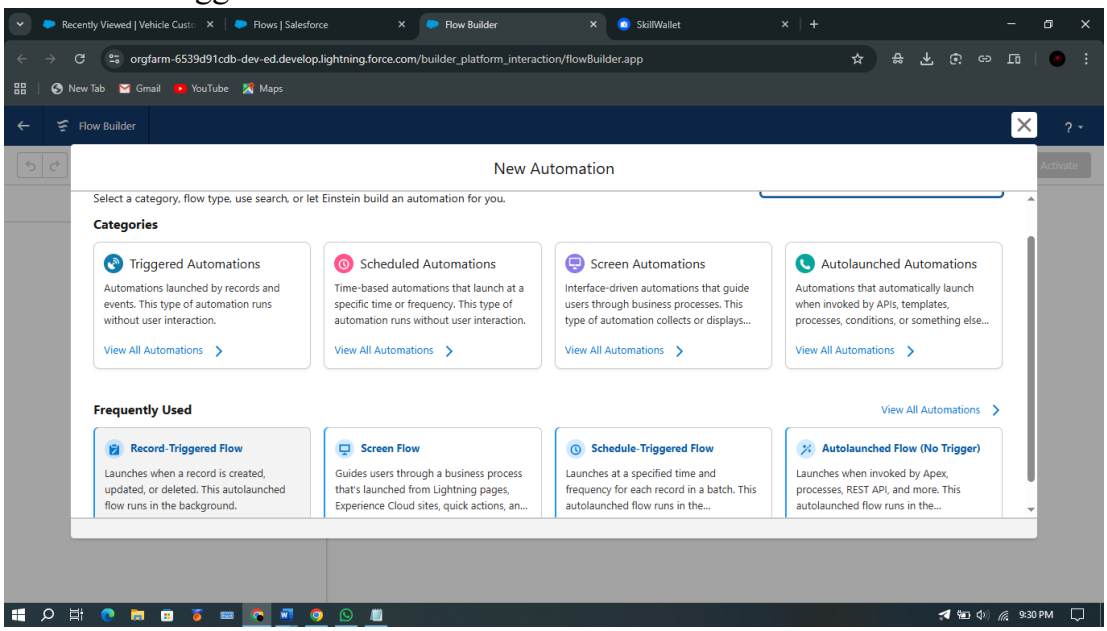


Fig: Record Trigger Flow

2)Configure Trigger

Object: Vehicle Order
Trigger When a record is created
Entry Condition:
Status__c = Pending
Condition Logic: All Conditions Are Met (AND)

