SALESFORCE INTERNSHIP PROJECT REPORT

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

This report tells the working of the "WhatNext Vision Motors" project, a Salesforce solution developed during my internship with Smart Bridge. This project was a deep dive into how technology can reshape the automotive industry.

Introduction:

In an era of rapid technological change and a growing demand for sustainability, What Next Vision Motors is steering towards a new frontier in the automotive world. Our goal isn't just to build cars, but to redefine the entire experience of mobility by weaving together digital technology, creative solutions, and an unwavering focus on the customer.

This project was born from that vision. We set out to tackle the core processes of vehicle ordering, stock management, and customer communication using the power of Salesforce. We aimed to move beyond manual tasks and build a system that was not only efficient but also intelligent, using automated flows and data-driven insights to make smarter decisions. This was more than a technical upgrade; it was about building a foundation for a smarter, more customer-focused future.

Learning Outcomes

- Designed Salesforce schema using custom objects and relationships
- Automated workflows with Record-Triggered Flows
- Applied Apex best practices using handler classes and reusable logic
- Built scalable solutions using Batch and Scheduled Apex

Objectives and Purpose of Project:

The project's objective is to streamline and automate vehicle ordering, dealership allocation, and stock management using Salesforce CRM features.

- Suggesting the nearest vehicle dealer based on customer location.
- Preventing orders for vehicles that are out of stock.
- Sending scheduled email reminders for test drives.

1. Salesforce CRM Implementation

- The ability to store and organize vehicle information, stock status, and dealer information within Salesforce.
- The ability to store and easily find customer orders, test drives, and service requests.
- The ability to assign requests and orders to the nearest dealer based on the customer's location.

2. Process Automation

- Stop customers from placing an order if no cars are available.
- Auto-assign customers to the nearest dealer.
- Automatically send email reminders for scheduled test drives.

3. Apex and Triggers

- Utilize Apex triggers to enforce business rules like stock approval and automatically assigning dealers to orders.
- Employ trigger handlers to ensure code is modular and easy to manage, following best practices.

4. Batch Jobs

- An Apex batch job will look at vehicle stock status on a regular basis to update it.
- Schedule individual emails notifying about stock and orders

Implementing the Objectives:

1. Salesforce CRM Implementation & Data Modelling

To achieve the objectives, a custom data model was created using several custom objects within a Salesforce Developer Account.

Automate those work flows where dealerships get customer orders based on the closest dealer to where the customer lives.

First, we need to create a few custom objects like Vehicle, Vehicle Dealer, Vehicle Customer, Vehicle Order, Vehicle Service Request, Vehicle Test Drive in the Salesforce Developer Account.

Custom Objects Created:

Object Name	Purpose	Relationships
Vehicle__c	Stores vehicle details	Lookup to Dealer & Vehicle Orders
Vehicle_Dealer__c	Stores authorized dealership information	Related to Orders
Vehicle_Customer__c	Stores customer details	Related to Orders & Test Drives
Vehicle_Order__c	Tracks vehicle purchase orders	Lookup to Customer & Vehicle
Vehicle_Test_Drive__c	Tracks test drive appointments	Lookup to Customer & Vehicle
Vehicle_Service_Request__c	Tracks service requests	Lookup to Customer & Vehicle

These custom objects were then organized with custom tabs and integrated into a new Lightning App named "WhatNext Vision Motors" created through the Lightning Experience App Manager. This app was made available to the System Administrator profile.

2. Process Automation

Fields and Relationships:

- **Vehicle__c:** Contains fields like Vehicle Name, Stock Quantity, Status, Price, and a lookup relationship to the Dealer.

  Dealer (Lookup / Master-Detail to Dealer__ c):
  The vehicle is linked to a specific Dealership.

- **Vehicle_Dealer__c:** Includes Dealer Name, Dealer Location, and contact information.

  Vehicle __c → Dealer __c (Lookup / Master-Detail):
  One dealer can have multiple vehicles.
  Order __c → Dealer __c

  Orders must be connected to the correct dealer as either the Dealer has the order and inventory available, or the Dealer is the closest dealer to the customer.

- **Vehicle_Customer__c:** Holds customer details including name, contact info, and address fields like City, State, and Pin code.

  Order __c → Customer __c
  Test _Drive __c → Customer __c

- **Vehicle_Order__c:** Has lookup relationships to Customer, Vehicle, and Dealer.

  Customer (Lookup to Customer __c):

  Connects order to the person ordering.

  Vehicle (Lookup to Vehicle __c):

  Connects to an ordered vehicle.

  Dealer (Lookup to Dealer __c):

  Makes clear which dealer is processing the order.

- **Vehicle_Service_Request__c:** Includes lookups to Customer and Vehicle.

  Vehicle (Lookup to Vehicle __c)

  Dealer (Lookup to Dealer __c):

- **Vehicle_Test_Drive__c:** Features lookups to Customer and Vehicle.

  Customer (Lookup to Customer __c):
  Who is taking the test drive.
  Vehicle (Lookup to Vehicle __c):
  Which vehicle is going out for the test drive.
  Allows us to schedule and track test drives.

