Salesforce Project Implementation Phases with Concepts (Admin + Developer)

Project Title: Manufacturing After-Sales & Service CRM

Phase 8: Data Management & Deployment

This phase covers the strategies and tools used for managing data within the CRM and deploying the developed features from one environment to another. A robust data management and deployment process is essential for maintaining data quality and ensuring a smooth and reliable release cycle.

Data Import Wizard and Data Loader

- **Use Case:** To populate the Salesforce org with initial data for testing, demonstration, and training purposes, both the Data Import Wizard and Data Loader can be utilized. This project includes sample data in CSV format for all major custom objects, such as products, cases, engineers, and feedback.
- **Implementation Details:**
  - **Data Import Wizard:** Used for importing smaller sets of data (up to 50,000 records) for objects like Service_Agent__c and Engineer__c. Its user-friendly interface makes it ideal for ad-hoc data loads.
  - **Data Loader:** For larger data sets, such as Registered_Product__c and Service_Case__c, the Data Loader is the preferred tool. It provides more advanced features, such as bulk API support for faster processing.
  - **Sample Data:** The project includes the following sample data files that can be used for import:
    - sample_products.csv
    - sample_service_cases.csv
    - sample_engineers.csv
    - sample_dispatch.csv
    - sample_feedback.csv
    - sample_serviceagent.csv
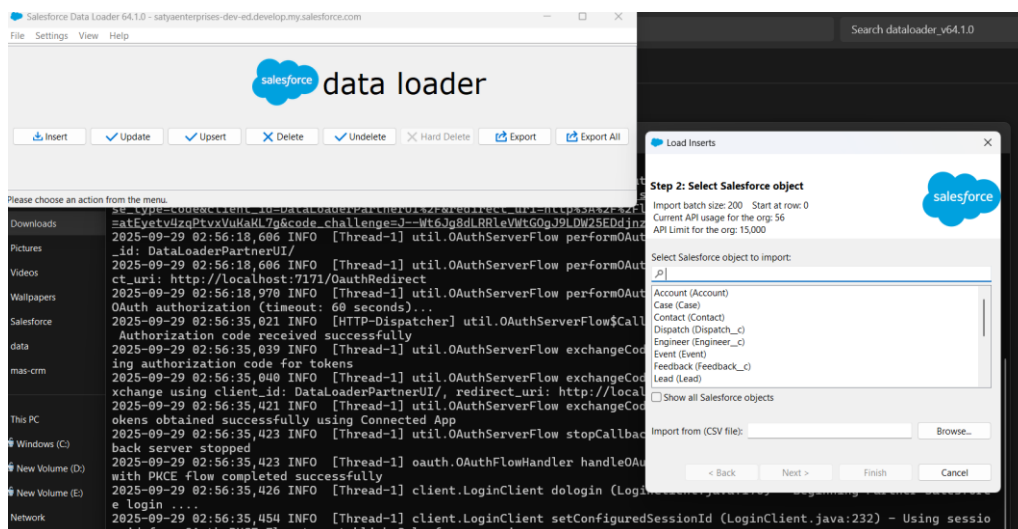
Step 3: Mapping

Import batch size: 200   Start at row: 0
Current API usage for the org: 27
API Limit for the org: 15,000

[Choose an Existing Map]   [Create or Edit a Map]

CSV Column Header Name	Salesforce Object Field Name
Subject__c	Subject__c
Customer_Name__c	Customer_Name__c
Customer_Email__c	Customer_Email__c
Customer_Address__c	Customer_Address__c
Customer_Location__Latitude__s	Customer_Location__Latitude__s
Customer_Location__Longitude__s	Customer_Location__Longitude__s
Registered_Product__r.Serial_Name__c	
Priority__c	Priority__c

Duplicate Management

- **Use Case:** To maintain data quality and prevent the creation of duplicate records, a duplicate rule is implemented for the Registered_Product__c object. Since each registered product should have a unique serial number, this rule ensures that a new product record cannot be created with a serial number that already exists in the system.
- **Implementation Details:**
  - **Object:** Registered_Product__c
  - **Field:** Serial_Number__c
  - **Configuration:** The Serial_Number__c field is marked as a unique field, which acts as an implicit duplicate rule, preventing duplicate entries.

SETUP
Duplicate Rules

Registered Product Duplicate Rule
Registered Product - Serial Exact

Help for this Page

Duplicate Rule Detail

[Edit] [Delete] [Clone] [Deactivate]

Rule Name	Registered Product - Serial Exact
Description	
Object	Registered Product
Record-Level Security	Enforce sharing rules
Action On Create	Block
Action On Edit	Allow
Alert Text	A product with this Serial Number already exists: {matchingRecordLink}
Active	✓
Matching Rule	Rule1 Mapped
Conditions	
Created By	Puneet Verma, 26/09/2025, 3:49 pm

Order	1 of 1 [Reorder]
Operations On Create	Alert Report
Operations On Edit	Alert Report
Matching Criteria	Registered Product: Serial_Number EXACT MatchBlank = FALSE
Modified By	Puneet Verma, 26/09/2025, 3:49 pm

[Edit] [Delete] [Clone] [Deactivate]

Data Export & Backup

- **Use Case:** To protect against data loss and for compliance purposes, a regular data backup strategy is essential. Salesforce provides several tools for exporting data, which can then be stored in a secure, off-platform location.
- **Implementation Details:**
    - **Data Export Service:** This built-in Salesforce feature can be scheduled to perform a weekly or monthly export of all the organization's data. The exported data is provided as a set of CSV files in a zip archive.
    - **Data Loader:** The Data Loader can also be used to manually export data from specific objects as needed.

VS Code & SFDX

- **Use Case:** The entire project is developed and managed using the Salesforce DX (Developer Experience) toolset and Visual Studio Code. This source-driven development methodology allows for better collaboration, version control, and automated deployments.
- **Implementation Details:**
    - **Project Structure:** The project is organized in a Salesforce DX project structure, with all metadata and code stored in a local repository.
    - **Scratch Orgs:** Scratch orgs are used for development and testing, providing a clean and disposable environment for each new feature or task. The config/project-scratch-def.json file defines the shape of the scratch org.
    - **Deployment:** The sf project deploy start command is used to deploy the metadata from the local project to a scratch org, sandbox, or production org. This command-driven approach is ideal for integrating into a CI/CD pipeline.
    - **Source Control:** The project is intended to be used with a version control system like Git, with the .gitignore file configured to exclude files and directories that should not be committed to the repository.

> applications
> aura
> classes
> contentassets
> customMetadata
> flexipages
> flowDefinitions
> flows
> layouts
> lwc
> namedCredentials
> objects
> permissionsets
> staticresources
> tabs
> triggers
> manifest
> scripts
.forceignore
.gitignore
.prettierignore
.prettierrc
CUSTOM_JSON_RESPONSES_GUIDE.md   U
data-dictionary.md
docformat.md
eslint.config.js
jest.config.js
NAMED_CREDENTIALS_DIAGRAMS.md   U
NAMED_CREDENTIALS_GUIDE.md   U
NAMED_CREDENTIALS_SUMMARY.md   U

> OUTLINE
> RUNNING TASKS

main* ⊗ 0 ⚠ 0   MAS-Dev   ⊗ RHDA analysis has failed