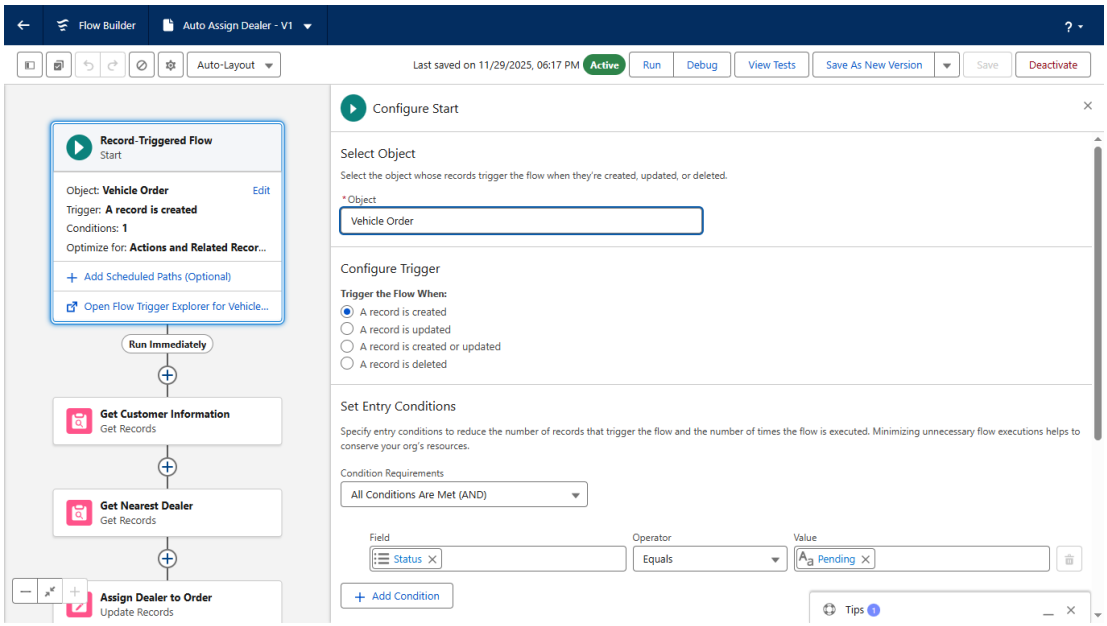


Fig: Vehicle Order Trigger

3)Get Customer Details

+ → Get Records

Label: Get Customer Information

Object: Vehicle Customer

Filter: Id = {!$Record.Vehicle_Customer__c}

Store First Record and Automatically store all fields

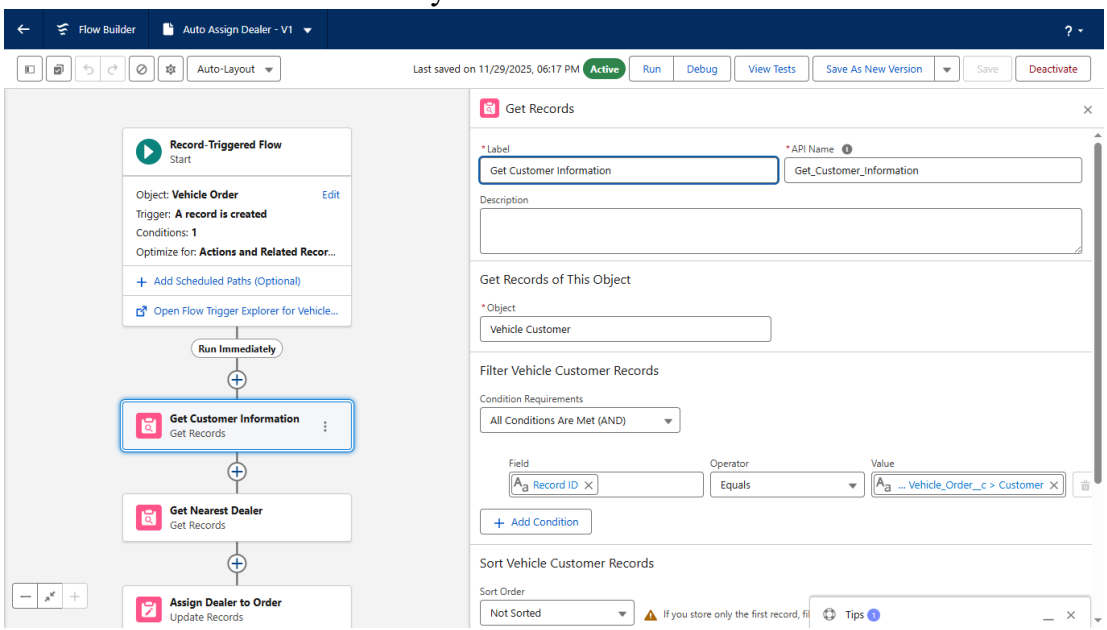


Fig: Get Record Vehicle Customer

4)Get Nearest Dealer

+ → Get Records
Label: Get Nearest Dealer
Object: Vehicle Dealer
Filter: Dealer_Location__c = {!Get_Customer_Information.Address__c}
Store First Record and Automatically store all fields

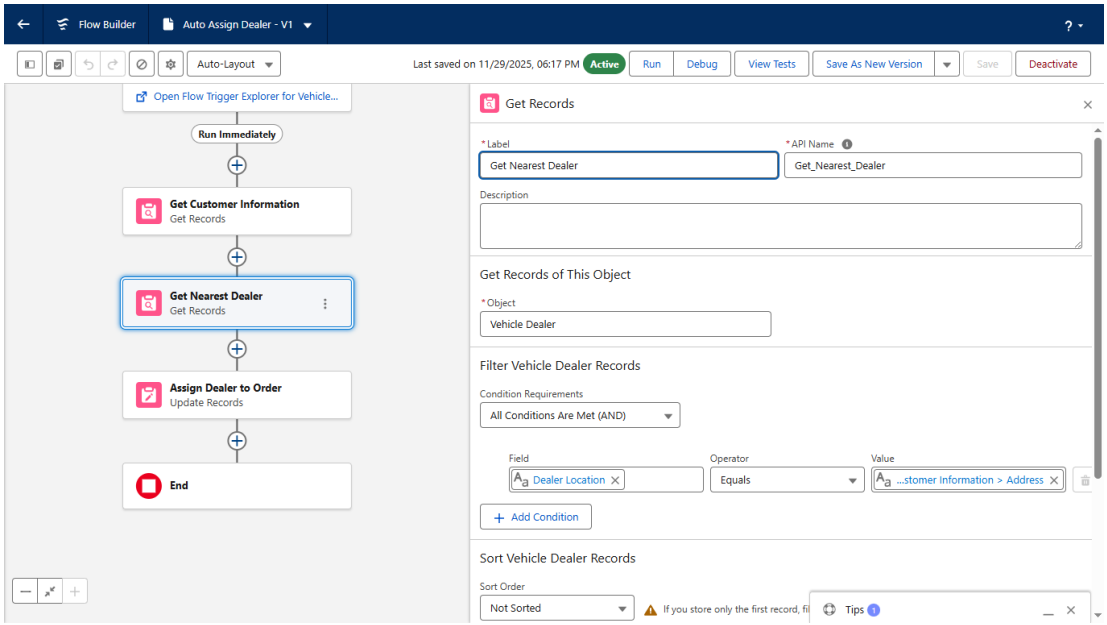


Fig: Get Record Vehicle Dealer

5)Assign Dealer to Order

+ → Update Records
Label: Assign Dealer to Order
How to Find Records to Update:
Use the IDs and all field values from a record
Record to Update: {!Get_Nearest_Dealer}

