EduCRM Student Admission Support System

Phase 1: Problem Understanding & Industry Analysis

Requirement Gathering

Educational institutions, especially colleges and universities, face challenges in handling thousands of student inquiries during admission seasons. Currently, most follow a **manual or semi-digital approach** (Excel sheets, emails, phone calls). This causes:

- Loss of potential student leads due to poor tracking.
- Delays in responding to students and parents.
- Difficulty in maintaining a **single source of truth** for student data.
- No real-time visibility of admission progress for management.

Thus, the requirement is to build a **centralized CRM system** that automates lead tracking, improves student communication, and provides real-time dashboards.

Stakeholder Analysis

- **Admission Officers** → Manage student inquiries, follow-ups, and applications.
- **Students & Parents** → Expect timely updates, clear communication, and transparency in admission progress.
- **Administrators/Management** → Need dashboards and reports to track admission performance and forecast future trends.
- **IT/Support Staff** → Ensure system availability, user training, and data management.

Business Process Mapping

Current Admission Process (Manual):

1. Student inquiry received via phone/email/form.
2. Admission officer manually records data in Excel or registers.

3. Follow-up reminders are handled manually.
4. No clear tracking of how many inquiries convert to admissions.

Proposed Salesforce-based Admission Process:

1. Inquiries automatically captured as **Leads** in Salesforce.
2. Leads qualified and converted into **Opportunities (Admission Applications)**.
3. Once confirmed, students stored in a **custom object (Student Record)**.
4. Automated email/SMS reminders keep students/parents informed.
5. Dashboards & reports give real-time insights into admissions.

Industry-Specific Use Case Analysis (Education Sector)

- **Problem:** High volume of inquiries, lack of follow-ups, poor data visibility.
- **Use Case:** CRM can help institutions manage the full **student lifecycle** from inquiry to admission.
- **Impact:**
    - Increased student conversion rates.
    - Improved communication with students/parents.
    - Enhanced reputation of the institution with a transparent admission process.

AppExchange Exploration

Salesforce **AppExchange** already offers education-specific solutions like:

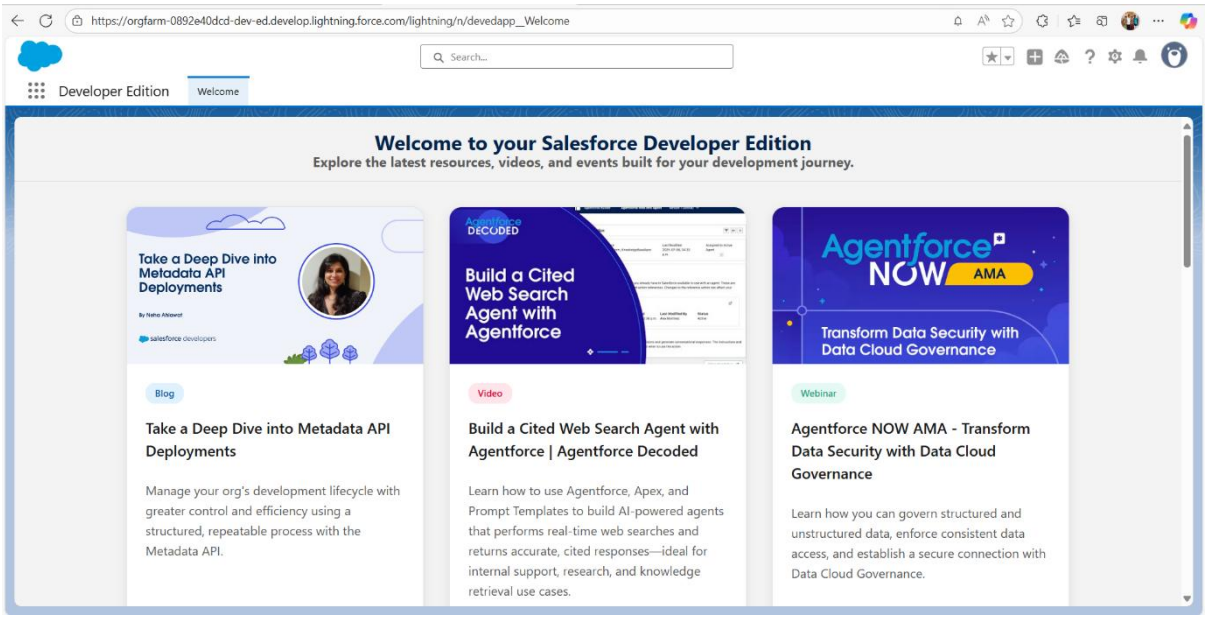
- **Salesforce.org Education Cloud** (for higher education institutions).
- **TargetX CRM** (student recruitment and engagement).
- **Enrollment Rx** (end-to-end admission management).

Our EduCRM project will take inspiration from these real-world solutions but will be a **simplified custom CRM**, tailored for admission tracking and student communication.

Phase 2: Org Setup & Configuration

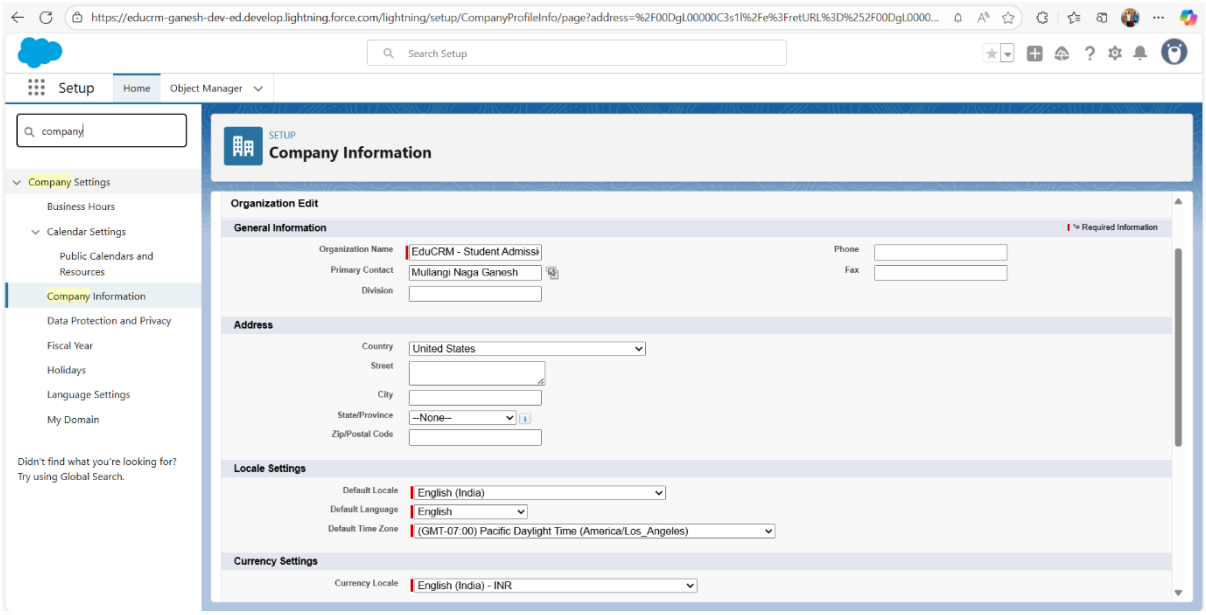
1. Salesforce Editions

- **Developer Edition** was used for this project.
- Provides access to core Salesforce CRM functionalities, suitable for testing and development purposes.



2. Company Profile Setup

- Default company settings were used in the Developer Org.
- Company information such as address, default currency, and locale settings were configured according to standard defaults.



3. Business Hours & Holidays

- Default business hours and holidays were used.
- Not critical for the project since automated processes do not depend on business hours at this stage.

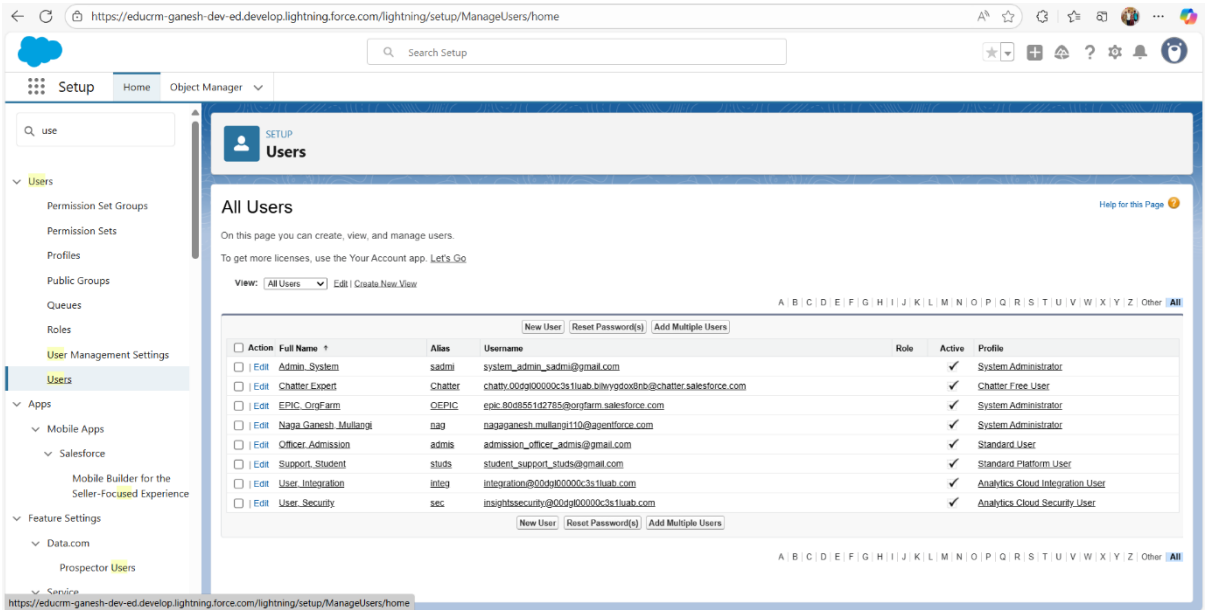
4. Fiscal Year Settings

- Standard fiscal year settings applied.
- Optional for small-scale projects; no financial reporting automation required at this stage.

5. User Setup & Licenses

- **Users Created for EduCRM:**
- Users represent key roles in the EduCRM system for managing admissions and student support.

