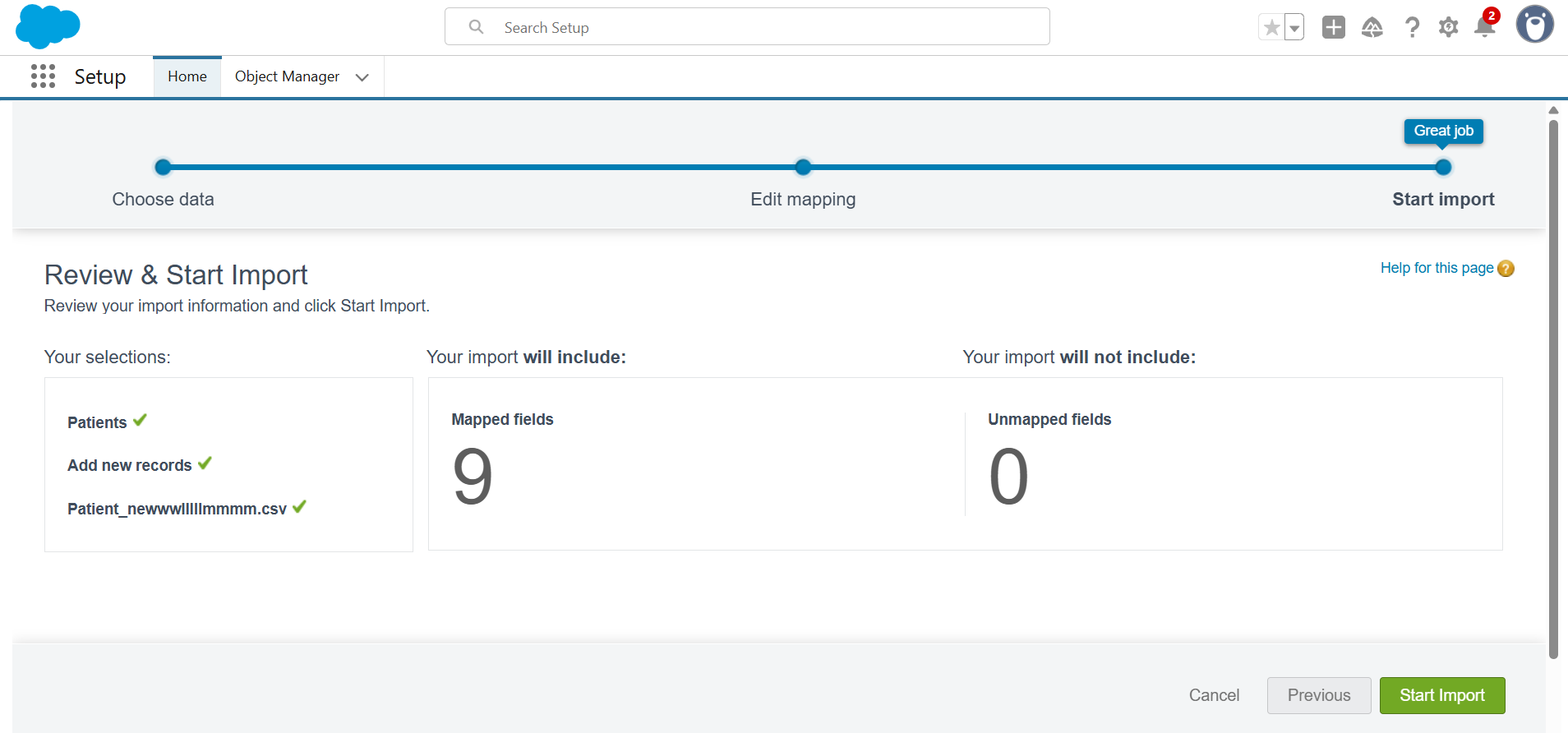
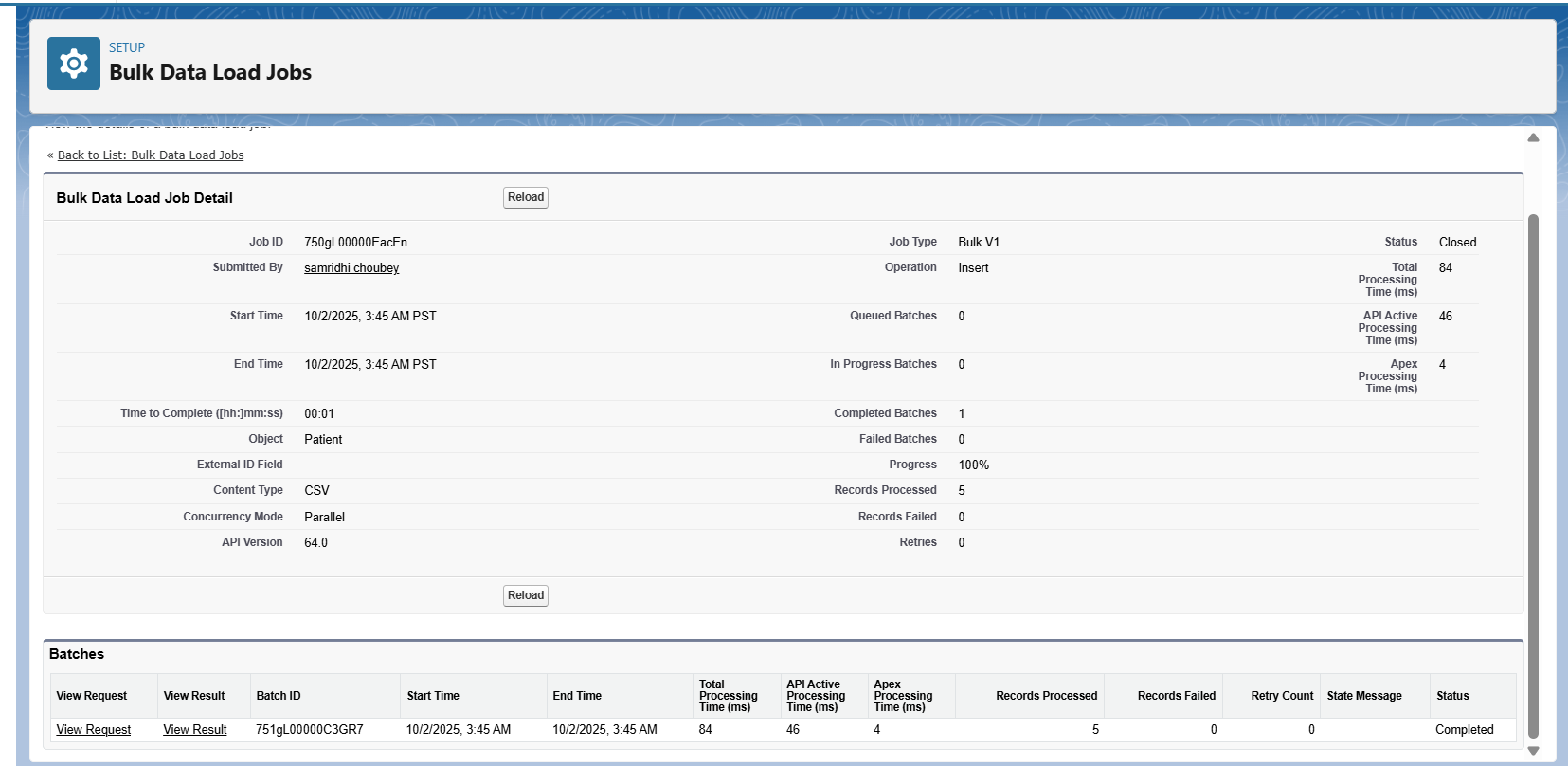
**Phase 8: Data Management & Deployment**