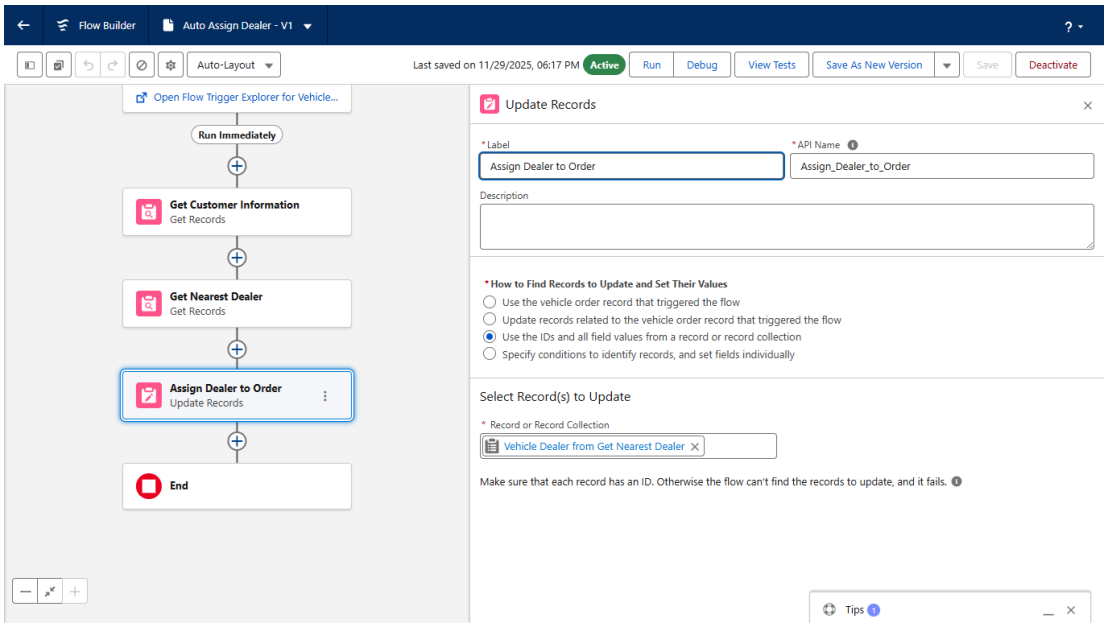


Fig: Update Record

6)Save & Activate

Name the Flow: Auto Assign Dealer

Click Activate

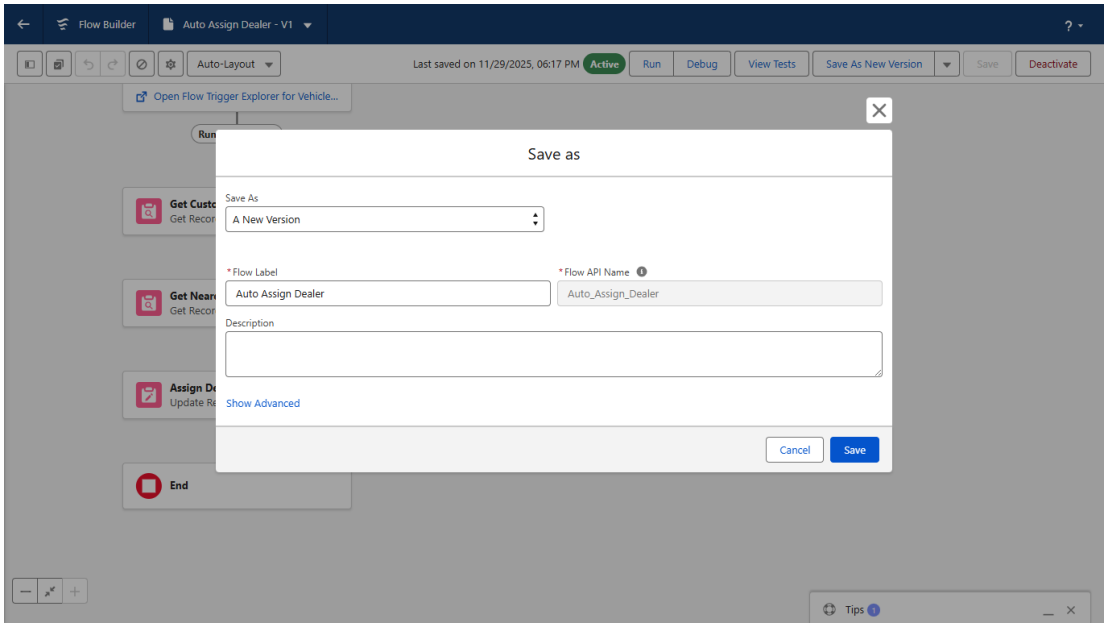


Fig: Auto Assign Dealer

Milestone 6 (Flow – 2) – Record-Triggered Flow to Send Test Drive Reminder Email

1) Create the Flow

Go to Setup → Flows → New Flow
Select Record-Triggered Flow → Create

2) Configure Trigger

Object: Vehicle Test Drive
Trigger the Flow When a record is created or updated
Entry Condition:
Status__c = Scheduled
Condition Logic: All Conditions Are Met (AND)

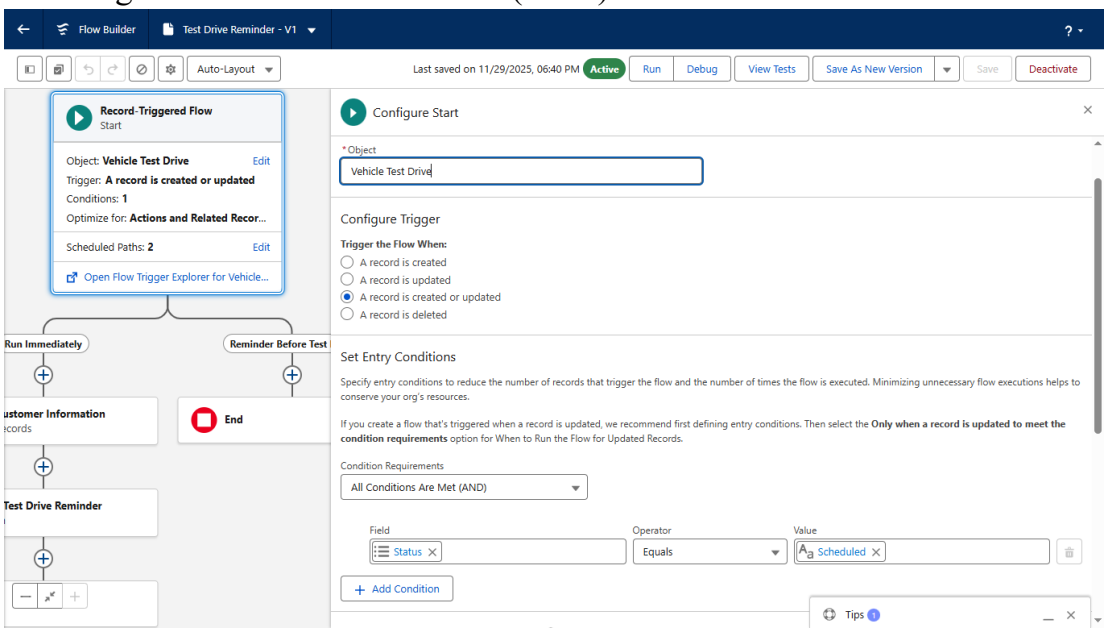


Fig: Record Trigger Test Drive

3) Create Scheduled Path

Click + Add Scheduled Paths (below the trigger)
Label: Reminder Before Test Drive
Time Source: Test_Drive_Date__c
Offset Number: 1
Offset Option: Days Before
Click Done