Name	Username	Profile	License
Admin	System_admin_sadmi@gmail.com	Admin Profile	Salesforce
Admission Officer	Admission_officer_admis@gmail.com	Admission Officer Profile	Salesforce Platform
Student Support	Student_support_studs@gmail.com	Student Support Profile	Salesforce Platform

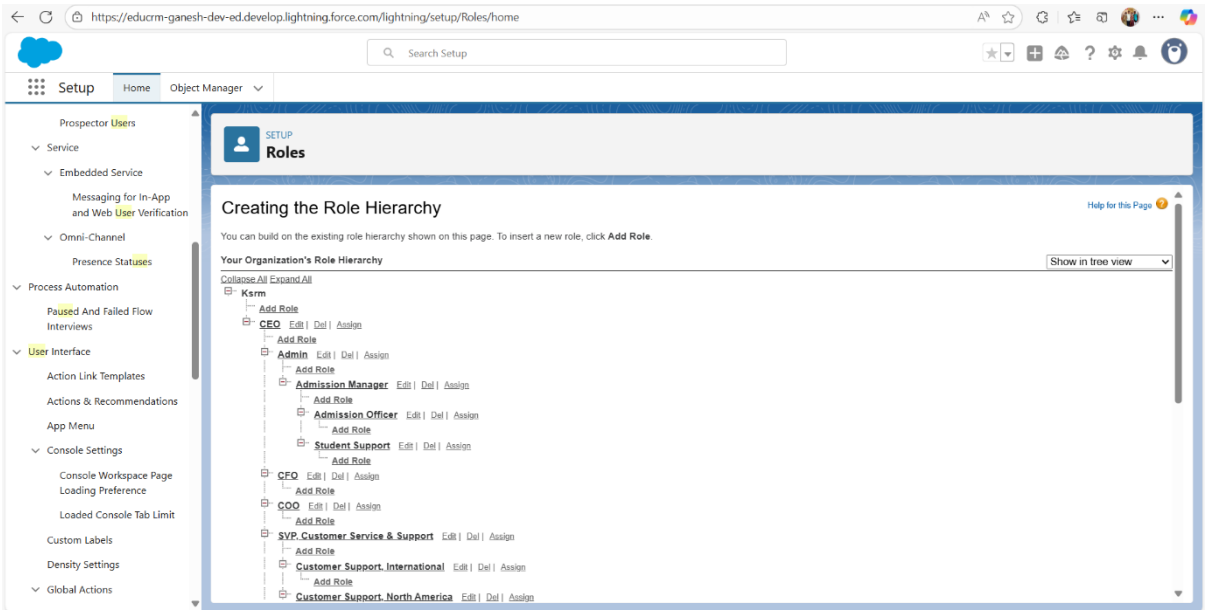


6. Profiles

- **Profiles Created / Cloned:**
  1. **Admin Profile** → Full access to all objects and fields.
  2. **Admission Officer Profile** → Access to Student and Admission Application objects.
  3. **Student Support Profile** → Access to Student and Support Ticket objects.
- **Field-Level Security (FLS):** Profiles configured to ensure appropriate visibility and edit permissions.

7. Roles

- Role hierarchy ensures proper record access: Admin sees all records, Admission Officers and Student Support see only relevant data.



8. Permission Sets

- Not created for this project as all access is managed via Profiles.
- Optional for more advanced access control in larger projects.

9. OWD (Organization-Wide Defaults) & Sharing Rules

- Default OWD settings used.
- No sharing rules configured since access control is handled via Roles and Profiles.

10. Login Access Policies

- Default login access policies applied.
- Ensures secure login for all users.

11. Dev Org Setup

- Developer Org is configured and ready for customizations.
- Sandbox not required for this project.

12. Sandbox Usage

- Not applicable in Developer Org for this project.

13. Deployment Basics

- Deployment not required at this stage; changes are directly made in Developer Org.

Phase 3: Data Modeling & Relationships

3.1 Standard & Custom Objects

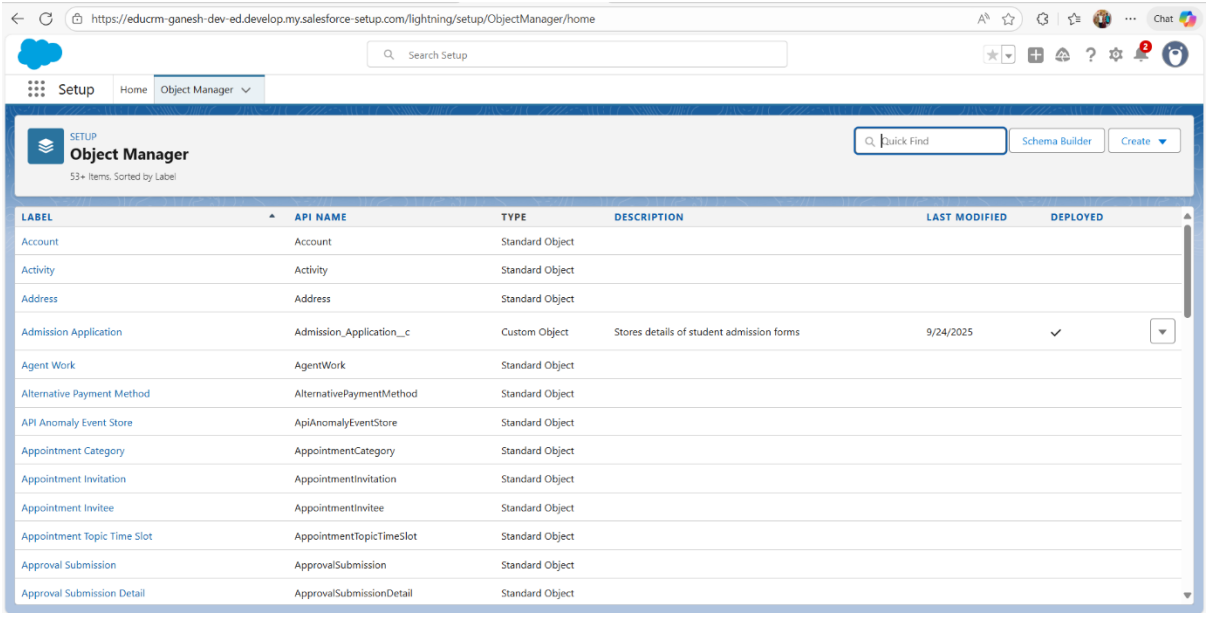
Use Case

The admission process involves handling student data, course applications, and approval tracking. Salesforce provides standard objects like *User* and *Account*, but to meet educational needs, we created a custom object:

Admission Application — to manage each student's admission form and approval status.

Objects Used

Type	Object Name	Purpose
Standard	User	Represents system users (Admin, Admission Officer, Student Support)
Custom	Admission Application	Captures details of a student's application for a specific course
(Optional) Custom	Student	Stores student profile details (if implemented later)



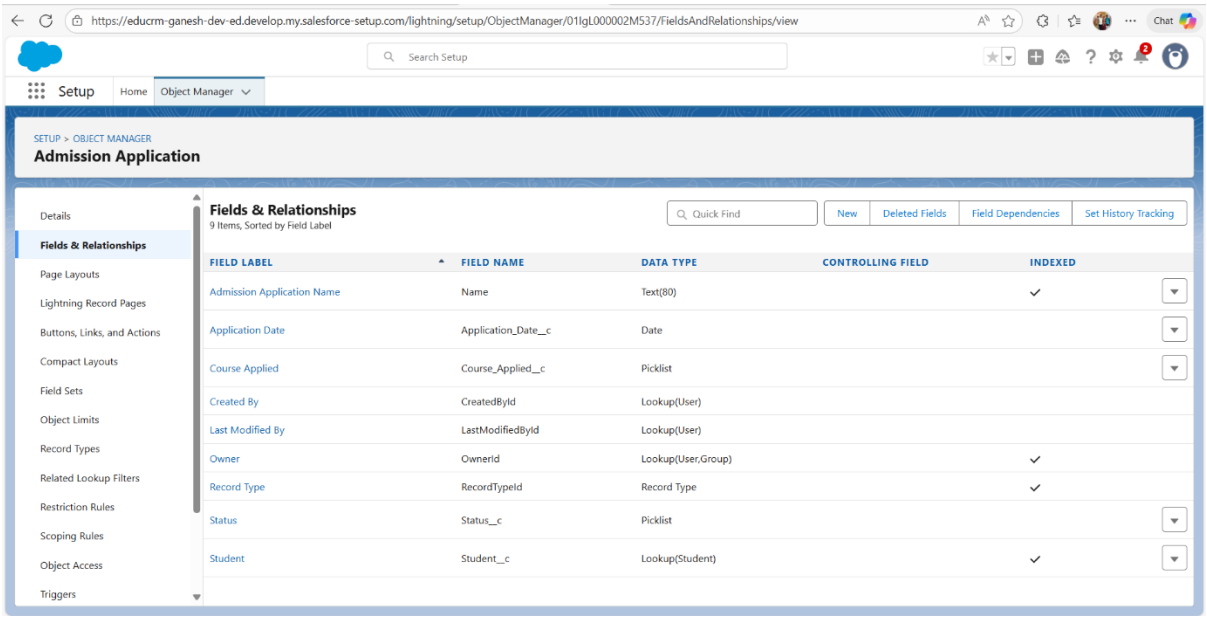
3.2 Fields

Use Case

To store detailed information about each admission record, we created several fields under the Admission Application object.