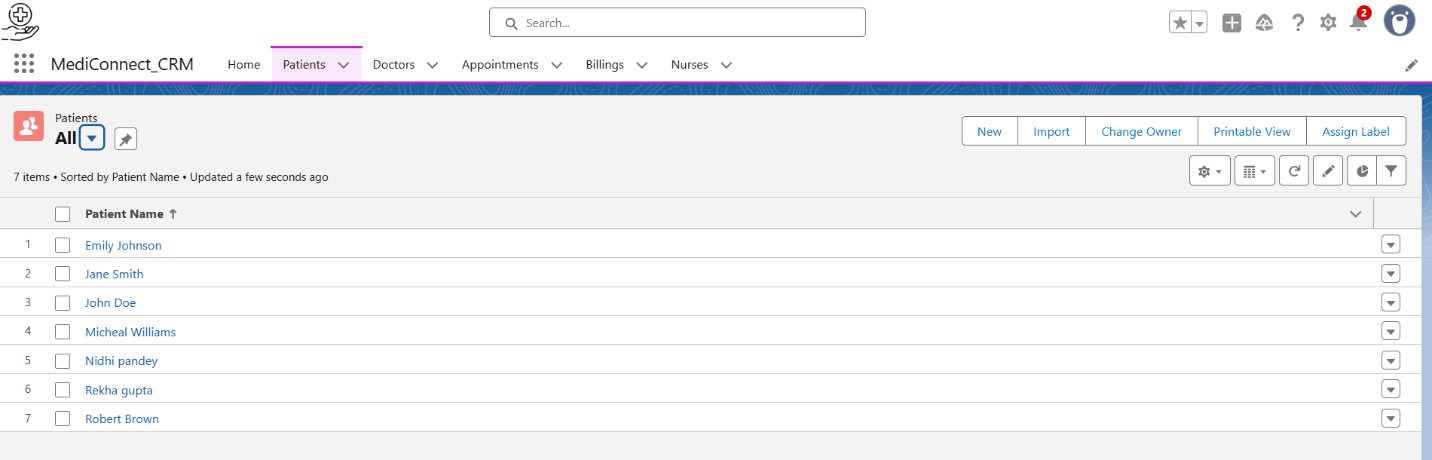
**1.Data Import Wizard :**

* A CSV file was prepared with patient details (First Name, Last Name, Age, Email Id, Mobile Number, Blood Group, Sex, Address, Disease Description, Past Medical History).
* Using Data Import Wizard, the file was uploaded and fields were mapped with the Patient\_\_c object.
* The data was imported successfully, and patient records were created in Salesforce.