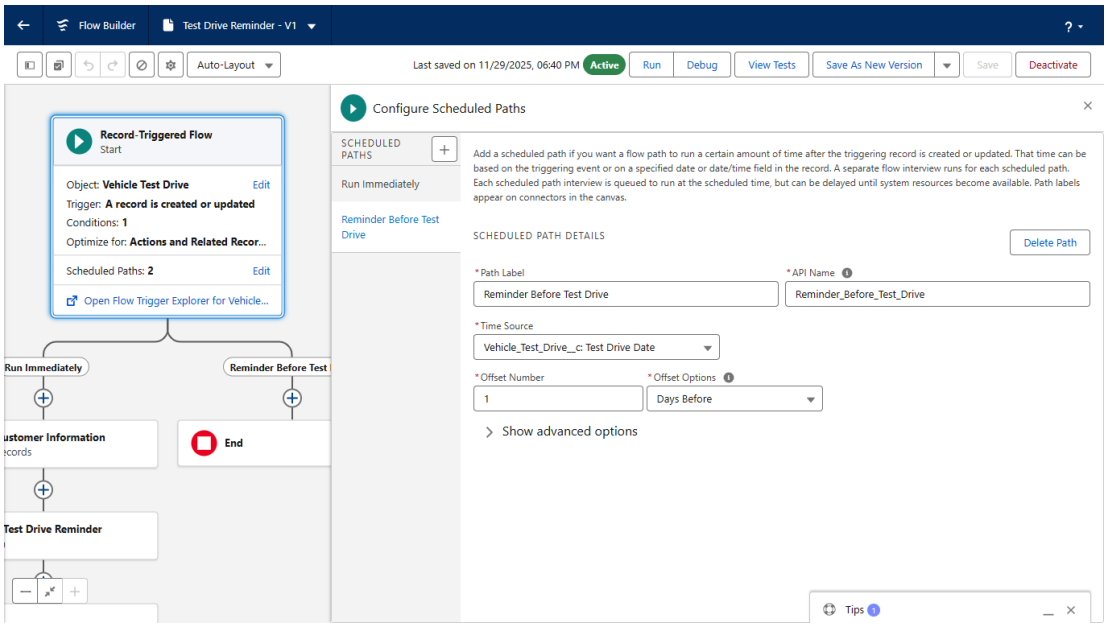


Fig: Scheduled Paths

4) Get Customer Details

+ → Get Records

Label: Get Customer Information

Object: Vehicle_Customer__c

Filter: Id = {!$Record.Customer__c}

Store Only the first record and Automatically store all fields

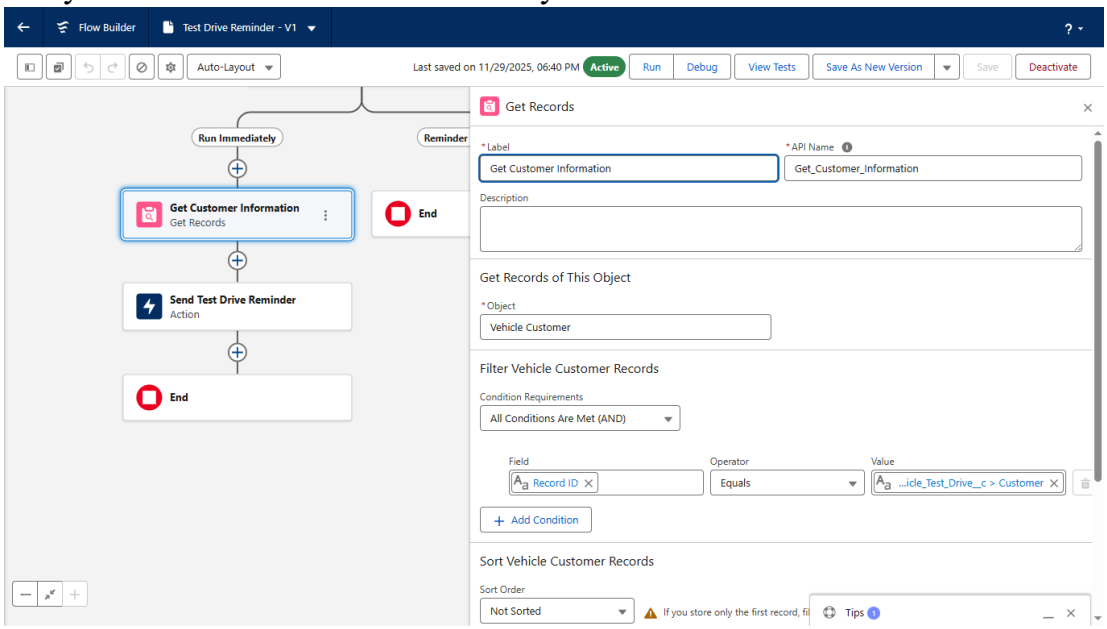


Fig: Get Record

5) Send Reminder Email

+ → Action
Action Type: Send Email
Label: Send Test Drive Reminder
Subject: "Reminder: Your Test Drive is Tomorrow!"
Recipient Address: {!Get_Customer_Information.Email__c}
Rich Text Body: Enabled
Body Variable API Name: EmailSent

