WhatNext Vision Motors

Shaping the Future of Mobility with Innovation and Excellence

Published By :Siddhardha Naidu Barla
Email : sbarla2@gitam.in
GITAM Deemed University

Description :

WhatsNext Vision Motors is revolutionizing its customer experience and operational efficiency with a cutting-edge Salesforce CRM implementation. The project streamlines the vehicle ordering process by auto-assigning orders to the nearest dealer based on customer location and preventing orders for out-of-stock vehicles. Automated workflows update order statuses dynamically and send scheduled email reminders for test drives. Key technical implementations include Apex triggers for stock validation, batch jobs for stock updates, and scheduled Apex for automated order processing. This initiative enhances customer satisfaction, improves order accuracy, and boosts overall operational efficiency.

Objects & Relationships :

Object Name	Purpose	Relationships
Vehicle__c	Stores vehicle details	Related to Dealer & Orders
Vehicle_Dealer__c	Stores authorized dealer info	Related to Orders
Vehicle_Customer__c	Stores customer details	Related to Orders & Test Drives
Vehicle_Order__c	Tracks vehicle purchases	Related to Customer & Vehicle
Vehicle_Test_Drive__c	Tracks test drive bookings	Related to Customer & Vehicle

Vehicle_Service_Request__c	Tracks vehicle servicing requests	Related to Customer & Vehicle

Fields & Relationships :

Key Fields for Each Object

1. Vehicle__c (Custom Object)

Vehicle_Name__c (Text)

Vehicle_Model__c (Picklist: Sedan, SUV, EV, etc.)

Stock_Quantity__c (Number)

Price__c (Currency)

Dealer__c (Lookup to Dealer__c)

Status__c (Picklist: Available, Out of Stock, Discontinued)

2.Vehicle_ Dealer__c (Custom Object)

Dealer_Name__c (Text)

Dealer_Location__c (Text)

Dealer_Code__c (Auto Number)

Phone__c (Phone)

Email__c (Email)

3. Vehicle_Order__c (Custom Object)

Customer__c (Lookup to Customer__c)

Vehicle__c (Lookup to Vehicle__c)

Order_Date__c (Date)

Status__c (Picklist: Pending, Confirmed, Delivered, Canceled)

4. Vehicle_Customer__c (Custom Object)

Customer_Name__c (Text)

Email__c (Email)

Phone__c (Phone)

Address__c (Text)

Preferred_Vehicle_Type__c (Picklist: Sedan, SUV, EV, etc.)

5. Vehicle_Test_Drive__c (Custom Object)

Customer__c (Lookup to Customer__c)

Vehicle__c (Lookup to Vehicle__c)

Test_Drive_Date__c (Date)

Status__c (Picklist: Scheduled, Completed, Canceled)

6. Vehicle_Service_Request__c (Custom Object)

Customer__c (Lookup to Customer__c)

Vehicle__c (Lookup to Vehicle__c)

Service_Date__c (Date)

Issue_Description__c (Text)

Status__c (Picklist: Requested, In Progress, Completed)

Automation :

1. Dealer Assignment Flow

Automatically assigns the nearest dealer to a Vehicle Order based on customer location.

- **Flow Type:** Record-Triggered Flow
- **Trigger Object:** Vehicle_Order__c
- **Trigger Condition:** When a record is created or updated
- **Logic:**
    - Retrieve related `Vehicle_Customer__c` record based on `Customer__c` lookup in Vehicle Order.
    - Fetch the `Address__c` field from Vehicle_Customer__c.
    - Query all `Vehicle_Dealer__c` records that match the customer address using `Dealer_Location__c`.
    - Assign the `Id` of the matching dealer to the `Dealer__c` field on the Vehicle Order.

2. Test Drive Reminder Flow

Sends SMS reminders to customers for upcoming test drives.

- **Flow Name:** Test Drive Reminder - V1
- **Flow Type:** Record-Triggered Flow
- **Object:** Vehicle_Test_Drive__c
- **Paths:**
    - **Run Immediately:**
        - Get Customer Information
        - Send Test Drive Reminder via Action (e.g., Twilio/Email Alert)
    - **Reminder Before Test Date:** (Scheduled Path)
        - Triggers before the test drive date/time

Apex Classes and Triggers

🔧 VehicleOrderTriggerHandler.cls

Handles validations and stock updates for Vehicle Orders.

```
None
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);
        }
    }
```

```
    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.');
                }
            }
        }
    }

    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

Trigger for Vehicle_Order__c to call the handler methods.

```
None
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.cls

Batch Apex job to process pending vehicle orders.

```
None
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.cls

Schedules the batch job to run periodically.

```
None
global class VehicleOrderBatchScheduler implements
Schedulable {
    global void execute(SchedulableContext sc) {
        VehicleOrderBatch batchJob = new
VehicleOrderBatch();
        Database.executeBatch(batchJob, 50); // Batch
size
    }
}
```

Summary

This Salesforce project automates critical processes in a vehicle ordering system:

- Ensures out-of-stock vehicles can’t be ordered.
- Automatically confirms and decrements stock for valid orders.
- Assigns nearest dealers to orders based on location.
- Sends reminders to customers for test drives.

Tools Used: Flows, Apex Triggers, Apex Classes, Batch Jobs, Schedulers
Objects Involved: Vehicle__c, Vehicle_Order__c, Vehicle_Dealer__c, Vehicle_Customer__c, Vehicle_Test_Drive__c

Let me know if you'd like to export this as PDF or add visuals (e.g., screenshots of flows).