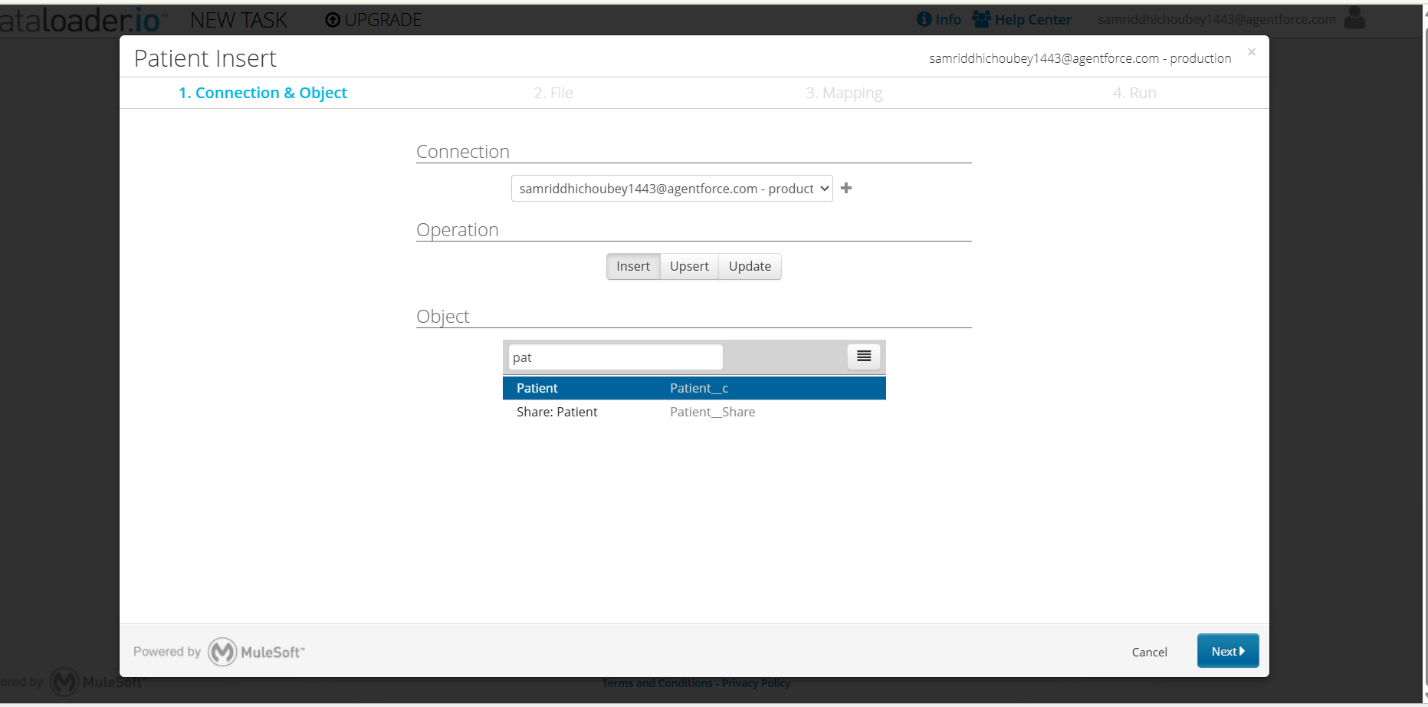


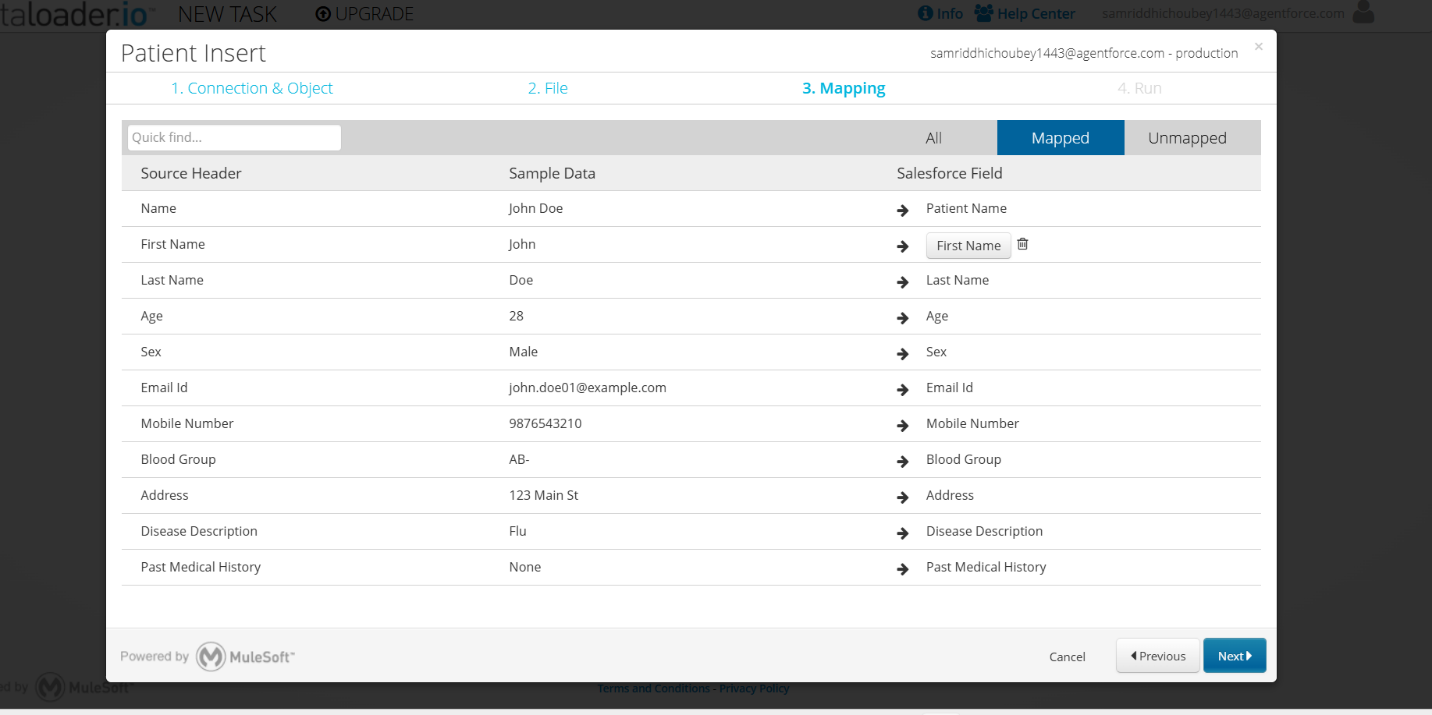
**Successfully imported all the records from .csv file:**