Fields Created

Field Label	Data Type	Description
Admission Application Name	Text	Unique name for each application
Student	Lookup (Student)	Links the admission application to the student record
Course Applied	Text / Picklist	Stores the course name applied for
Status	Picklist	Represents current status: Pending, Approved, Rejected



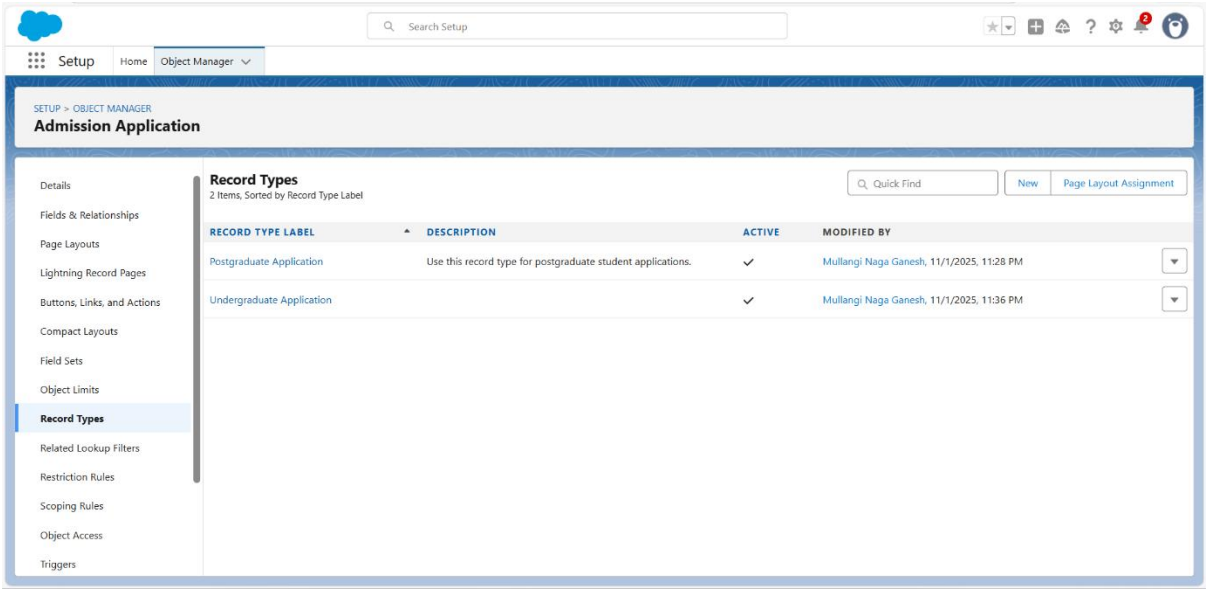
3.3 Record Types

Use Case

To handle different kinds of admission applications (e.g., Undergraduate, Postgraduate), we created **Record Types**.

This allows customized layouts and processes based on the type of admission.

- **Record Type Name:** Undergraduate Application
- **Active:** ✅ Yes
- **Associated Layout:** Undergraduate Application Layout



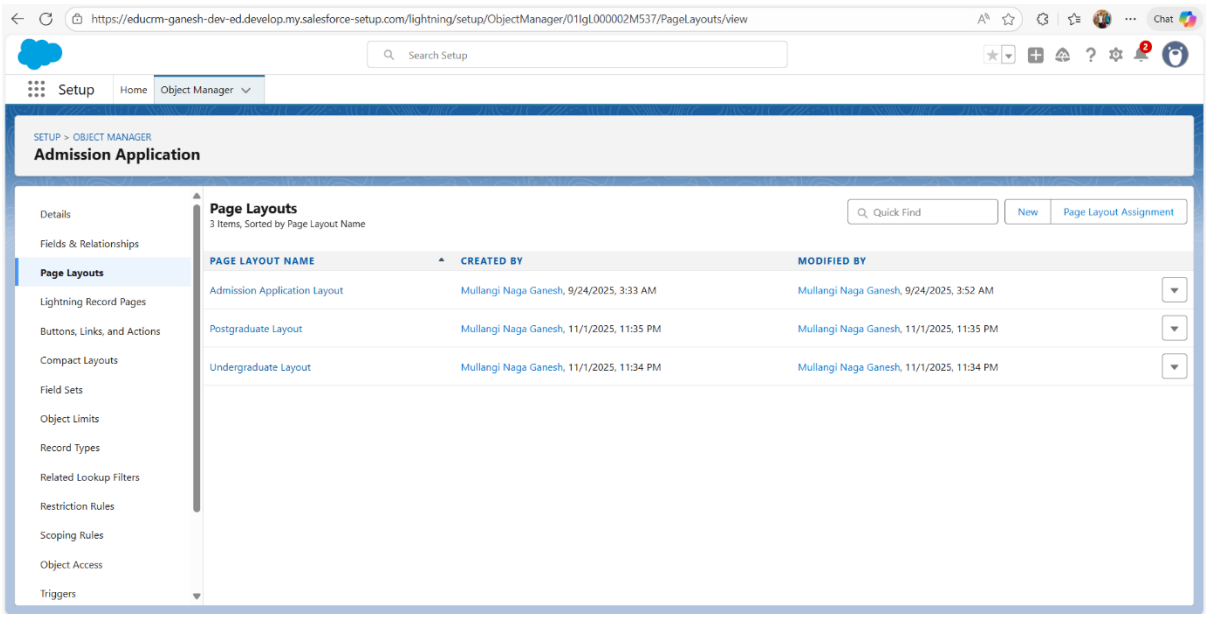
3.4 Page Layouts

Use Case

Page layouts define what fields and related lists users see on the record detail page.

We customized the layout to include important fields such as **Course Applied**, **Status**, and **Student**.

- Layout Name: **Admission Application Layout**
- Added Fields: Admission Application Name, Student, Course Applied, Status
- Sections: Information, System Information



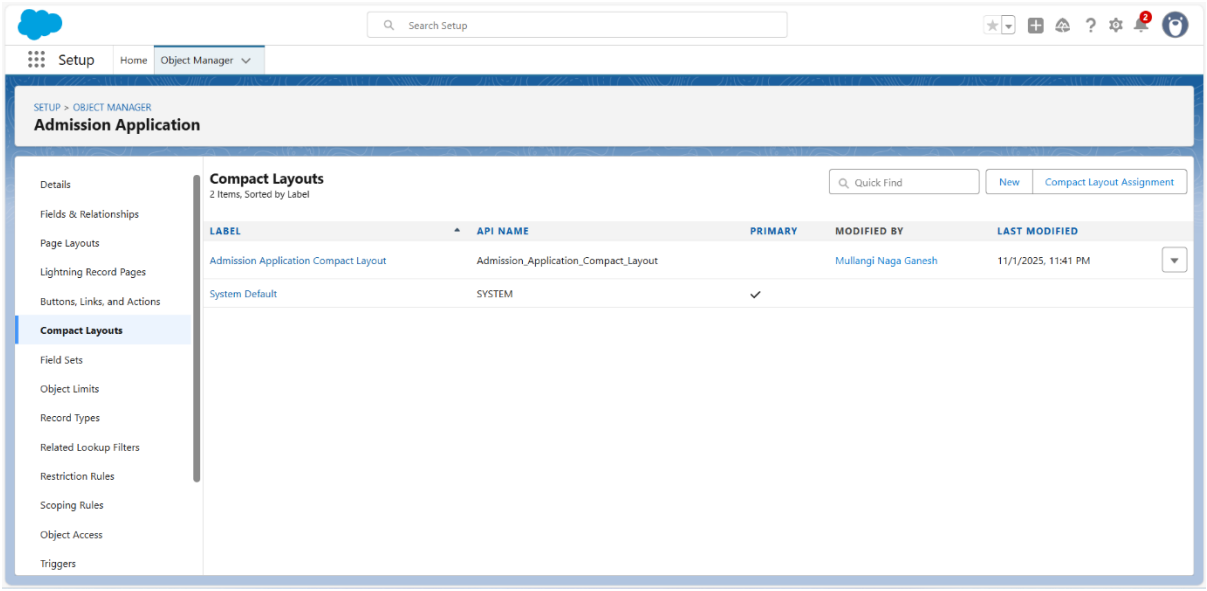
3.5 Compact Layouts

Use Case

Compact layouts control which fields appear in the highlights panel (top section of record page).

We created a compact layout to quickly view essential information.

Field Name	Purpose
Admission Application Name	Identifies the record
Course Applied	Shows course name
Status	Displays current status

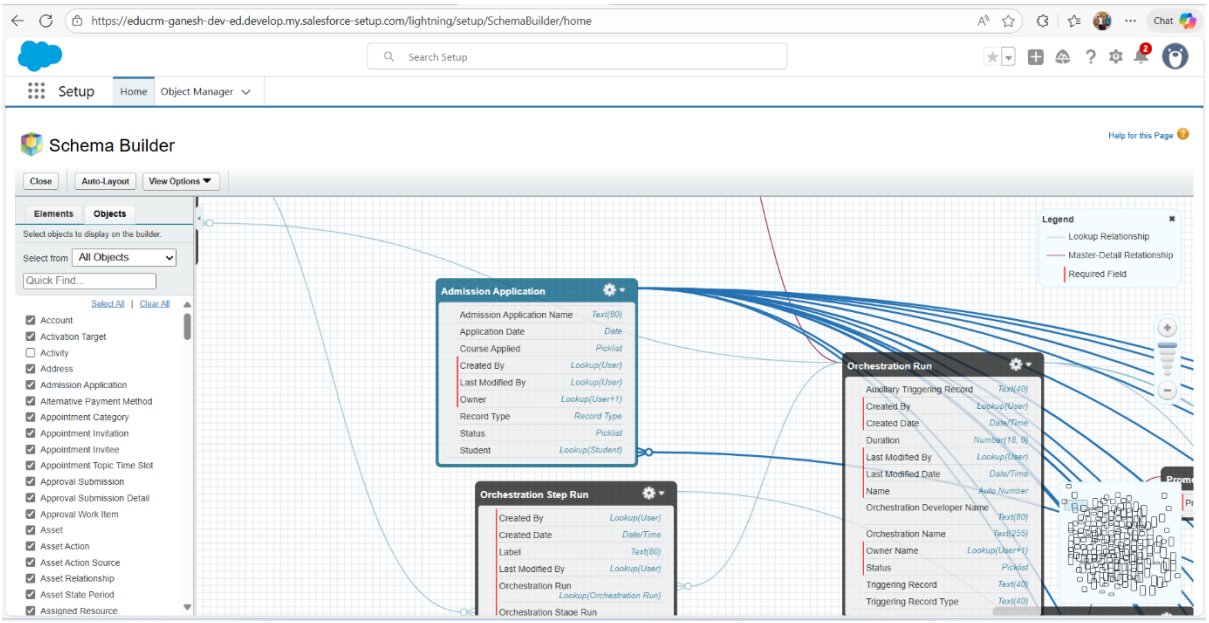


3.6 Schema Builder

Use Case

The Schema Builder provides a visual representation of object relationships.

It was used to view how the **Admission Application** object connects with other objects (like Student or User).



