# Practice Salesforce Implementation - Details & Requirements

The following are the details and requirements for a practice Salesforce project. This is geared toward a software developer who is learning the platform for the first time. The business case (managing sales and service for a car dealership) is meant to be general enough where it is well understood by most people, so that the focus can be on the technical aspects of the project rather than on understanding the business requirements. Additionally, the project was designed so that it provides the developer with plenty of exposure to the standard Sales and Service data model, but with heavy customizations on top of that.

## Goals

The project is intended to provide the learner with practice in the most widely used areas in the Salesforce Developer profession. The concepts tested here are practical and nearly universally applicable to large-scale Salesforce implementations.

If successful, this project will afford the developer familiarity in the following areas, among others:

* Data modeling and object creation
  + Standard objects
    - Account, Contact, Lead, Opportunity, Case, Task
  + Custom object creation and use
  + Creating fields of each type
  + Formula fields
  + Validation rules
  + Record Types
* User security model and provisioning
  + Profiles, Permission Sets, Roles
  + Queues and Groups
* Configurable User Interface Configuration
  + App setup, tabs, and pages
* SOQL
* Triggers
  + Validations
  + Synchronous data updates - automated record creation and update
  + Near universal trigger design patterns, including:
    - Trigger handlers
    - Trigger bypasses
* Apex
  + Data Types
    - Primitives
    - Collections
    - Complex Objects
  + Language constructs like control statements, annotations, classes, interfaces, etc.
  + HTTP Requests
  + Unit testing
  + SOQL in Apex
  + Error handling and logging
  + OOP
  + Asynchronous Apex
    - Future, Schedulable, Batch, Queueable
* Lightning Web Components
  + Will learn LWC and have a brief understanding of and exposure to Aura
  + Will be able to define Visualforce and its properties, but will not do Visualforce development
* Platform Technologies
  + Custom metadata, labels and settings
  + Static resources
  + Platform cache

## Details

We are going to use Salesforce to manage the sales for a **car dealership**.

Our primary goals of the system:

1. Manage data and contact information for our customers
2. Track and manage dealership inventory
3. Handle sales pipeline for our sales team
4. Compare performance and efficiency of sales team
5. Store the types of vehicles that are in production or have been previously produced
6. Manage the service department and its work

Initially, we are seeking to track inventory, vehicle details, and sales. Later we will expand to service.

We will utilize the standard Salesforce objects where appropriate, in the following ways:

* **Accounts** will represent our *customers*
* **Opportunities** will represent our *sales*
* **Leads** will represent our *prospects*
* **Users** will represent our *salespeople*

## Requirements

[Requirements Board - Trello](https://trello.com/b/riQJxlOd/development-work)

#### Data Model

We will utilize the following combination of standard and custom objects:

Standard Objects

* **Accounts** (Customers)
  + Enable Person Accounts
  + **Number\_of\_Purchases** - *req. - Integer*
  + **Customer\_Rating** - *req. - Decimal number from 0 to 100 with two decimal places*
  + Standard Person Account fields:
    - **FirstName, LastName, Address, Phone, Email**
* **Opportunities** (Sales)
  + Standard Opportunity fields:
    - Name, Amount, CloseDate, StageName, Probability,
  + **Vehicle** - *req. -* lookup to Vehicle record
  + **Account** - req. - lookup to customer Person Account
  + **Owner** - *req*. - (represents sales person who’s managing the sale)
  + **Make** - *req. -* pulled from Vehicle\_\_c relationship; should be formula to pull the Make\_\_r.Name from the Model\_\_c
  + **Model**  - *req.* - pulled from the Vehicle\_\_r.Model\_\_r.Name field
    - Make and Model as formulas improves user experience by making these important data points accessible directly on the Opportunity page.
  + **Lost Reason *-*** picklist
    - No Longer Buying
    - Went with Competitor
    - Financially Ineligible
    - Could not Agree on Terms
    - Considering Another Vehicle
  + **StageName** - picklist
    - The Opportunity.StageName field is a standard field that drives an organization’s sales process. The updates to this field are important, as they determine the probability of an Opportunity’s closing, and whether or not the Opportunity will be won or lost.
    - Our Opportunities are car sales, so the stages should be updated to reflect that sales process.
    - The stage names are in bold, along with their probabilities to the right:
      * **In-Person Inquiry** / 10%
      * **Test Drive** / 20%
      * **Application Started** / 40%
      * **Financial Verification** / 60%
      * **Papers Signed** / 90%
      * **Closed - Won / Delivered** / 100%
        + *This is a* ***closed*** *stage and a* ***won*** *stage.*
      * **Closed - Lost** / 0%
        + *This is a* ***closed*** *stage and a* ***lost*** *stage.*
    - Stage definitions:
      * **In-Person Inquiry:** Customer arrived at the dealership and is looking for a vehicle. This is different than someone who called or filled out a web form for more information, both of whom would be Leads.
      * **Test Drive:** This stage may be skipped over in rare instances. But it indicates that a customer is taking the vehicle for a test drive.
      * **Application Started:** The sales team and the customer are working together on paperwork to initiate the purchase of the vehicle.
      * **Financial Verification:** The finance team is validating the customer’s financial situation – either by pulling credit if the deal is financed, or verifying bank details if it’s a cash transaction.
      * **Papers Signed**: The salesperson and customer are finishing the final paperwork to complete the transaction.
      * **Closed - Won / Delivered:** The vehicle has been signed over and delivered to the customer. The transaction is final.
      * **Closed - Lost**: The customer is no longer interested in, or eligible for, the vehicle.
* **Lead** - *requirements forthcoming*
* **Case** - *requirements forthcoming*