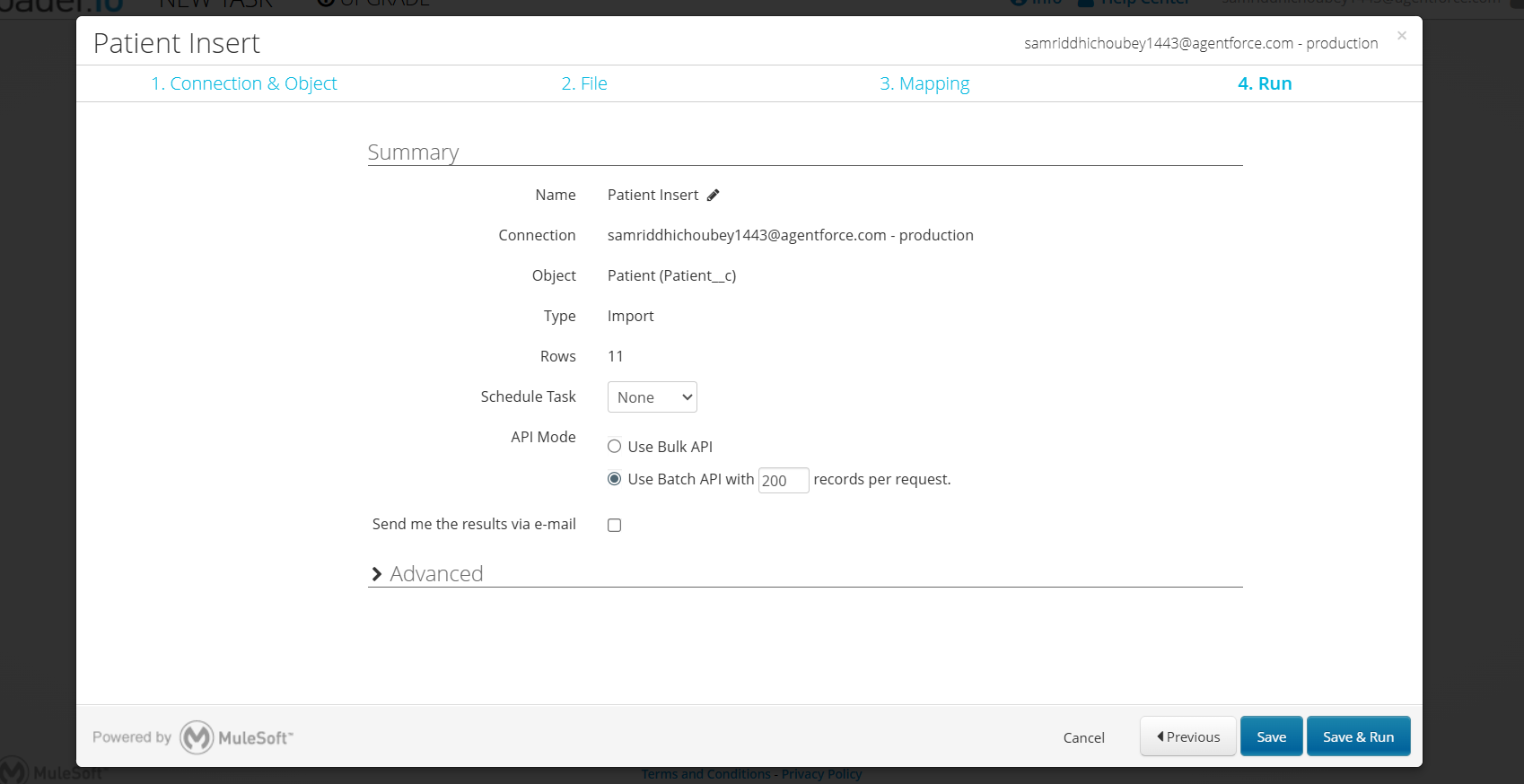


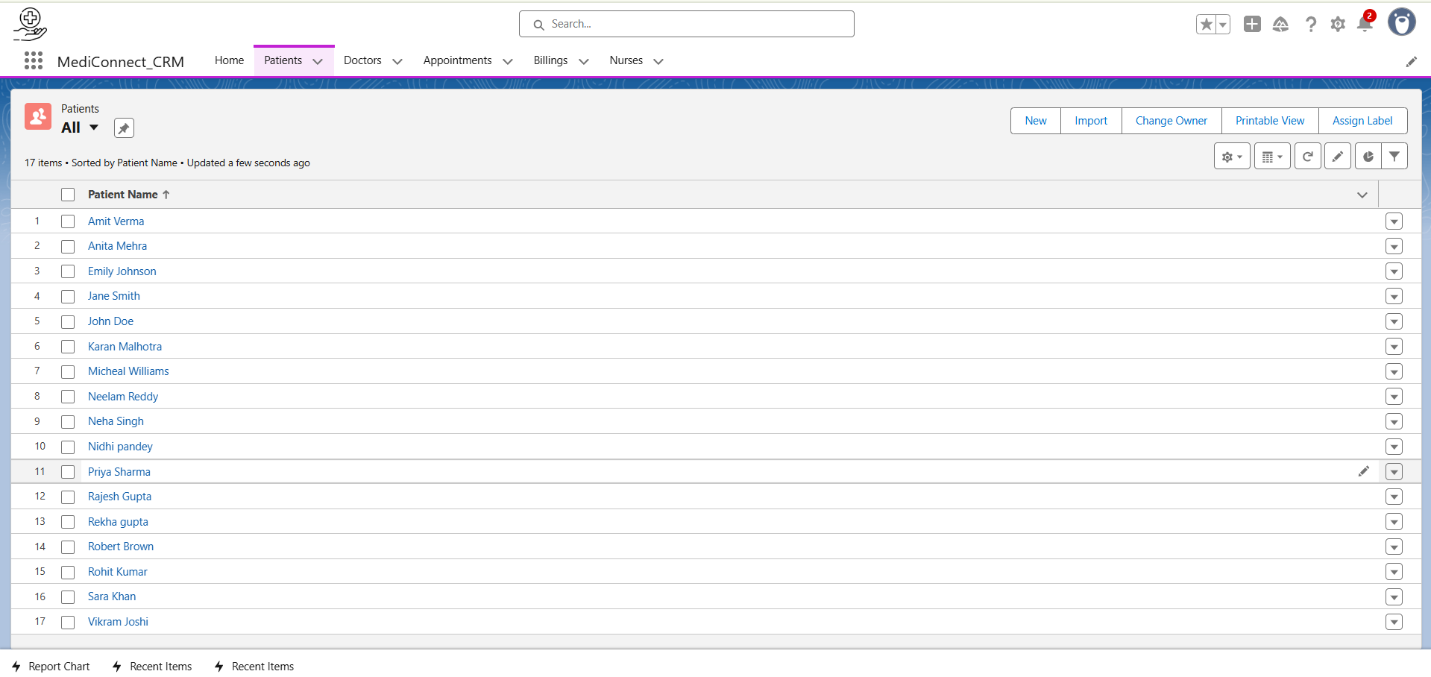
**2.Data Loader**

* **Data Loader** is a client application used to import, export, update, and delete Salesforce records in bulk.
* For MediConnect, Data Loader was used to import patient records into the Patient\_\_c object.
* A **CSV file** containing patient details was prepared.
* Using Data Loader:
  + Selected **Insert** operation.
  + Chose the **Patient\_\_c** object.
  + Uploaded the CSV file.
  + Mapped CSV headers with Salesforce fields.