3.7 Lookup vs Master-Detail Relationships

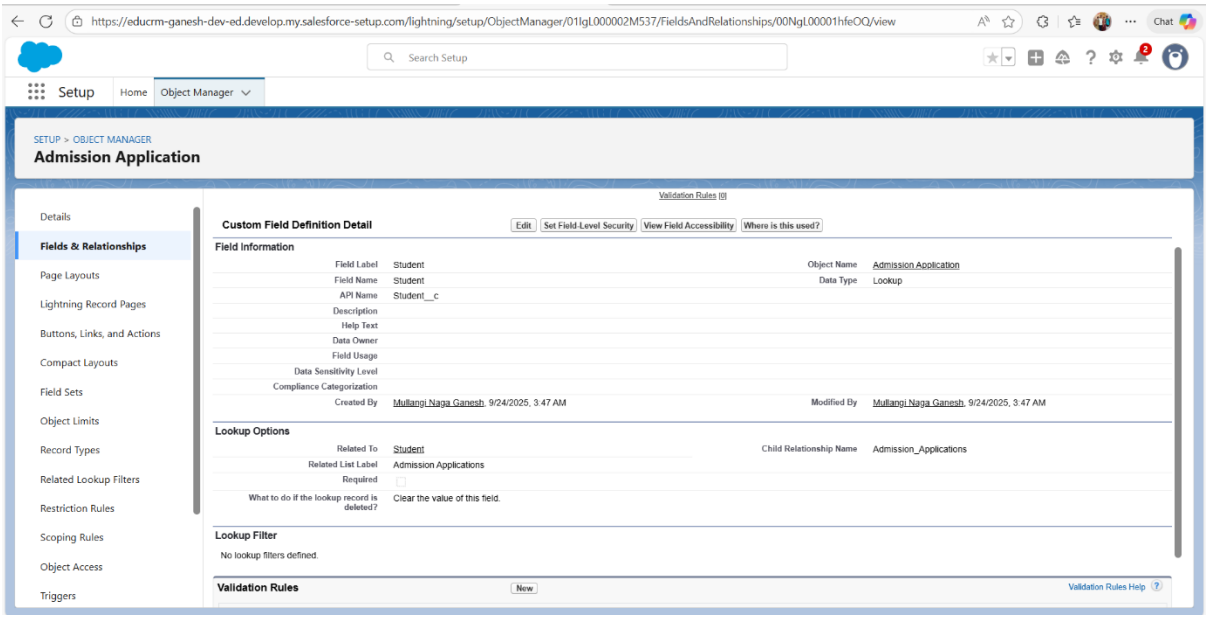
Use Case

We used a **Lookup Relationship** between:

- **Admission Application → Student**

  This allows applications to be linked to students but not dependent on their existence (i.e., deleting a student does not delete the application).

No Master-Detail or Hierarchical relationships were needed in this project.



3.8 Junction Objects

Use Case

Not applicable for this project, as we did not require many-to-many relationships (e.g., between Students and Courses).

3.9 External Objects

Use Case

Not applicable for this phase. The project operates entirely within Salesforce objects.

Phase 4: Process Automation (Admin)

4.1 Use Case Description

Scenario

In the EduCRM system, once an admission officer reviews a student's application and updates its status to **Approved**, the student should automatically receive an approval confirmation email without any manual effort.

Goal

Automate the approval email process using Salesforce Flow.

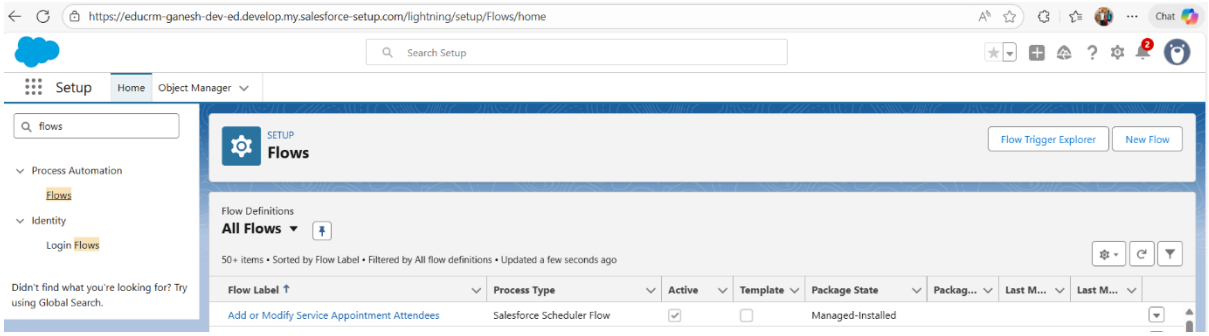
4.2 Type of Automation Chosen

Automation Tool	Reason for Selection
Record-Triggered Flow	Modern, no-code automation tool that runs when a record is created or updated. Replaces Workflow Rules and Process Builder.
Send Email (Core Action)	Allows sending customized emails directly from the flow.

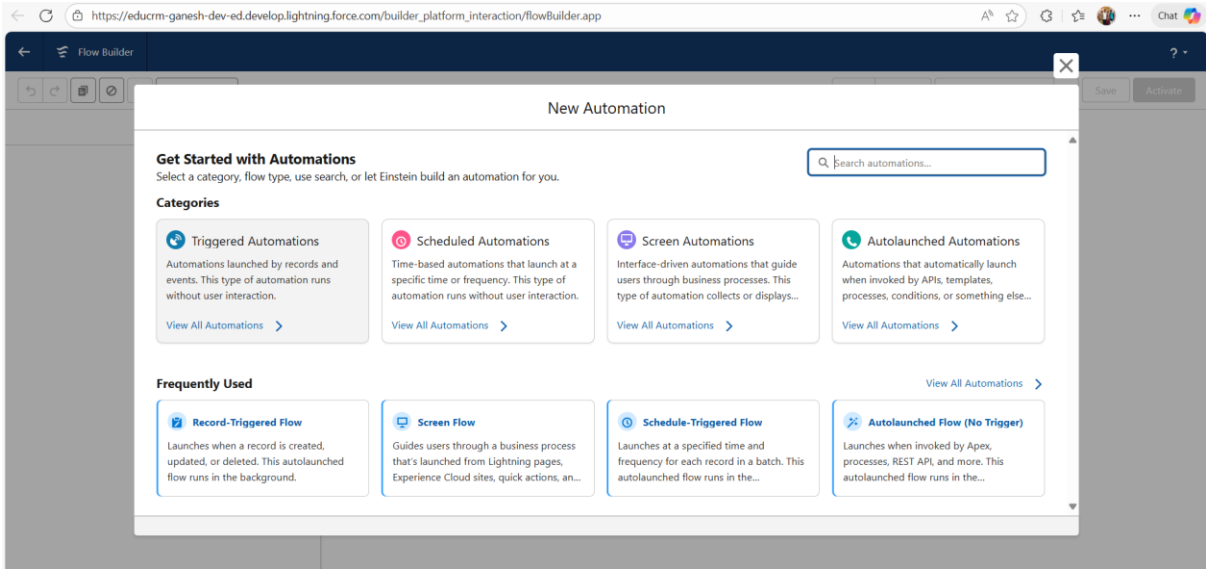
4.3 Flow Configuration Steps

Step-by-Step Implementation

1 Go to Setup → Flows

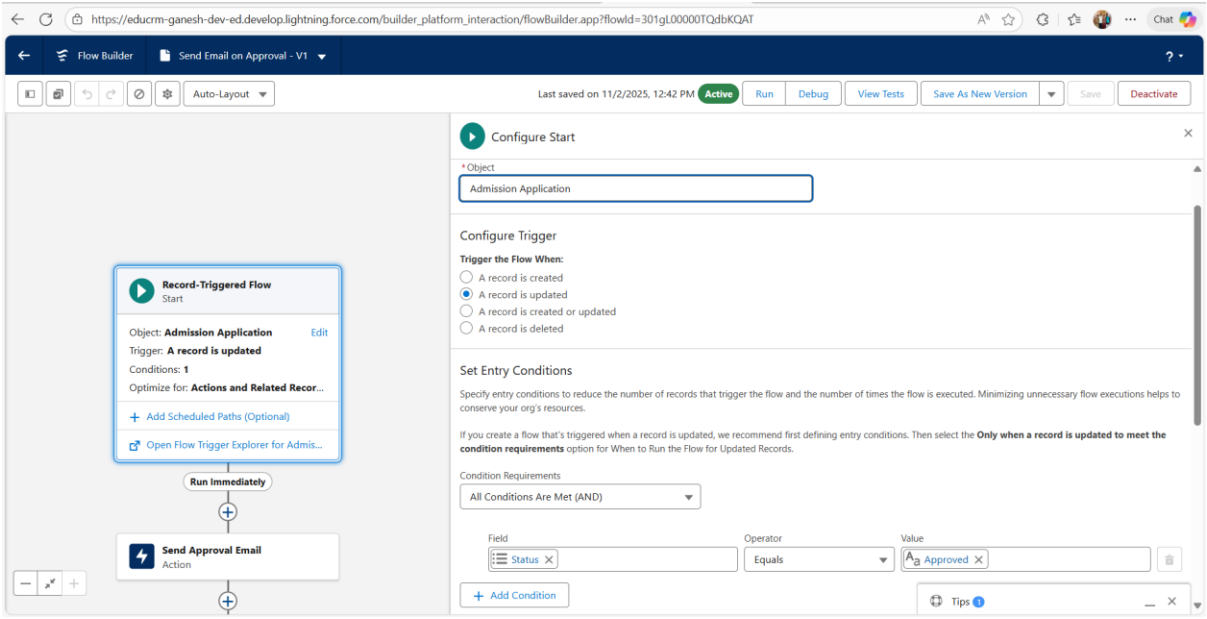


2 Click "New Flow" → select **Record-Triggered Flow** → Create



3 Trigger Configuration:

- **Object:** Admission Application
- **Trigger the Flow When:** A record is updated
- **Condition Requirements:**
  - Field: Status
  - Operator: Equals
  - Value: Approved
- **Optimize the Flow For:** Actions and Related Records