Custom Objects

* **Make** (Makes of Vehicles - Ford, Cadillac, etc.)
  + **Name** - *req.*
  + ***Number of Models*** *- req. -* Integer
    - Automatically populated based on the child Models where **In Production = true**
  + **Country of Origin** - *req. -* String (80)
    - Where manufacturer is based
* **Model** (Name of specific vehicle model - Mustang, Camry, etc.)
  + **Name** - *req.*
  + **Make** - *req. -* master/detail to Make\_\_c
  + **Vehicle Type** - *req. -* picklist
    - Car
    - Truck
    - SUV
    - Van
  + **In Production** - Boolean
* **Vehicle** (Represent our dealership’s inventory)
  + **Name -** *req. - auto number*
  + **VIN Number** - *req. -* String (17)
  + **Model** - *req. -* master/detail to **Model**
  + **Year** - *req.*  - Integer (4)
  + **Pre-Owned** - *req. -* Boolean
    - Indicates if Vehicle is new or used
  + **Price** - *req. -* Currency
    - How much we are selling the car for
  + **Purchase Price** - *req. -* Boolean
    - What the dealership paid for the vehicle
  + **Status** - *req. -* Boolean
    - In Stock
    - Pending
    - Sold
  + **Purchase Date -** Date
  + **Sell Date -** Date

#### Vehicle Validation

* Opportunities represent a potential sale. If the customer wants to buy a different vehicle than originally intended, a new Opportunity should be created. That is, once Vehicle is set, we should not permit updates to the **Opportunity.Vehicle\_\_c** field.
  + If attempted, display error message: “The vehicle cannot be changed in an ongoing sale. If the customer would like to switch vehicles, please close this opportunity and create a new one with the updated vehicle.”
* When an Opportunity is created, verify the related Vehicle exists and is In Stock
  + If **Vehicle\_\_c.Status\_\_c** is not In Stock, display error message: “Sorry, this vehicle is not available.”
* When a Vehicle is added to an Opportunity, update the Vehicle record status to ‘Pending’. This will prevent the Vehicle from being sold or applied for by other customers.

#### Opportunity Data Updates

* Automatically populate the **Opportunity.Name** field with the following format:
  + Account.Name (Full Name of Person Account) - Vehicle\_\_r.Make\_Name\_\_c (created above) Vehicle\_\_r.Year Vehicle\_\_r.Model\_\_r.Name
  + **Example:**
    - John Doe - 2023 Porsche 911
    - Carlos Lopez - 2019 Toyota Camry
* Create **Make\_\_c** field on **Vehicle\_\_c** - formula field - reference Model\_\_r.Make\_\_r.Name
* When the **CloseDate** of the Opportunity is in the past, the Owner should be notified via email, with their Manager CC’d. This should run nightly at 1am. Template details and AC in Trello.
* When **OwnerId** is changed on **Opportunity**, check if that user is a member of the **Sales** queue; if not, add them to the Sales queue. (Group and GroupMember objects).

Closing a Sale

Winning a Sale

* When an Opportunity Stage is changed to **Closed - Won**:
  + Update its Vehicle record Status to ‘Sold’
  + Increment the related customer (Account) **Number of Vehicles Purchased** field by one

Losing a Sale

* When a sale/opportunity is **Closed - Lost**, the user should be required to specify the reason. This will help improve our processes in the future.
* This **Lost Reason** field is required if the **StageName** is **Closed - Lost**, and should not be completed otherwise.
  + This validation lends itself to a validation rule. Implement this logic with one or two validation rules, to make sure those fields cannot be set without the other one in a valid state.

#### Data Creation

* Dan will create a class for the creation of data in this org.
* The class, called **DataInitializer**, should create:
  + **Makes** (10)
  + **Models** (20+)
  + **Vehicle Inventory** (100)
    - The following Vehicle details should be randomized:
      * Parent **Model\_\_c** (pick at random to simulate a real inventory)
      * **VIN** **Number** (random 17-character alphanumeric string)
      * **Year** (random assortment of new / used)
* Class will not handle creation of:
  + Accounts
  + Opportunities
  + Leads
* Class will contain a public method to **initData()** and will insert a number of records in the above tables

#### Technical Features

* (Dan) The system should support logging and monitoring. Create an exception handling/logging object and class: **Logger.cls** and **Error\_Log\_\_c**. This will be used to log exceptions from our code and our API integrations, and we will report on these errors later.
* (Dan) Create framework / helper for making HTTP Requests. Use custom metadata to make these configurable, so we have an easier way to call external systems. We will leverage this later when we write integration code, like sending sales details to Slack.
* (Dan) Support a custom metadata-driven Trigger Bypass framework, where triggers can be enabled/disabled without deployments. Should work for either specific objects, or shut off all triggers. Should have the ability to work globally, for specific users, or for specific profiles.
* (Dylan / Dan) All features, both on this slide and preceding/subsequent slides, must have Apex tests with coverage of at least 95% to be considered complete.
* (Dylan / Dan) No hardcoded strings or “magic numbers” in the codebase. Everything should be placed in a **Constants** class. Later we will use custom metadata / custom labels for certain values.
* (Dylan) Alert users when their sale is overdue – i.e., the sale’s CloseDate is in the past but the sale is not yet finished
* (Dylan) Sales should follow-up with the customers approximately five weeks after the purchase to ensure things are going well. Schedule follow-up reminders for the sales team. If any follow-up reminders from previous weeks were not completed, mark them as Overdue and forward this in a report to management.