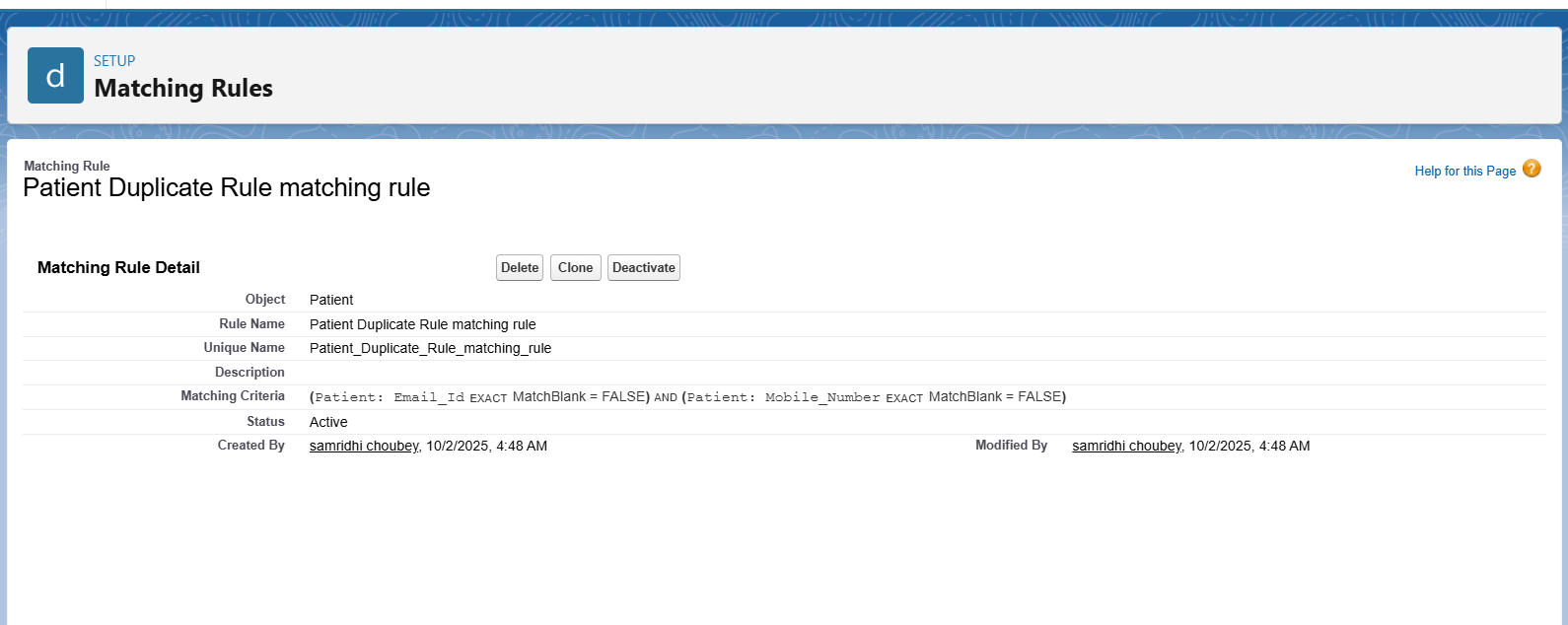


* **The records were successfully inserted into Salesforce, and new patient data was visible in the application.**

**3.Duplicate Rule:**

* A Duplicate Rule was created for Patient\_\_c to prevent duplicate patient records.
* The rule uses Email Id and Mobile Number as matching criteria.
* When a duplicate is detected, Salesforce either alerts the user or blocks record creation, ensuring data integrity in MediConnect.

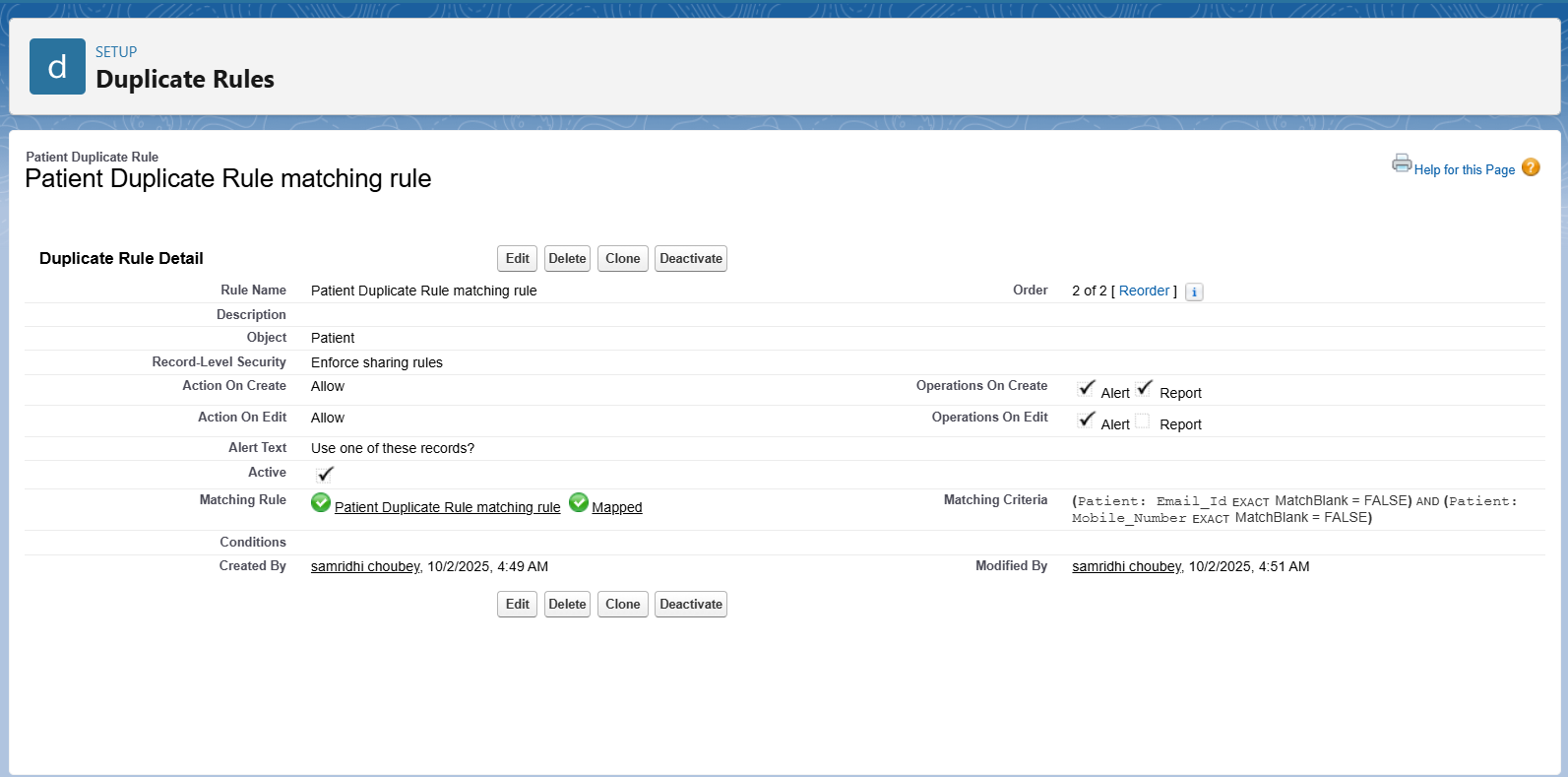
For creating duplicate rule first we have to create **matching rule**. Therefore **Patient Duplicate Rule matching** is created,