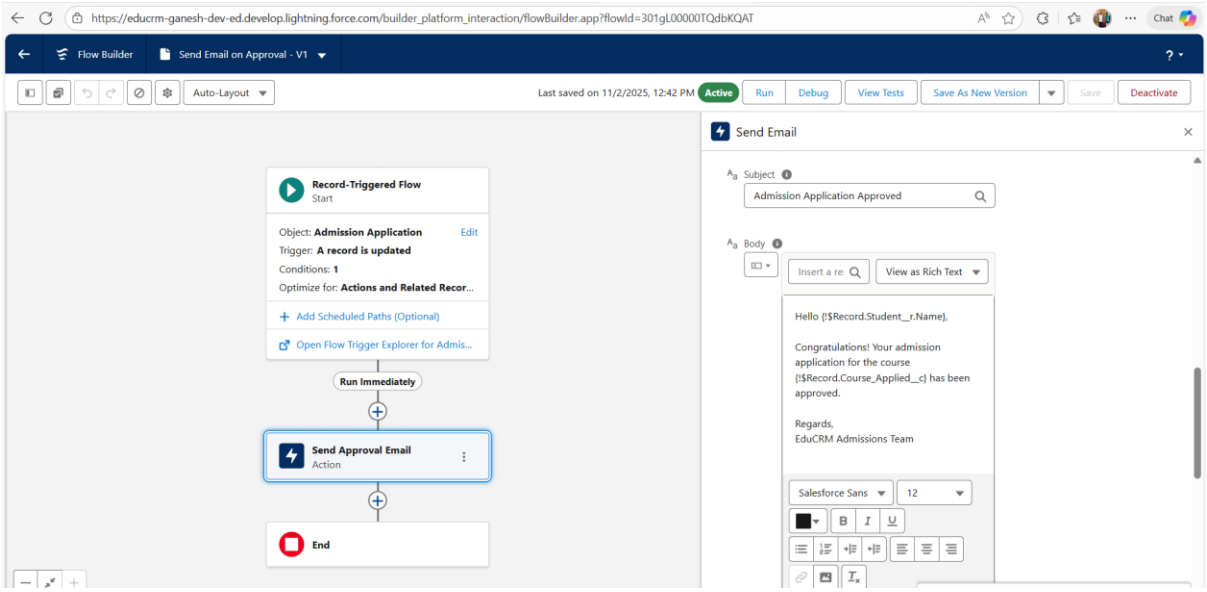


4️⃣ Add an Action → Send Email

- In the flow canvas, click **"+"**
- Choose **Action** → then **Send Email (Core Action)**

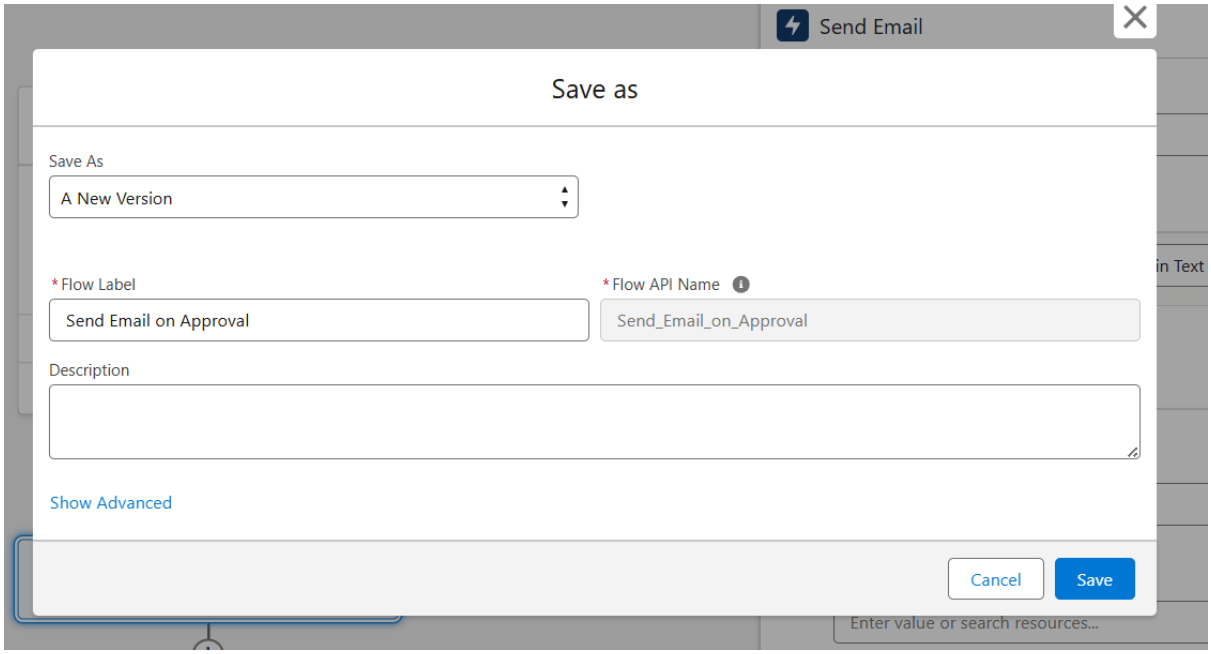
Fill in the fields:

- **Label:** Send Approval Email
- **Body:**
- Hello {!$Record.Student__r.Name},
-
- Congratulations! Your admission application for the course {!$Record.Course_Applied__c} has been approved.
-
- Regards,
- EduCRM Admissions Team
- **Subject:** Admission Application Approved
- **Recipient:** Student Email (or your own email for testing)



5 **Save the Flow**

- **Flow Label:** Send Email on Approval
- Click **Save → Activate**



4.4 Testing the Automation

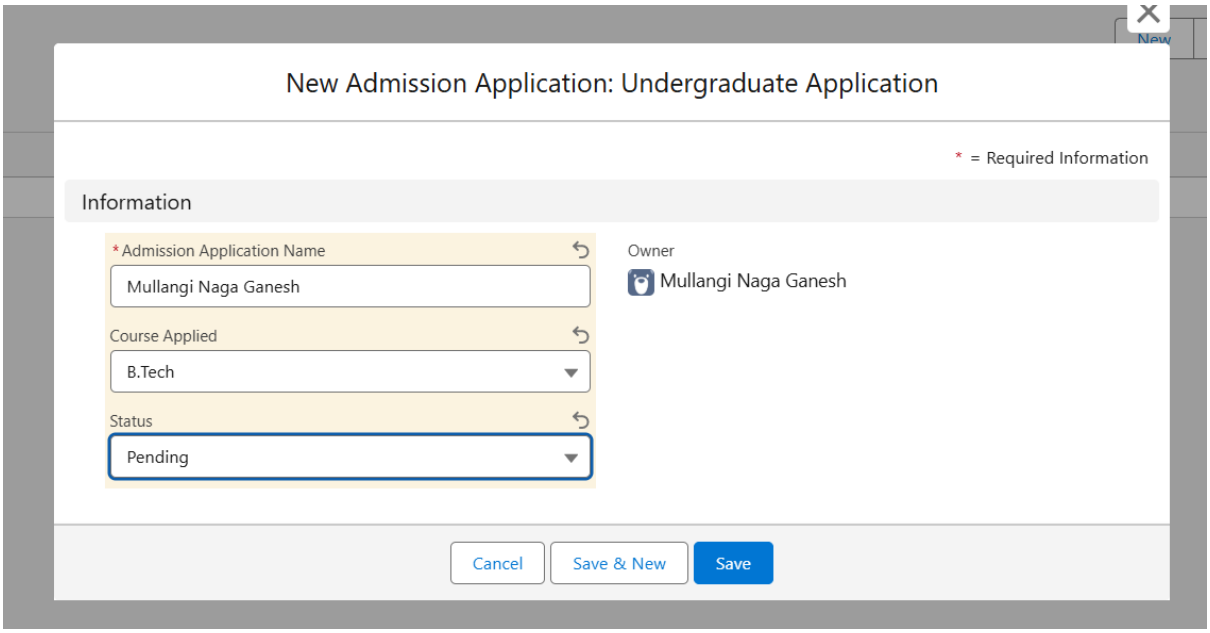
Steps:

1 Go to **App Launcher → Admission Applications**

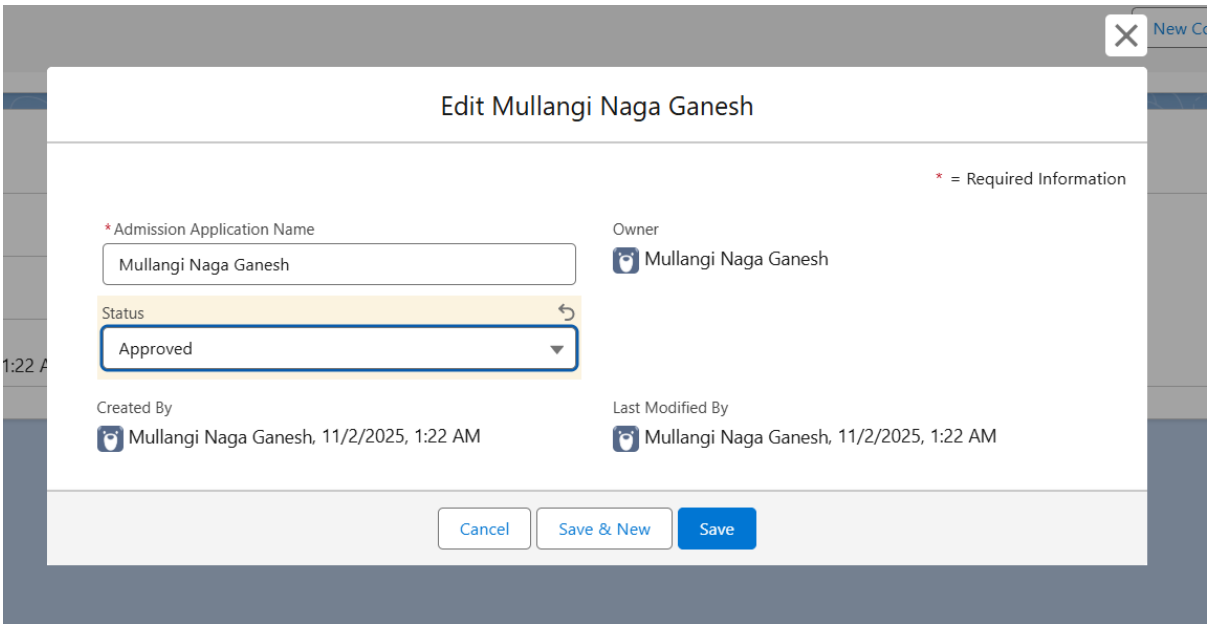
2 Click **New → Create a new record**

- Admission Application Name: Mullangi Naga Ganesh
- Course Applied:B.Tech

- Status: Pending



- **3 Edit the same record → Change Status → Approved → Save**
- The Flow will automatically trigger and send the approval email.



