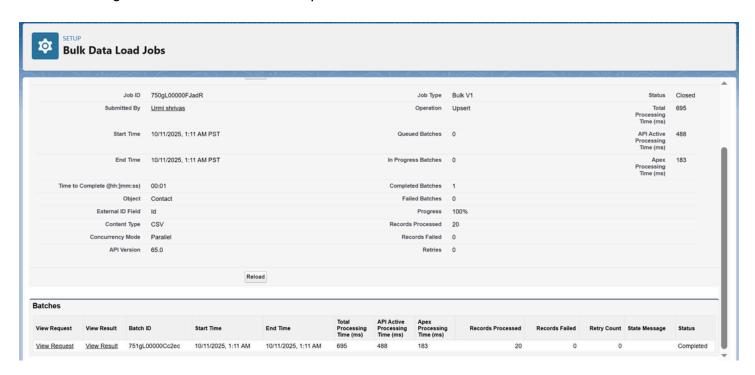
Phase 8: Data Management & Deployment

1. Data import Wizard

- Definition:
- Data Import Wizard ek Salesforce ka built-in tool hai.
- Purpose
- Manual data entry ki jagah fast aur accurate import provide karna.
- Projects me testing aur data migration ke liye use hota hai.
- Usage in Project:
- Client data ko CSV se Salesforce me migrate karna.
- Marketing event ke leads bulk me import karna.



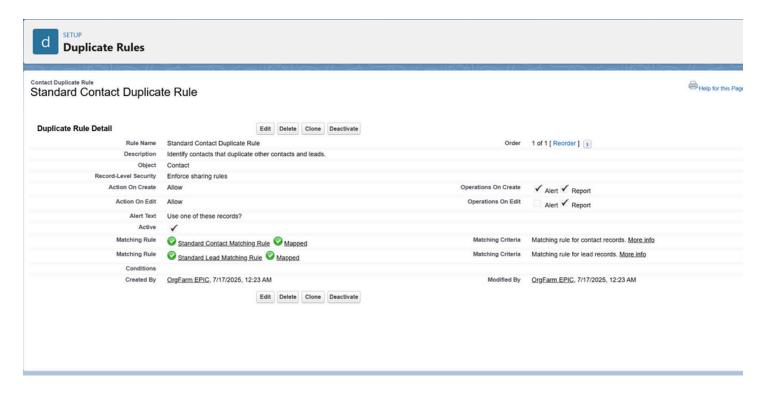
2. Data Loader

- Data Loader is a client application used for bulk data operations like insert, update, upsert, delete, and export.
- It efficiently handles large datasets and supports advanced field mapping and automation.
- Not implemented for this project since the dataset is small and all records were handled using the Data Import Wizard.

3. Duplicate rules

- The Standard Contact Duplicate Rule automatically prevents duplicate contacts in Salesforce.
- It checks matching criteria like Email, First Name, and Last Name to identify duplicates.

 In this project, it was tested to ensure duplicate contacts trigger an alert without blocking critical data entry.



4. Data export and Backup

- Data Export was used to create a secure backup of all project records in CSV format.
- It ensured that Contacts, Leads, and Accounts are safely stored offline.
- This was done to prevent data loss and maintain data integrity for the project.

Monthly Export Service

Help for this Page

Data Export lets you prepare a copy of all your data in salesforce.com. From this page you can start the export process manually or schedule it to run automatically. When export is ready for download you will receive an email containing a link that allows you to download the file(s). The export files are also available on this page for 48 hours, after which time they are deleted.



5. Change sets

- Change Set is used to deploy metadata components like fields, objects, and rules from one Salesforce org to another.
- It allows smooth migration between sandbox and production environments without manual recreation.
- Not used in this project because all development was done in a single org, so deployment between orgs was not needed.

6. Unmanaged vs Managed Packages

- **Unmanaged Package:** A package where components (like objects, fields, and code) can be freely modified after installation in the target org.
- **Managed Package:** A controlled package where components are locked, updates are managed by the publisher, and it is mostly used for AppExchange apps.
- In this project, packages were not required as all development was done directly within a single org.

7. ANT Migration Tool

- The ANT Migration Tool is a command-line utility used to deploy and retrieve metadata between Salesforce orgs.
- It enables automated, bulk, and scripted migration, ideal for large or complex projects.

8. VS Code & SFDX

- VS Code with SF CLI was set up to manage project metadata and connect the local project to the Salesforce org.
- Commands like sf org list, sf project status, and sf org open were used to verify the connection.
- Some errors appeared at multiple points, so most development and testing were completed directly in the Salesforce org (browser).

```
> SEARCH
                                                export default class AccountManager extends LightningElement {
DEBUGGER-PROJECT
                                                    handleInputChange(event) {
 ∨ force-app\main\default
                                                         this[event.target.name] = event.target.value;

✓ lwc

✓ accountDashboard

   JS accountDashboard.js
                                                     handleCreateAccount() {
   accountDashboard.js-meta.xml
                                                         createAccount({ name: this.name, type: this.type, phone: this.phone })
  \checkmark accountManager\_tests_
                                                             .then(result => {
 accountManager.html
                                                                 this.createdAccountId = result;
                                                                  console.log('Account created with Id: ' + result);
    accountManager.js-meta.xml
   JS accountManager.test.js
                                                                  const createdEvent = new CustomEvent('accountcreated', { detail: { id: result, name:
 > objects
                                                                 this.dispatchEvent(createdEvent);
 > staticresources
                                                                 this.name = '';
 > tabs
                                                                 this.type = '';
  > triggers

∨ LocalServiceCRM

 > .husky
                                                                 console.error('Error creating account:', error);
  > .vscode
  > config
  > force-app
                                                 Ĭ
  > scripts
> OUTLINE
```

•