Now we will create Duplicate Rule

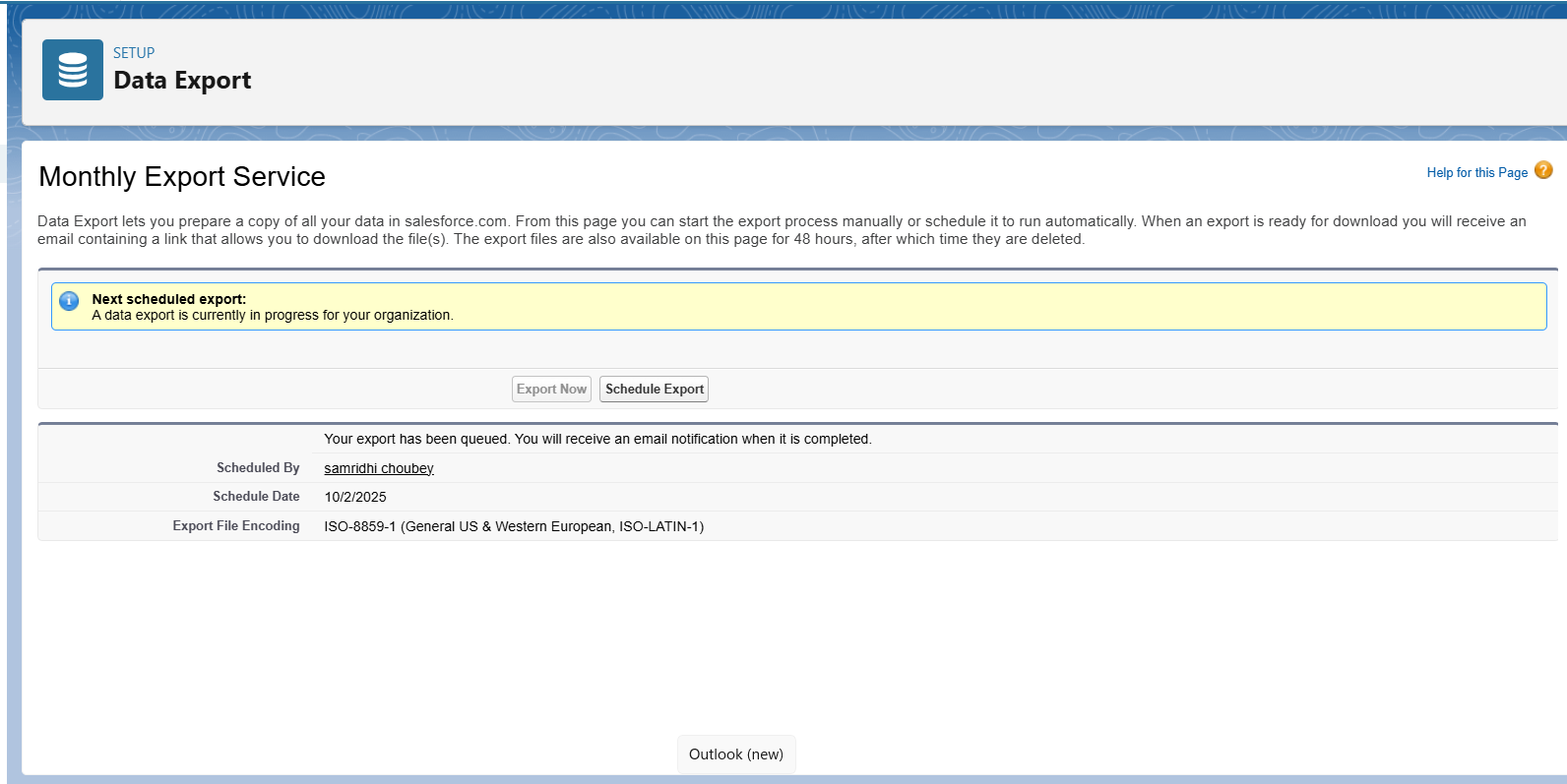
Rule name- **Patient Duplicate matching rule**

* The rule uses Email Id and Mobile Number as matching criteria.