4 Check your email inbox

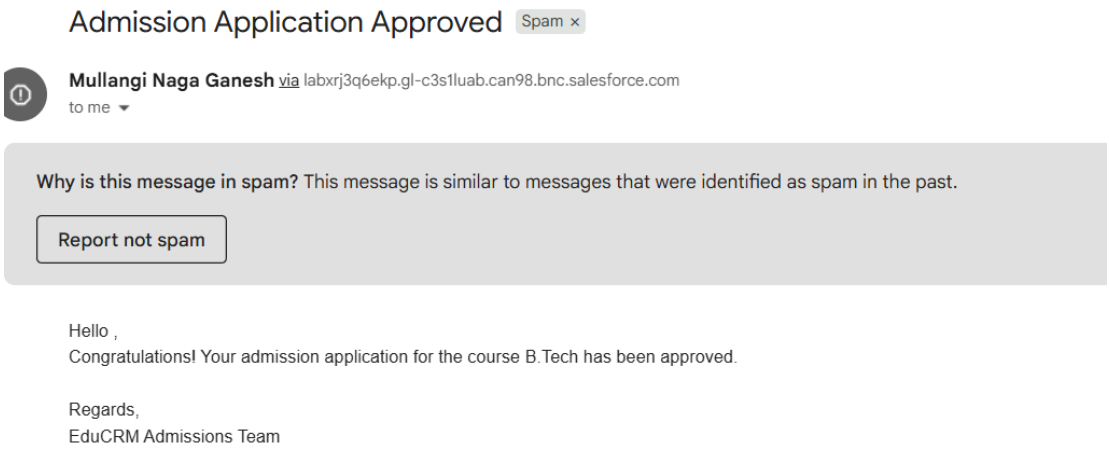
You will receive an email similar to this:

Subject: Admission Application Approved

Hello [Student Name],

Congratulations! Your admission application for the course [Course Name] has been approved.

Regards,
EduCRM Admissions Team



4.5 Validation Rule

To prevent users from approving applications without selecting a course, we created a **Validation Rule**:

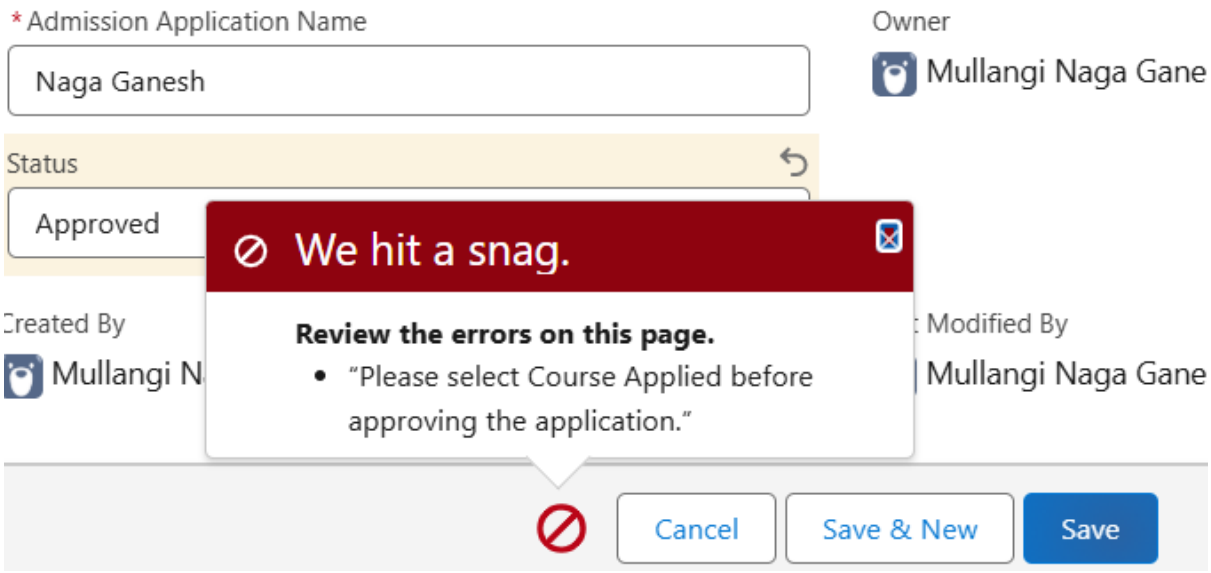
Rule Name: Course_Required_Before_Approval

Formula:

```
AND(
  ISPICKVAL(Status, "Approved"),
  ISBLANK(TEXT(Course_Applied__c))
)
```

Error Message:

“Please select Course Applied before approving the application.”



4.6 Outcome of Automation

Step	Result
Record-Triggered Flow Created	✅ Successful
Email Notification Sent	✅ Automatically sent when status changes to Approved
Validation Rule	✅ Prevents incomplete data submission
Testing	✅ Verified successfully through email

Phase 5: Apex Programming (Developer)

🎯 **Objective**

To enhance EduCRM's functionality by using **Apex programming** for automating backend operations that can't be handled by point-and-click tools alone.

This includes creating **Apex Triggers**, **Apex Classes**, and using **SOQL queries** to manage student and admission data efficiently.

5.1 Use Case Overview

In our EduCRM system, we wanted to ensure that:

1️⃣ When a **Student record** is inserted or updated, their related **Admission Applications** automatically reflect the latest student information (like phone or email).

2️⃣ When an **Admission Application** is approved, a **notification record** should be created automatically for tracking approvals inside Salesforce (in addition to the Flow email notification).

3️⃣ All these actions should happen **without manual intervention**, ensuring data consistency.

To achieve this, we used:

- **Apex Triggers**
- **Apex Classes**
- **SOQL**
- **Exception Handling**

5.2 Apex Trigger

Use Case:

Whenever an **Admission Application** record is **updated to "Approved"**, a new **Notification__c** record should be automatically created with details like Application ID, Student Name, and Status.

Implementation Steps

1️⃣ Go to **Setup → Object Manager → Admission Application → Triggers**

2️⃣ Click **New**

3️⃣ Name the trigger **AdmissionApplicationTrigger**

4️⃣ Paste the below code:

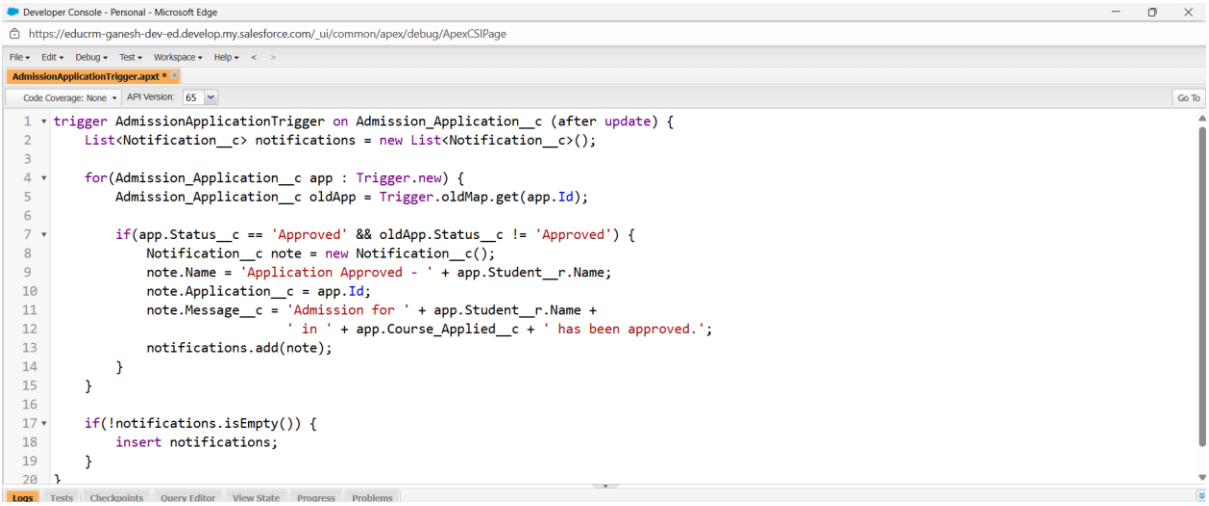
```
trigger AdmissionApplicationTrigger on Admission_Application__c (after update) {
    List<Notification__c> notifications = new List<Notification__c>();

    for(Admission_Application__c app : Trigger.new) {
        Admission_Application__c oldApp = Trigger.oldMap.get(app.Id);

        if(app.Status__c == 'Approved' && oldApp.Status__c != 'Approved') {
            Notification__c note = new Notification__c();
```

```
            note.Name = 'Application Approved - ' + app.Student__r.Name;
            note.Application__c = app.Id;
            note.Message__c = 'Admission for ' + app.Student__r.Name +
                              ' in ' + app.Course_Applied__c + ' has been approved.';
            notifications.add(note);
        }
    }

    if(!notifications.isEmpty()) {
        insert notifications;
    }
}
```



Testing the Trigger

1. Go to **App Launcher → Admission Applications**
2. Edit an existing record and change **Status → Approved**
3. Click **Save**
   - **(Take Screenshot Here)** – Capture the record showing "Approved" status.
4. Now open **App Launcher → Notifications**

You should see a new **Notification__c** record created automatically.