Custom Objects	Fields and Relationships
Vehicle	Vehicle name, Vehicle Model, Stock Quantity, Status, Price, Dealer
Vehicle Dealer	Dealer Name, Dealer Code, Dealer Location, Phone, Email
Vehicle Customer	Vehicle Customer Name, Preferred Vehicle Type, Phone, Email, Customer Name
Vehicle Order	Vehicle, Vehicle Order Name, Status, Order Date, Customer
Vehicle Service Request	Vehicle, Vehicle Service Request Name, Status, Service Date, Customer, Issue Description
Vehicle Test Drive	Vehicle, Test Drive Date, Status, Customer

Auto-assign orders to the nearest dealer based on the customer’s location:

The flow does the following in sequence:Retrieve the customer's location (e.g., city, pin code) from the associated customer record.

Search for the nearest dealer based on the location criteria.

Update the Vehicle Order to auto-assign the most appropriate dealer.

In automating the process, the flow ensures that:

- The dealer is assigned faster without any manual intervention.
- The customer's experience is improved (orders routed as intended).
- All orders consistently apply the same business logic accurately.

Send reminders for scheduled test drives automatically via email:

This is a Record-Triggered Flow, which will run whenever a new Vehicle Test Drive record is created or updated.

The flow does the following steps, in order:

By automating this communication process, the flow enables:

- Customer reminders to be provided in a timely manner (i.e. reducing the chance of missed appointments).
- Sales teams to save time, as reminders can go out without the need for any manual follow up.
- Test drive reminders to follow a standard process, which facilitates consistency and customer happiness.

3. Apex and Trigger

In this project, Apex was used to implement custom business logic that could not be handled through declarative tools like Flows. Apex Triggers, combined with Handler Classes and Helper Classes, played a crucial role in automating key operations like stock validation and automatic dealer assignment during the vehicle ordering process.

Trigger: VehicleOrderTrigger

This trigger is defined on the custom object Vehicle_Order__c. It is responsible for:

- Preventing customers from placing an order if the selected vehicle is out of stock.
- Automatically assigning the nearest available dealer based on customer location.

Events Covered: before insert, before update

Instead of embedding logic directly inside the trigger, it delegates processing to a separate **handler class**, following best practices for modular and maintainable code.

Trigger Handler: VehicleOrderTriggerHandler

This Apex class contains the actual logic executed during trigger events. It ensures:

- **Stock Validation**:
  Before an order is saved, it checks the stock level of the selected vehicle. If the stock quantity is 0 or less, an error is thrown using addError(), preventing the order from being saved.
- **Stock Decrement on Confirmation**:
  If the order is confirmed, the vehicle's stock is automatically reduced by 1 to reflect accurate inventory.
- **Separation of Concerns**:
  Each method in the handler class handles a specific function (e.g., preventOrderIfOutOfStock(), updateStockOnOrderPlacement()), making the code clean and easy to debug or enhance.

4. Batch Apex Job: Stock Re-Evaluation

To handle large volumes of order records and ensure real-time accuracy of stock and order status, a **Batch Apex job** was implemented. This component automates the review of pending orders and stock levels on a scheduled basis, reducing manual effort and improving data consistency.

1. Class: VehicleOrderBatch

This Apex class implements the Database.Batchable interface and is responsible for:

- **Identifying** all Vehicle_Order__c records with status "Pending".
- **Checking** whether the requested vehicle is now back in stock.
- **Updating** the order status to "Confirmed" if stock is available.
- **Decrementing** the vehicle stock by 1 upon confirmation.

2. Core Logic Flow

- **Start method**: Collects all Vehicle_Order__c records where status = "Pending".
- **Execute method**:
    - Iterates over each pending order.
    - Retrieves associated vehicle stock.
    - If stock is available:
        - Updates the order status to "Confirmed".
        - Reduces vehicle stock count.
- **Finish method**: Can be extended to send notifications or log the results.

3. Scheduled Apex Class: VehicleOrderBatchScheduler

To ensure this job runs automatically every day, the scheduler class implements the Schedulable interface. It uses Salesforce’s System.schedule() method to invoke the batch class daily.

Schedule Details:

- Cron Expression: '0 0 12 * * ?' (Runs daily at 12:00 PM)
- Command:

System.schedule('Daily Vehicle Order Processing', cronExp, new VehicleOrderBatchScheduler());

Conclusion

WhatNext Vision Motors' successful adoption of Salesforce CRM revolutionised key business operations in the automotive industry. We automated dealer assignment, expedited vehicle order placement, and guaranteed precise stock validation through this project, all of which directly increased operational efficiency and enhanced the general customer experience.

This project demonstrates Salesforce's potent ability to connect intelligent data management and automation. It has established the groundwork for a strong dealership management system that can grow with the company in the future while also addressing the operational difficulties of today.

Key Takeaways:

- Used customer geolocation data to eliminate manual dealer assignment.
- Using Apex-based order validation, stock-related errors were avoided.
- Using Flows, I made sure clients received timely reminders for test drives.
- Using planned batch jobs, system performance and integrity were maintained.

In conclusion, this project is a great illustration of how Salesforce can support digital transformation in the automotive sector by providing CRM solutions that are intelligent, effective, and customer-focused. In addition to accomplishing its technical objectives, it had a significant influence on future-readiness and business process optimisation.

Submitted By

Name: D. Krishna Karthik
Email: krishnakarthik4129@gmail.com
GitHub Repository: github.com/kk4129/WhatNext-Vision-Motors
Roll No: 22761A5413
College: Lakireddy Bali Reddy College of Engineering