**5.Data Export & Backup**

* Data Export & Backup was performed for Doctor\_n\_\_c records in MediConnect.
* Salesforce provides the exported data in CSV format inside a .zip file.
* This ensures a secure backup of patient data for recovery or migration purposes.



**5.Change Sets**

Purpose

* Change Sets allow you to deploy metadata changes (like objects, fields, Apex classes, Lightning pages) from one Salesforce org to another.
* For MediConnect, used to move customizations from Sandbox → Production safely.

**Steps (Conceptual)**

* **Create Outbound Change Set in Sandbox**
* Go to **Setup → Change Sets → Outbound Change Sets**.
* Click **New** → provide a **Name** and **Description** (e.g., MediConnect\_Objects\_Deployment).

**Add Components**

* Add all required components to the Change Set:
* Custom Objects: Patient\_\_c, Appointment\_\_c
* Fields & Relationships
* Apex Classes & Triggers
* Lightning Pages / App
* Save the Change Set.
* **Upload to Target Org**
* Click **Upload → select Production org** (target org).
* Salesforce sends a notification in Production.
* **Deploy in Production**
* Log in to **Production → Setup → Inbound Change Sets**.
* Select the uploaded Change Set → click **Deploy**.
* All components are deployed to Production.

**Outcome**

* All **custom objects, fields, Apex classes, triggers, and Lightning pages** are conceptually moved to Production.
* Provides a **safe deployment method**, avoids manual errors, and tracks all deployed components.

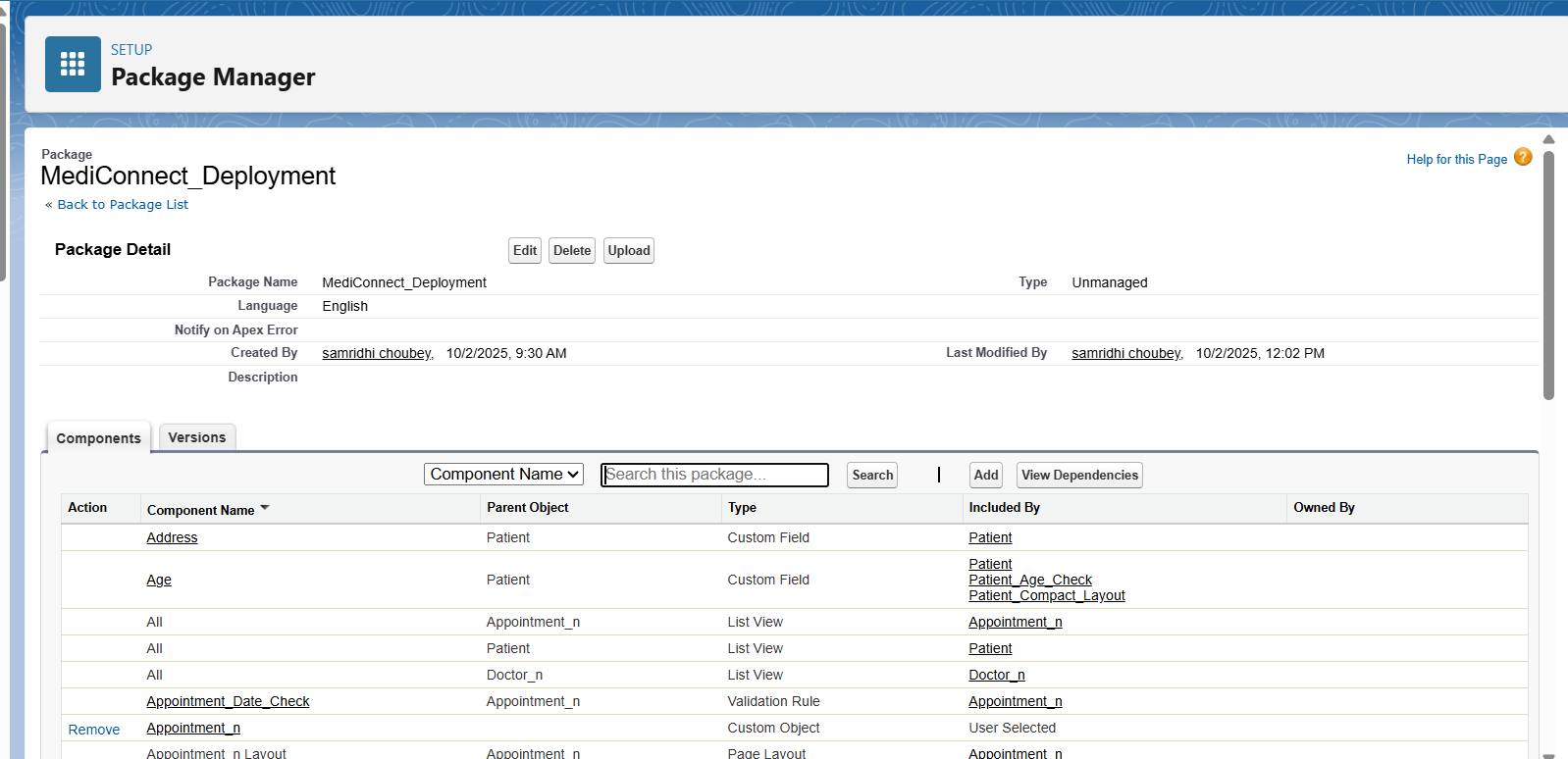
**5.Deployment – Unmanaged vs Managed Packages**