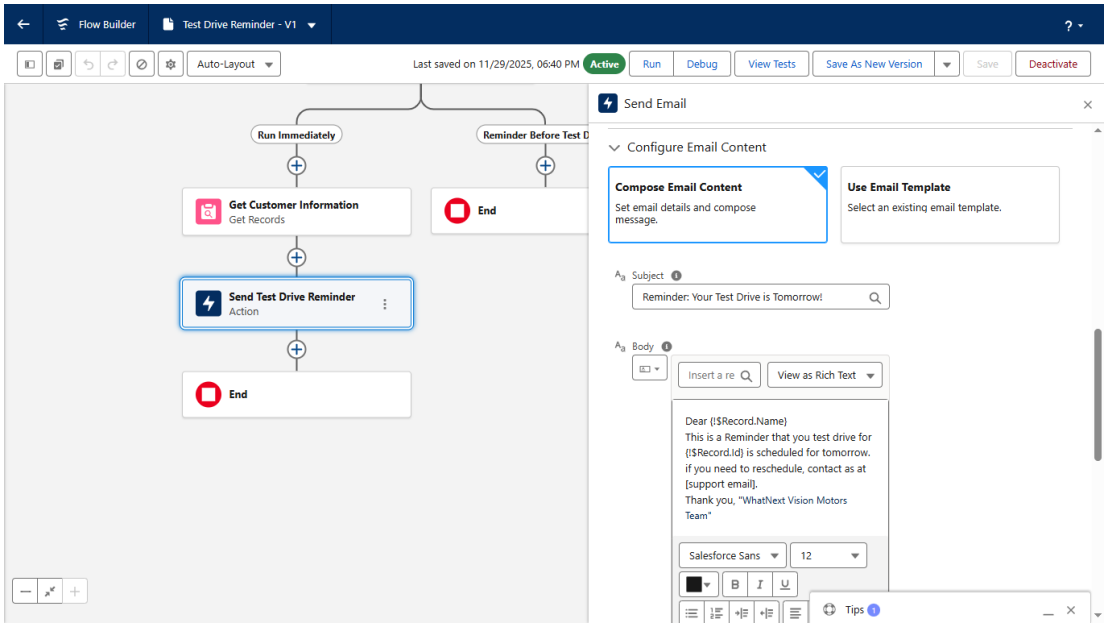


Fig: Send Email

6) Save & Activate

Label Name: Test Drive Reminder
Click Activate

Milestone 7 – Apex Trigger, Batch Job & Scheduler with Full Code

Step 1: Create Apex Trigger Handler Class

Class Name: 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);
        }
    }
}
// Method to prevent orders when the vehicle is out of stock
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.');
                }
            }
        }
    }
}
// Method to update vehicle stock when an order is 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;
            }
        }
    }
}
```

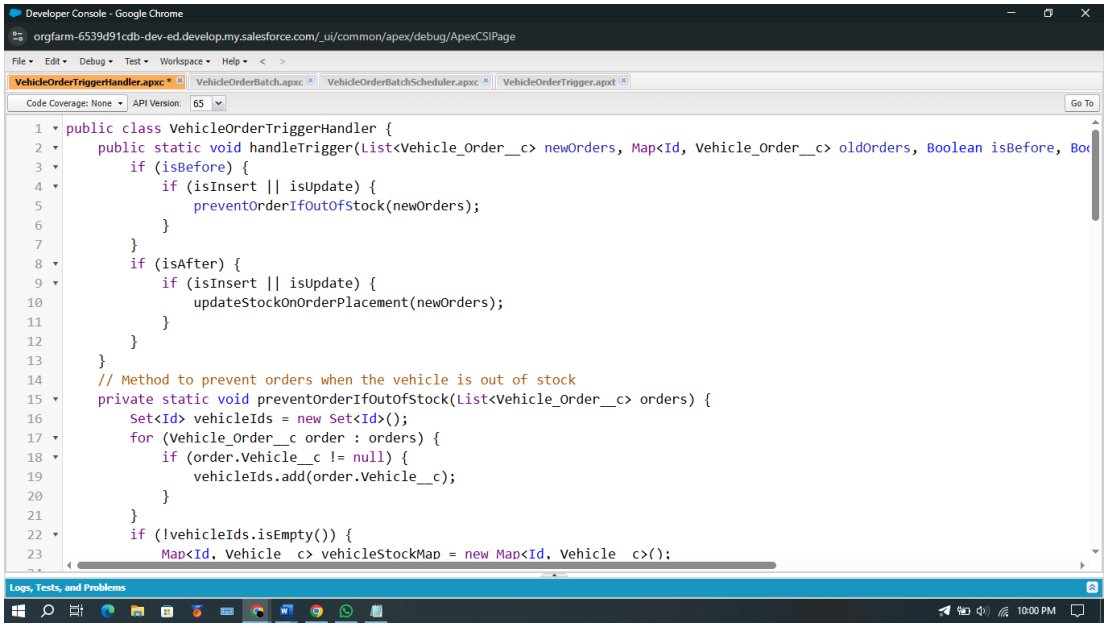


Fig: VehicleOrderTriggerHandler

Step 2: Create Trigger on Vehicle Order Object

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);

}

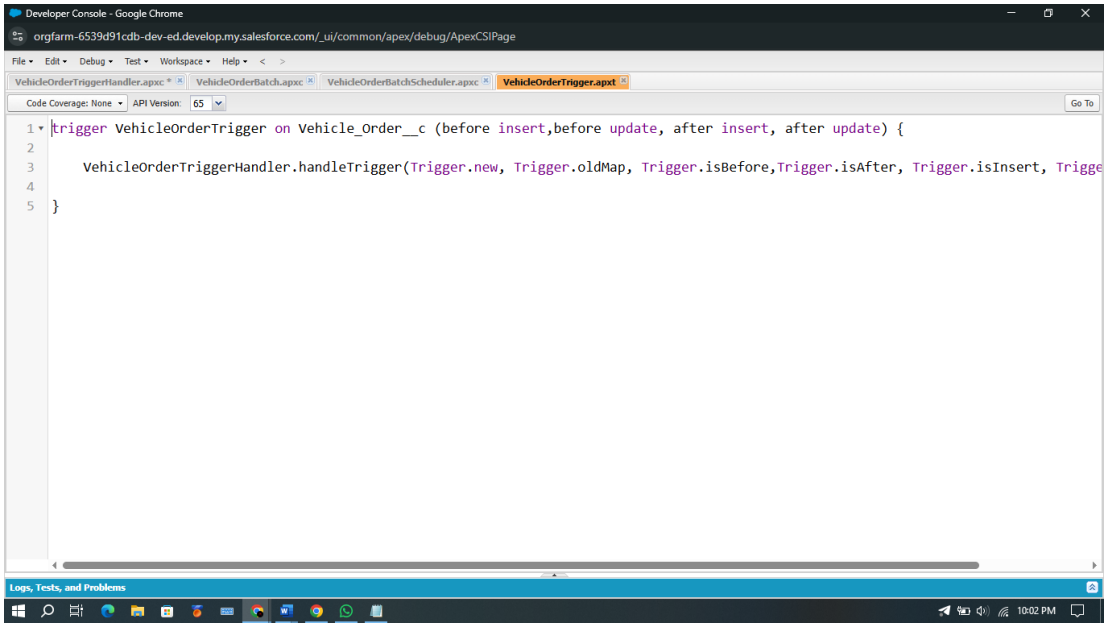
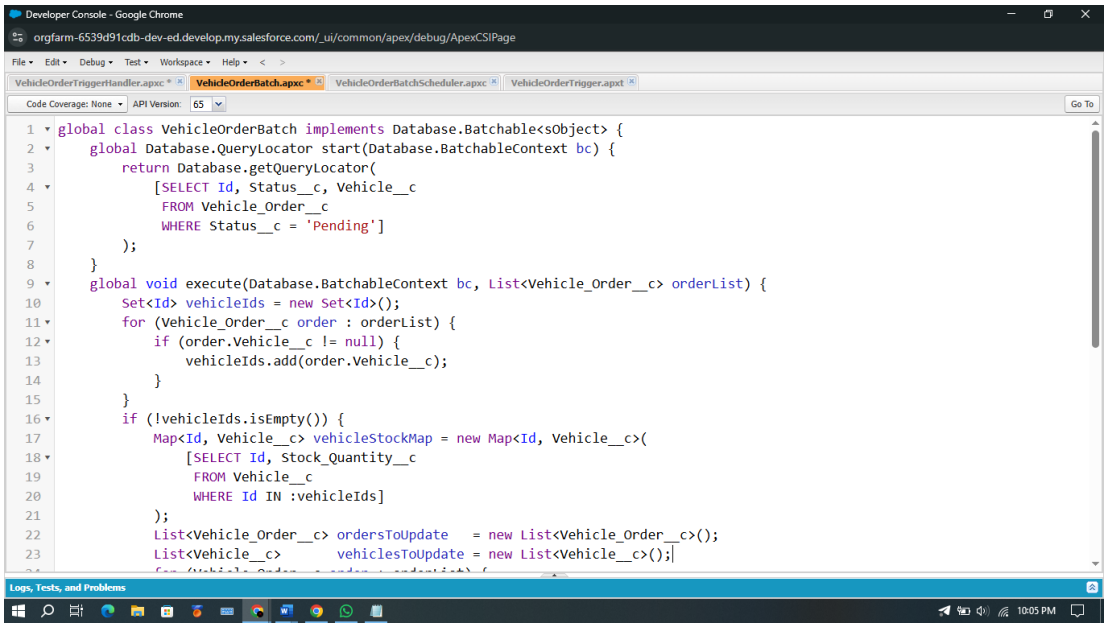


Fig: VehicleOrderTrigger

Step 3: Create Batch Class to Auto-Confirm Pending Orders When Stock is Available