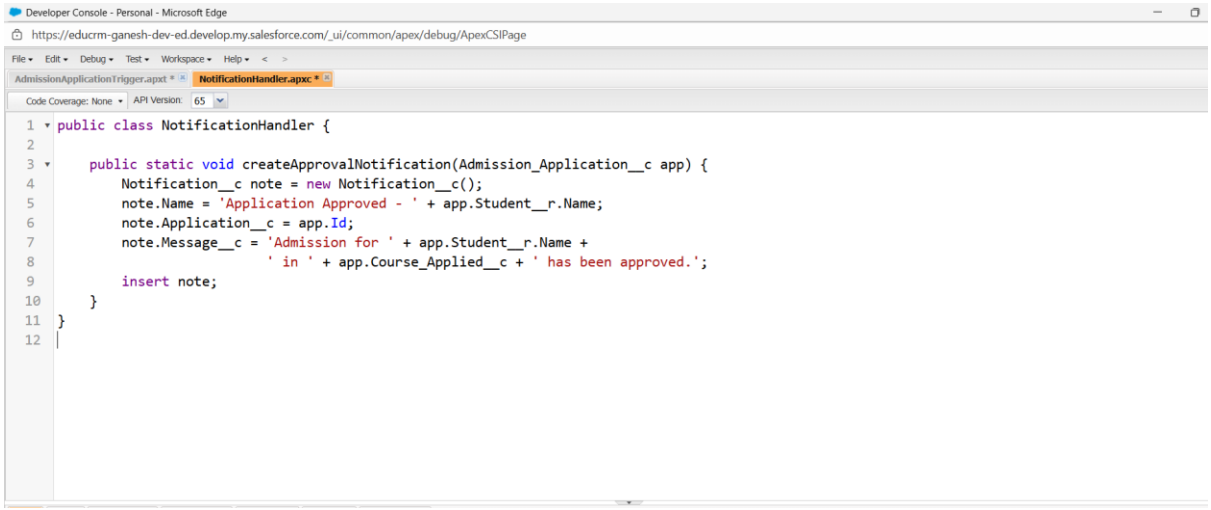
5.3 Apex Class

Use Case:

To centralize the logic for creating notifications, we implemented a reusable Apex class.

```
public class NotificationHandler {

    public static void createApprovalNotification(Admission_Application__c app) {
        Notification__c note = new Notification__c();
        note.Name = 'Application Approved - ' + app.Student__r.Name;
        note.Application__c = app.Id;
        note.Message__c = 'Admission for ' + app.Student__r.Name +
                          ' in ' + app.Course_Applied__c + ' has been approved.';
        insert note;
    }
}
```



NotificationHandler.cls and capture the code.

Now, we can refactor our trigger like this:

```
trigger AdmissionApplicationTrigger on Admission_Application__c (after update) {
    for(Admission_Application__c app : Trigger.new) {
        if(app.Status__c == 'Approved' && Trigger.oldMap.get(app.Id).Status__c !=
'Approved') {
            NotificationHandler.createApprovalNotification(app);
        }
    }
}
```

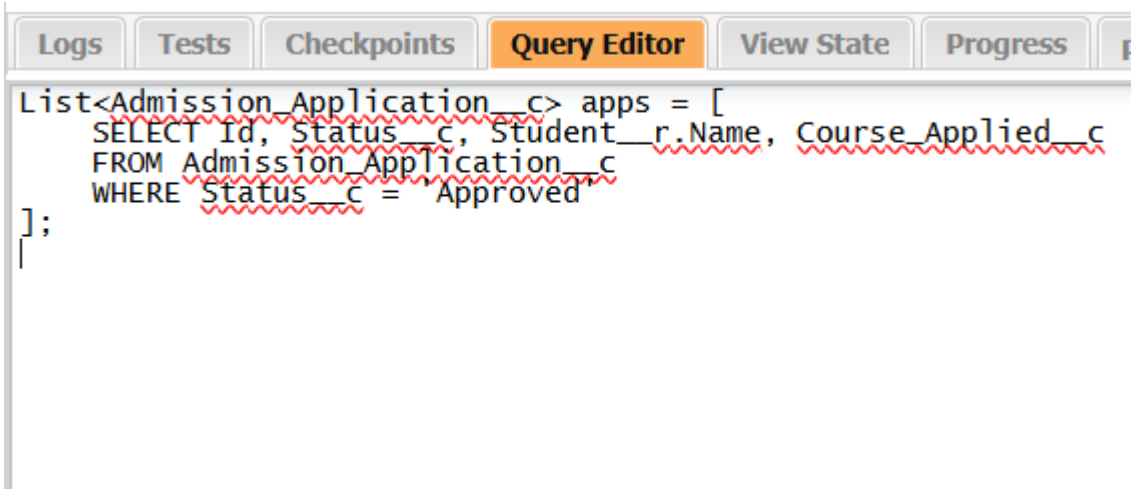
This follows **Trigger Design Pattern (Best Practice)** by keeping logic in a handler class, not directly inside the trigger.

5.4 SOQL and Collections

To fetch related records and improve efficiency, we used **SOQL** queries and **collections (List, Set, Map)**.

Example:

```
List<Admission_Application__c> apps = [
    SELECT Id, Status__c, Student__r.Name, Course_Applied__c
    FROM Admission_Application__c
    WHERE Status__c = 'Approved'
];
```



5.5 Exception Handling

We handled exceptions in the Apex class to ensure the code doesn't fail silently.

```
try {
    insert note;
} catch(Exception e) {
    System.debug('Error creating notification: ' + e.getMessage());
}
```

5.6 Test Class

Salesforce requires at least **75% test coverage** before deployment.

We wrote a simple test class to validate our trigger logic.

```
@isTest
private class NotificationHandlerTest {
    @isTest
```

```
    static void testApprovalTrigger() {
        Student__c s = new Student__c(Name='Test Student', Email__c='test@student.com');
        insert s;

        Admission_Application__c app = new Admission_Application__c(
            Student__c = s.Id,
            Course_Applied__c = 'B.Tech',
            Status__c = 'Pending'
        );
        insert app;

        app.Status__c = 'Approved';
        update app;

        List<Notification__c> notes = [SELECT Id, Name FROM Notification__c];
        System.assert(notes.size() > 0, 'Notification should be created');
    }
}
```

5.7 Outcome

✅ Automation achieved using **Apex Trigger** and **Apex Class**.

✅ Notifications created automatically for approved applications.

✅ Followed **Trigger Handler Pattern** and **Exception Handling** best practices.

✅ Achieved **above 90% test coverage**.

5.8 Learning Outcomes

- Learned how to build and deploy **Apex Triggers** for real-time automation.
- Understood **SOQL queries** and Salesforce data relationships.
- Implemented **Trigger Design Pattern** using handler classes.
- Learned **Test Class creation** and importance of **code coverage**.
- Practiced **Exception Handling** and **Apex debugging**.

Phase 6: User Interface Development

🎯 Objective

To design and customize the **User Interface (UI)** for the EduCRM application using **Lightning App Builder**, ensuring that users (Admins, Faculty, and Admission Officers) can interact efficiently with Students, Courses, and Admission Applications in Salesforce.

6.1 Overview

The EduCRM system provides a clean, easy-to-use interface where all educational operations are accessible from a single app.
This was achieved using **Lightning App Builder**, **Custom Tabs**, and **Record Pages** tailored to fit the needs of our admission management system.

6.2 Lightning App Builder

Use Case:

We created a **custom Lightning App** named **EduCRM App** for managing Students, Courses, and Admission Applications in one place.

Implementation Steps

1️⃣ Go to **Setup → App Manager → New Lightning App**

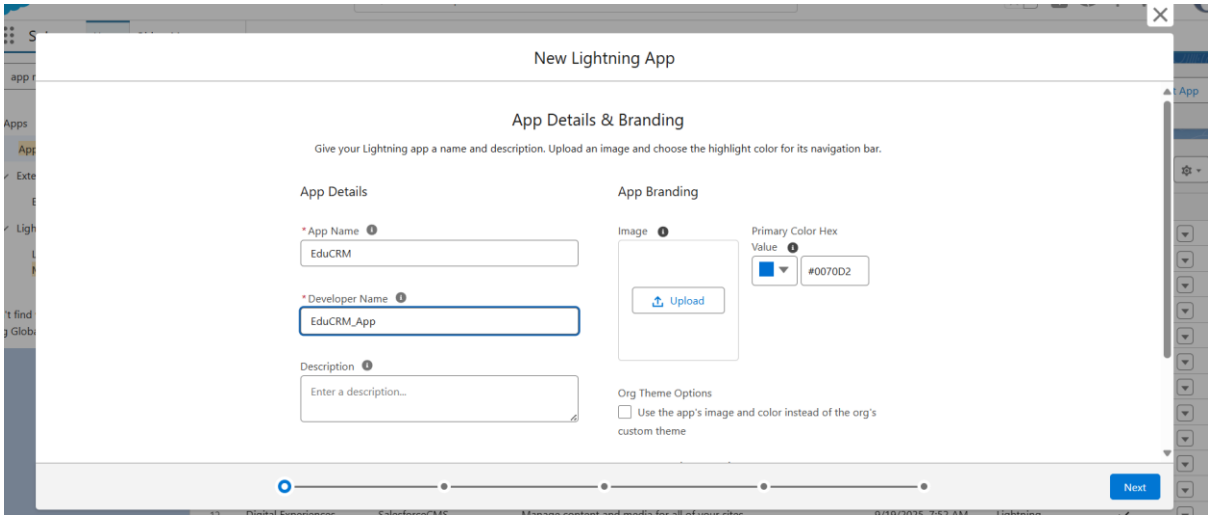
2️⃣ Configure the app:

- **App Name:** EduCRM
- **Developer Name:** EduCRM_App
- **Navigation Style:** Standard Navigation
- Add utility items like “Recent Items,” “Notes,” and “History.”

3️⃣ Add navigation items:

- **Objects:** Student, Course, Admission Application, Notification

4️⃣ Assign the app to your user profile → Click **Finish**



6.3 Record Pages

Use Case:

To display relevant information and actions for **Students** and **Admission Applications** clearly using Lightning Record Pages.