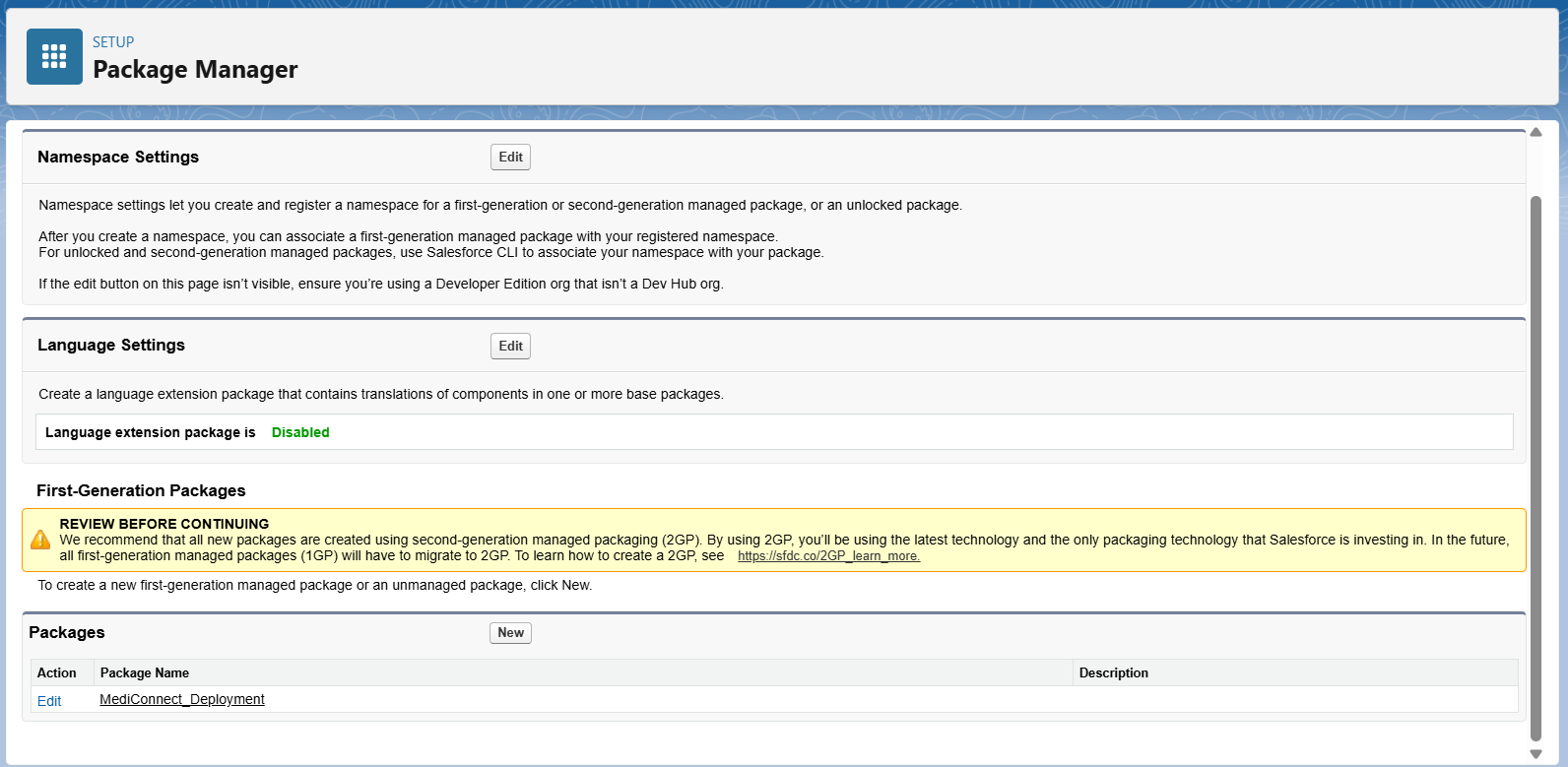
Salesforce provides two types of packages to bundle and distribute customizations:

1. **Managed Packages**
   * Created by Salesforce ISV partners.
   * Components are protected (cannot be edited after installation in subscriber orgs).
   * Supports versioning and upgrades.
   * Requires Apex test coverage ≥ 75% for deployment.
2. **Unmanaged Packages**
   * Used for one-time distribution of components.
   * Components are editable in the target org.
   * No versioning support.
   * Commonly used for learning, demo, or small projects.

**Steps Followed to Create & Upload an Unmanaged Package**

1. **Create the Package**
   * Navigate to **Setup → Packages → New**.
   * Provide:
     + **Package Name:** e.g., HospitalManagementPackage
     + **Description:** Short description of components.
     + **Type:** Select **Unmanaged**.
2. **Add Components to the Package**
   * Click **Add Components**.
   * Include:
     + **Custom Objects**: Patient\_\_c, Doctor\_\_c, Appointment\_\_c
     + **Fields**: All custom fields created for these objects
     + **Page Layouts & Compact Layouts**
     + **Validation Rules**: e.g., Patient\_Age\_Check, Appointment\_Date\_Check
     + **Apex Classes & Triggers**: e.g., CreateDefaultAppointment trigger
     + **List Views**, **Record Types**, **Custom Tabs**, etc.
3. **Specify Object & Feature Requirements**
   * Check **Record Types, Sharing, Feed Tracking** for all custom objects used.
   * Verify **additional feature requirements** if any (Duplicate Rules, Custom Notifications).
4. **Upload the Package**
   * Go to the **Upload** option of the package.
   * Provide:
     + **Version Name:** e.g., Spring 2025
     + **Version Number:** e.g., 1.0
     + **Release Notes:** Optional
   * Leave URL and Post-install instructions as **None** (for this project).
5. **Handle Apex Test Requirement (if any)**
   * For **Managed Packages**, ensure **Apex test coverage ≥ 75%**.
   * For **Unmanaged Packages**, tests are optional but recommended.
   * Test classes can be created via **Developer Console → File → New → Apex Class**.
6. **Verify the Package**
   * Once uploaded, note the **Installation URL**.
   * Test installation in a **different Salesforce org** if needed.





**6. ANT Migration Tool**

The ANT Migration Tool is a Java/Ant-based command-line utility that allows developers and admins to deploy and retrieve Salesforce metadata between orgs (sandbox, production, or dev orgs).

**Key Features:**

* Deploy and retrieve metadata using XML configurations (package.xml).
* Supports incremental deployment.
* Can be integrated into CI/CD pipelines.

**7.VS Code & Salesforce DX (SFDX)**

Salesforce DX is a modern, source-driven development approach that integrates with VS Code to manage Salesforce metadata and development lifecycle efficiently.

**Key Features:**

* Source-driven development.
* Easy integration with version control (Git).
* CLI commands for metadata deployment, retrieval, and testing.