```
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.');
  }
}
```

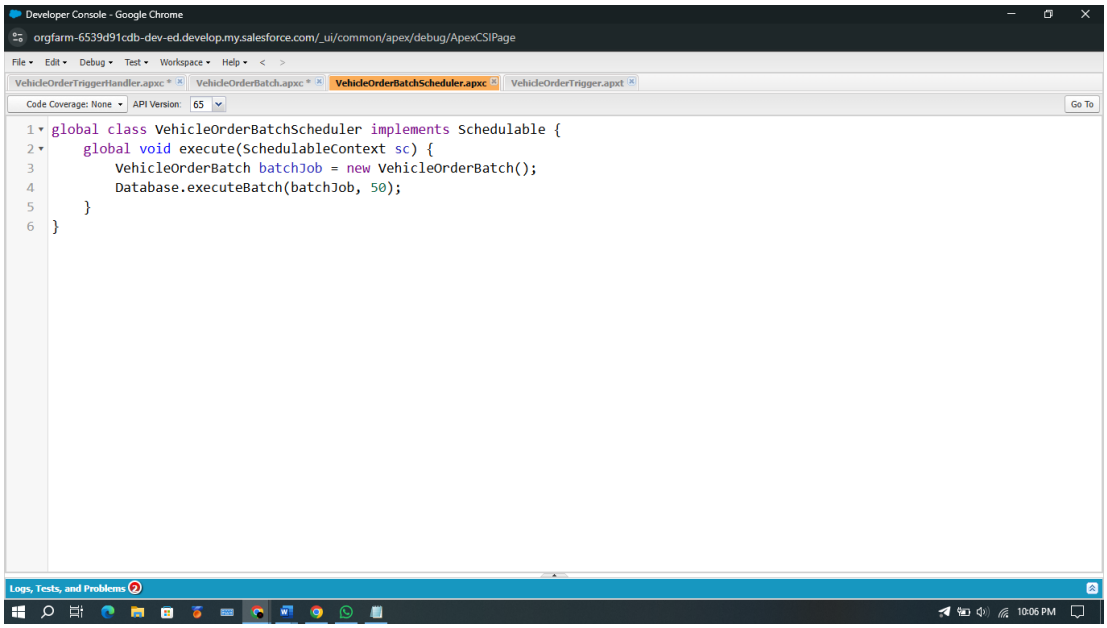



```
global class VehicleOrderBatch implements Database.Batchable<sObject> {
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator(
            [SELECT Id, Status__c, Vehicle__c
             FROM Vehicle_Order__c
             WHERE Status__c = 'Pending']
        );
    }
    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>();
```

Fig: VehicleOrderBatch

Step 4: Scheduler Class to Run Batch Automatically

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




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

Fig: VehicleOrderBatchScheduler

Step 5: Schedule the Batch Job

String cronExp = '0 0 12 * * ?'; // Runs daily at 12:00 PM
System.schedule('Daily Vehicle Order Processing', cronExp, new VehicleOrderBatchScheduler());

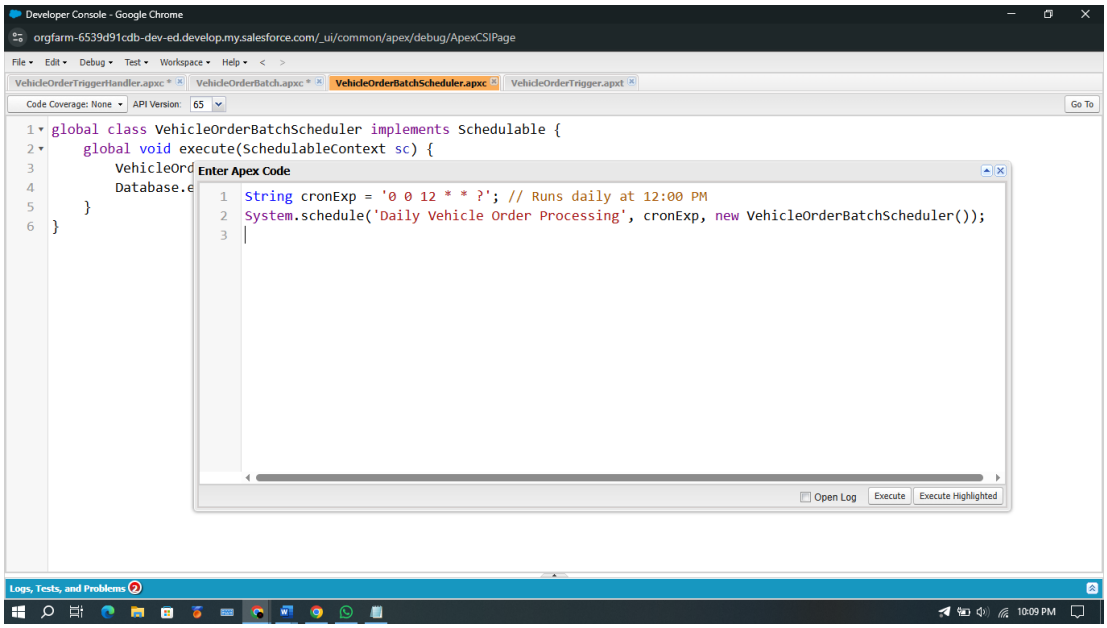


Fig: Schedule

Milestone 8 Output:

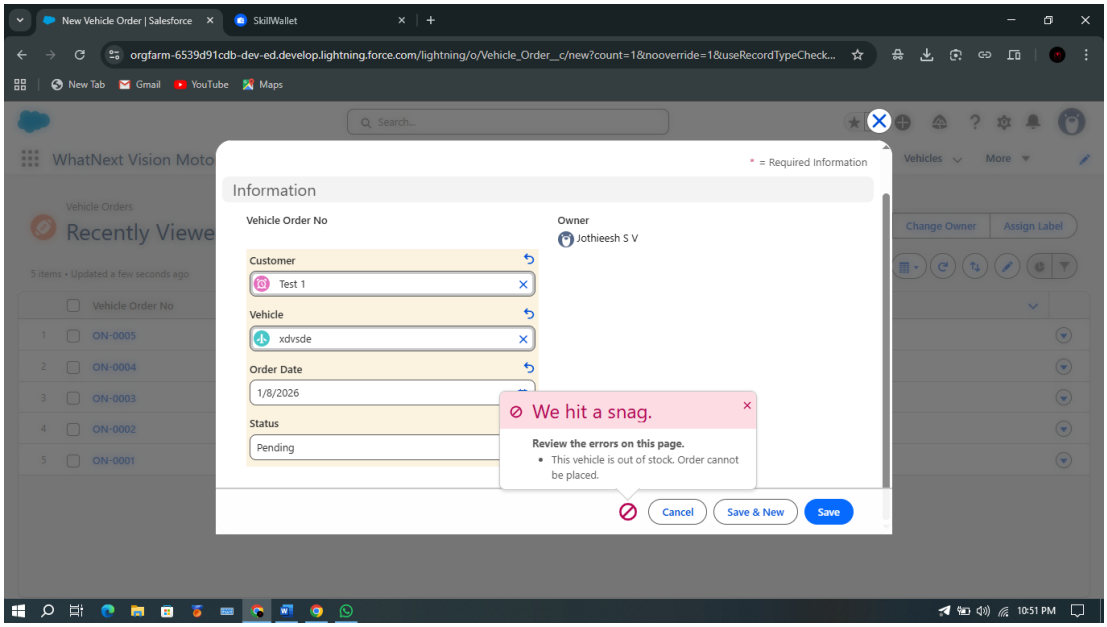


Fig: Output

Phase 3: UI/UX Development & Customization

UI/UX (User Interface & User Experience) plays a crucial role in making a CRM system practical, user-friendly, and efficient for day-to-day operations. In the WhatNext Vision Motors CRM Project, the interface was designed with a strong focus on simplicity, speed, and smooth navigation to ensure that users such as sales executives, dealers, and customer managers can interact with the system with minimal effort.

This phase involves designing a visually appealing and workflow-aligned environment using Salesforce Lightning components, Lightning App Builder, page layouts, and object tabs. The goal was to create a seamless end-to-end experience for handling vehicle listings, customer lookup, dealer assignment, customer orders, test drive bookings, and service request tracking

Creation of Lightning App – "WhatNext Vision Motors CRM"

A custom Lightning App named "WhatNext Vision Motors CRM" was created to provide a centralized entry point for all modules. The app includes branding, navigation items, utilities, and profile access criteria.

The Lightning App contains navigation items representing all major object modules:

- Vehicles
- Vehicle Dealers
- Vehicle Customers
- Vehicle Orders
- Vehicle Test Drives
- Vehicle Service Requests
- Reports
- Dashboards

This ensures that users can access every CRM feature from a single application interface without searching through the App Launcher.

Navigation & User Flow

When a user opens the Lightning App, the CRM navigation bar clearly organizes the complete business process:

1. Viewing vehicles in stock
2. Viewing list of dealers and their locations
3. Accessing customer details
4. Managing orders and checking delivery status
5. Scheduling or tracking test drives
6. Recording and monitoring service requests

Each navigation item was placed in the most logical sequence to match an actual automotive customer-sales lifecycle. This enhances user workflow memory and reduces learning time for new users.

Page Layout Customization

Each object in the CRM was customized with page layouts suitable to the type of users accessing them.
Vehicle Page Layout

Designed for inventory managers and sales executives.
Key enhancements include:

- Vehicle Name, Model, Price, and Stock positioned at the top for immediate visibility
- Related list for Orders and Test Drives added for quick vehicle demand tracking
- Dynamic visibility for field Status__c based on stock quantity

Dealer Page Layout

Optimized for dealer lookup during order placement:

- Contact panel (Phone, Email)
- Dealer location field placed at top
- Related vehicles and orders linked to track performance and sales volume

Customer Page Layout

Designed for quick customer verification:

- Phone and Email highlighted at the top
- Order history and test drive history kept in related lists
- Address placed prominently for dealer assignment automation

Order Page Layout

Built to support fast order processing:

- Order Date and Order Status centered for clarity
- Vehicle and Customer lookup fields placed next to each other
- Custom notification banner to highlight when vehicle is out of stock

Test Drive & Service Request Layouts

- Calendar view fields added for scheduling and time management
- Status fields highlighted to support progress tracking (“Scheduled”, “Completed”, “Canceled”)

Designing with Salesforce Lightning Experience

To improve usability and aesthetics, the CRM interface includes:

- Card-style layout sections
- Icons and colors aligned with automotive brand theme
- Minimum scroll design
- Field grouping based on workflow logic

The Lightning framework ensures users complete tasks with fewer clicks — for example:

- From a vehicle record, users can directly open “Related Orders”
- From the customer page, users can create a “New Vehicle Order” with a single button click
- From order record, users can review stock details and dealer information instantly

Dynamic Forms & Conditional Visibility

Dynamic forms were implemented for:

- Hiding the “Order Confirmation” field until the vehicle is selected
- Showing “Out of Stock Alert” when vehicle stock is zero
- Showing "Dealer Lookup Field" only when order status = Pending
- Displaying service fields only when a customer has an existing service record

This conditional visibility reduces confusion and prevents data entry errors, making CRM usage much easier.

Tab Creation for CRM Objects

Tabs were created for all custom objects:

- Vehicle
- Vehicle Dealer
- Vehicle Customer
- Vehicle Order
- Vehicle Test Drive
- Vehicle Service Request

Benefits of Tabs:

- Faster access to records for all user roles
- No need to search through multiple apps
- Easier navigation during live customer interactions or support calls

UI/UX Benefits After Implementation

Improvement	Impact
Fast navigation & visibility	Faster customer service and sales
Organized page layouts	Reduced data entry errors
Dynamic forms	Only relevant fields are shown to users
Clear tab-based access	Higher productivity and ease of use
Card-based screen design	Clean and modern UI experience
Lightning App centralized entry	No confusion for first-time users

Overall Summary of Phase 3

The UI/UX customization transforms the CRM from a simple data storage tool into a powerful business-friendly sales portal. The system is designed so that a sales executive, customer support team, or dealer coordinator can perform daily tasks in the least number of clicks while maintaining complete data accuracy.
With the Lightning App, optimized layouts, dynamic forms and Salesforce standard components, the CRM delivers:

- Smooth navigation
- Faster order management
- Clear visibility of vehicle stock
- High adoption rate among users
- Professional and premium interface experience

Phase 4: Data Migration, Testing & Security:

Data Migration:

- Customer details, stock inventory, order information imported using Data Import Wizard & Data Loader
- Validations ensured duplicate emails or invalid order entries are blocked

Testing Performed:

- Unit testing for triggers & batch jobs
- Flow testing for dealer assignment & email reminders
- UAT with sample customers and dealer logins

Security Configuration:

- OWD = Private for Order, Test Drive & Service objects
- Sharing rules for controlled access

Phase 5: Deployment, Documentation & Maintenance:

Deployment:

In the scope of the WhatNext Vision Motors CRM project implementation:

- The complete Salesforce CRM for vehicle management — including custom objects, page layouts, flows, triggers, batch classes, schedulers, and Lightning App has been built and configured in a Salesforce Developer Org.
- The primary objective of this phase is to understand how a Vehicle Ordering & Dealer Management CRM can be developed, tested, and prepared for deployment inside a real organization's Salesforce environment.
- This includes configuring navigation items, security access, Lightning App, and reusable pages for modules such as:
    - Vehicle Inventory
    - Dealer Management
    - Customer Management
    - Vehicle Ordering
    - Test Drive Scheduling
    - Service Request Management
- Each module is designed for real-time vehicle stock tracking, seamless order handling, and structured customer interactions, improving sales and service efficiency.

Actual deployment to Production is not performed because:

- Developer Edition Orgs are standalone and not linked to production.
- Production deployment requires Sandbox, Change Sets, or DevOps CI/CD tools (GitHub, Azure DevOps, Salesforce DevOps Center) — typically used in enterprise projects.

However, all objects, page layouts, automation and Apex components are packaged and structured for easy production migration using standard Salesforce deployment tools in the future.

Maintenance, Monitoring & Troubleshooting

Maintenance ensures that vehicle inventory, order processing, dealer assignments, and customer records remain accurate and up to date.

Ongoing monitoring includes:

- Reviewing Lightning App performance and page-load time.

- Ensuring profiles, permission sets and sharing rules remain aligned with organizational roles (sales team, service team, dealer admin).
- Monitoring flows for:
    - Order assignment issues
    - Test drive reminder email failures
    - Dealer auto-assignment updates
- Reviewing trigger execution logs and batch processing for:
    - Stock decrement errors
    - Order confirmation logic
- Updating picklists, price lists, and vehicle inventory regularly.
- Tracking user and customer feedback to improve UI design and automation reliability.

Although developed inside a Salesforce Developer Org for learning and demonstration, these maintenance activities are crucial for real-time enterprise deployment to ensure application stability, scalability, and reliability.

Project Documentation

Documentation for the WhatNext Vision Motors Salesforce CRM serves as a comprehensive record of the system's purpose, design, and development.
It ensures that business requirements — such as vehicle inventory, dealer assignments, ordering, test drives, and service management — are clearly mapped to system functionalities.

Documentation Benefits

- Acts as a blueprint for developers, admins, and testers.
- Supports new user onboarding and training.
- Helps with troubleshooting, audits and continuous improvement.
- Enables reusability and scalability for future enhancements.

Guidelines for Documentation Submission

- Submit PDF or Word in professional format.
- Use clear headings and bullet points.
- Maintain Times New Roman (12 or 13 size) for readability.
- Errors, plagiarism and misalignment must be avoided.

Mandatory Sections to Include

Project Overview

The WhatNext Vision Motors CRM is a Salesforce-based application that manages the vehicle business lifecycle including inventory management, dealer allocation, customer management, order processing, test drives and service booking.

Customers and internal team members can:

- View available vehicles
- Place orders based on stock availability
- Schedule test drives
- Raise vehicle service requests

The CRM improves customer satisfaction through automation, reduced manual effort, and accurate dealer assignment based on customer location.

Objectives

- Automate customer order handling and dealer assignment
- Prevent orders for out-of-stock vehicles
- Automate test drive reminders through scheduled email notifications
- Enhance customer satisfaction and transparency across the purchase workflow
- Improve business decision-making through real-time vehicle availability and reporting

Phase 1: Requirement Analysis & Planning

- Identify the need to manage vehicle stock, customer details, dealer distribution and order processing inside Salesforce
- Define features:
    - Vehicle inventory
    - Dealer master
    - Vehicle orders
    - Test drives
    - Service requests
- Design data model and security model including custom objects, relationships, page layouts and access levels

Phase 2: Salesforce Development – Backend & Configurations

- Developer Org configuration and Lightning App setup
- Create custom objects and all required fields
- Configure validation rules to prevent invalid/duplicate data
- Build automation using Flows, Triggers, Batch and Scheduled Apex for:
    - Stock validation
    - Stock decrement on order confirmation

    - Daily batch job to convert Pending orders to Confirmed when stock is replenished

Phase 3: UI/UX Development & Customization

- Lightning App WhatNext Vision Motors
- Page layouts for:
    - Vehicles
    - Dealer
    - Customer
    - Orders
    - Test Drives
    - Service Requests
- Navigation Items based on business modules
- Profiles and permission setups based on internal roles

Phase 4: Data Migration, Testing & Security

- Test sample data imported using Data Import Wizard
- Field history tracking for Stock, Status and Price
- Duplicate prevention rules for customer phone/email
- Unit Testing, UAT testing for automation and backend logic
- Security testing for object-level and field-level access

Phase 5: Deployment, Documentation & Maintenance

- Built completely in Salesforce Developer Org
- Production deployment not executed (Developer Org limitation)
- All components configured for future Change Set / DevOps deployment
- Regular monitoring of flows, triggers and scheduled jobs ensures application reliability

Conclusion:

The WhatsNext Vision Motors CRM project demonstrates how Salesforce can automate the entire automotive ordering ecosystem through stock validation, dealer assignment, order management, test-drive reminders and service workflows.

By integrating Flows, Apex Triggers, Batch Apex and Scheduled Apex, the system replaces manual processes with automation, improves accuracy, reduces order delays, and enhances customer satisfaction. Although implemented in a Salesforce Developer Org for demonstration, the architecture is scalable and production-ready for real automobile business use.