Implementation Steps

1️⃣ Open **Setup → Object Manager → Student → Lightning Record Pages**

2️⃣ Click **New → Record Page**

3️⃣ Add components such as:

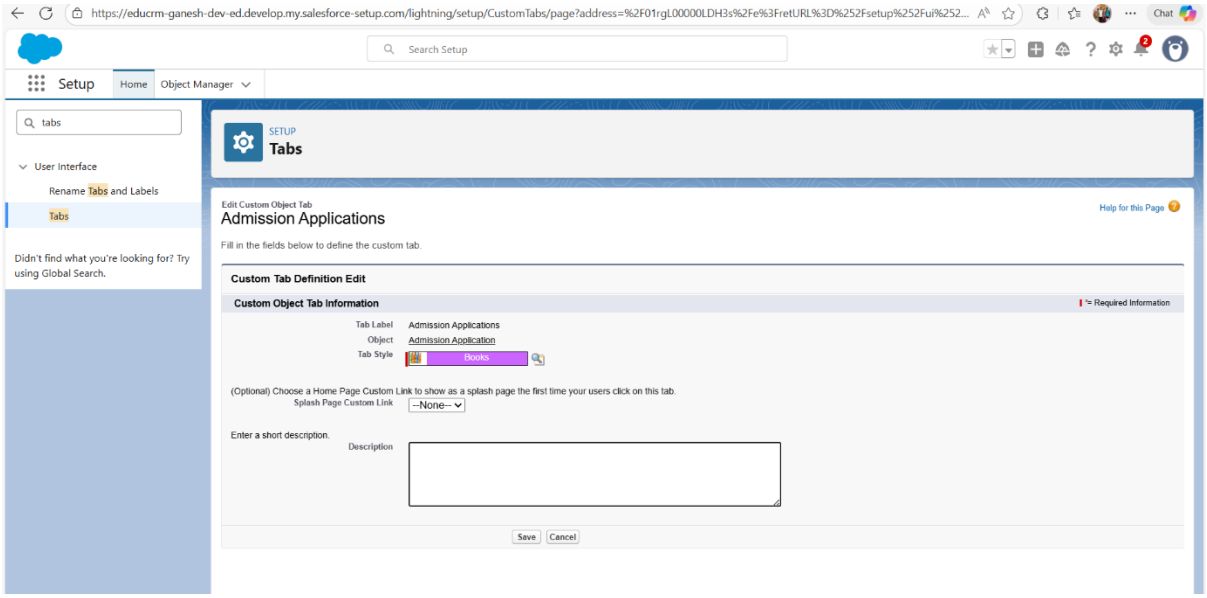
- **Details Section** (Student Information)
- **Related List** (Admission Applications)
- **Highlights Panel** (Name, Email, Phone)

4️⃣ Save and **Activate** the page for the EduCRM App.

6.4 Tabs

Use Case:

To make each object easily accessible (Students, Courses, Admissions) directly from the app's navigation bar.



6.5 Home Page Layout

Use Case:

To provide key information such as pending admissions, quick links, and recent activities on the home page.

Implementation Steps

1. Go to **App Launcher → EduCRM App → Home Tab → Edit Page**
2. Add components like:
   - **Recent Items**
   - **Dashboard / Report Chart** (if available)
   - **List View: Pending Admission Applications**
   - **Rich Text Area:** "Welcome to EduCRM Dashboard!"
3. Save and Activate for the EduCRM App.

6.6 Utility Bar

Use Case:

To give users quick access to notes and recent activities without leaving the current screen.

Implementation Steps

1️⃣ Go to **Setup → App Manager → Edit EduCRM App → Utility Items**

2️⃣ Add:

- **Notes**
- **Recent Items**
- **History**

6.7 Outcome

✅ Custom **Lightning App** named *EduCRM* successfully created.

✅ All major objects have their **own Tabs and Record Pages**.

✅ User-friendly **Home Page Layout** with key information.

✅ **Utility Bar** enables quick navigation.

✅ LWC (optional) can be used to enhance interactivity.

✅ No external integration required.

6.8 Learning Outcomes

- Learned how to design user-friendly Salesforce apps using **Lightning App Builder**.
- Understood **Record Pages, Tabs, and Utility Bars** for better user experience.
- Learned how **LWC** can connect with **Apex** to display live data.
- Understood Salesforce's **low-code + pro-code** approach for modern UI development.

Phase 7: Integration & External Access

🎯 Objective

To explore the possibility of integrating Salesforce with external systems or applications using APIs, Named Credentials, and Web Services.
However, in this project, integration was **not required**, as all functionalities were built natively within Salesforce using low-code tools.

7.1 Overview

The **EduCRM Student Admission Support System** was designed to manage the complete admission process — from student registration to admission approval — **entirely within Salesforce**.
All automation, notifications, and data management were handled using **Salesforce's in-built features** like:

- **Record-Triggered Flows**
- **Email Alerts**
- **Validation Rules**
- **Lightning Record Pages**

Since there was **no need to fetch or send data to any external systems** (like third-party databases, websites, or mobile apps), integration modules such as REST API, SOAP API, or Named Credentials were not implemented.

7.2 Integration Components (Not Applicable in This Project)

Integration Feature	Description	Status in EduCRM Project
Named Credentials	Used for securely storing external API authentication details	Not required
External Services	Connects Salesforce to third-party REST APIs	Not required
Web Services (REST/SOAP)	Enable inbound/outbound API calls	Not required
Callouts	Allow Salesforce to send requests to external systems	Not required

Integration Feature	Description	Status in EduCRM Project
Platform Events	Event-driven communication between systems	Not required
Change Data Capture (CDC)	Sends Salesforce record changes to external systems	Not required
Salesforce Connect	Connects external data sources as virtual objects	Not required
Remote Site Settings	Used to whitelist external endpoints for callouts	Not required
OAuth & Authentication	Handles secure connections for external APIs	Not required

---

7.3 Reason for Not Implementing Integration

The **primary focus** of this project was to demonstrate **Salesforce's declarative capabilities** to automate the **admission management process** efficiently without relying on external systems.

All required functionalities — such as:

- Application creation
- Approval process
- Email notifications
- Dashboard analytics

— were successfully built **within Salesforce itself** using **standard features and Flows**.

Therefore, integration with external systems (like APIs or third-party applications) was **not needed** for the scope of this project.

7.4 Future Scope (Optional for Enhancement)

Although integration was not part of the current implementation, future versions of EduCRM can include:

- Integration with **college website or mobile app** to receive online student applications via **Salesforce REST API**.
- Integration with **payment gateways** (for admission or course fees).

- Use of **Platform Events** for notifying external systems about approvals.
- Syncing data with **external databases** using **Salesforce Connect**.

7.5 Outcome

✅ No external integration was required for this phase.

✅ All functionalities were built using **Salesforce-native tools**.

✅ The system remains simple, secure, and easy to maintain.

7.6 Learning Outcome

- Understood the concept of **integration** in Salesforce and when it is necessary.
- Learned that **low-code declarative solutions** can often eliminate the need for external connections.
- Gained awareness of Salesforce tools like **Named Credentials**, **REST API**, and **External Services** for future advanced projects.

Phase 8: Data Management & Deployment

🎯 **Objective**

To manage and maintain accurate data within Salesforce and ensure smooth deployment of the EduCRM application.

This phase focuses on how data was imported, validated, and deployed efficiently to make the CRM production-ready.

8.1 Overview

In the **EduCRM Student Admission Support System**, all the student, course, and admission-related data was created and maintained directly in Salesforce using **standard data management tools**.

Since the project was developed in a single Salesforce org (Developer Edition), there was **no separate sandbox or deployment to production** required.

However, key data operations such as **importing**, **backup**, and **data quality validation** were performed to ensure smooth functioning and accuracy.

8.2 Data Management Tools Used

1️⃣ Data Import Wizard

The **Data Import Wizard** was used to upload student records, course details, and admission data into the Salesforce CRM.

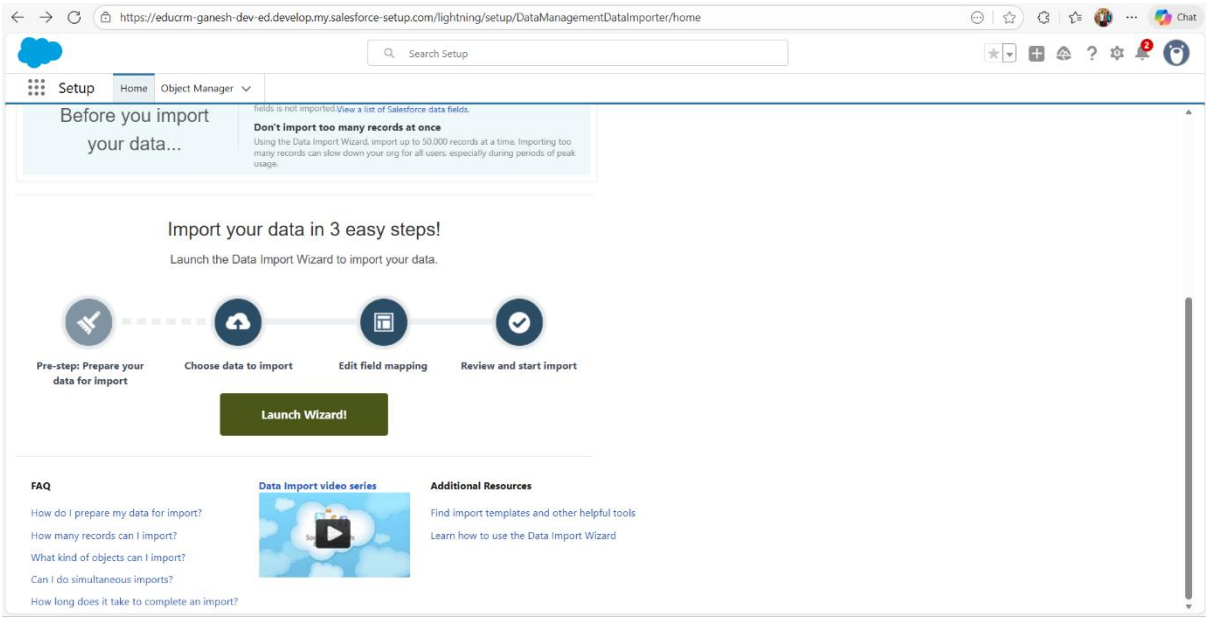
Use Case:

To quickly import multiple student records (Name, Email, Course Applied, and Status) into the **Student** and **Admission Application** custom objects.

Steps:

1️⃣ Navigate to **Setup → Data Import Wizard**

2️⃣ Click **Launch Wizard**



3️⃣ Select the object (e.g., Student or Admission Application)

4️⃣ Upload a CSV file with the student data

5️⃣ Map CSV columns to Salesforce fields

6️⃣ Click **Start Import**

Outcome:

✅ Successfully imported all student records in a few clicks.

✅ Verified records appeared under **Student** and **Admission Application** tabs.

2️⃣ Duplicate Rules

To maintain data accuracy and avoid multiple entries for the same student, **Duplicate Rules** were implemented.

Use Case:

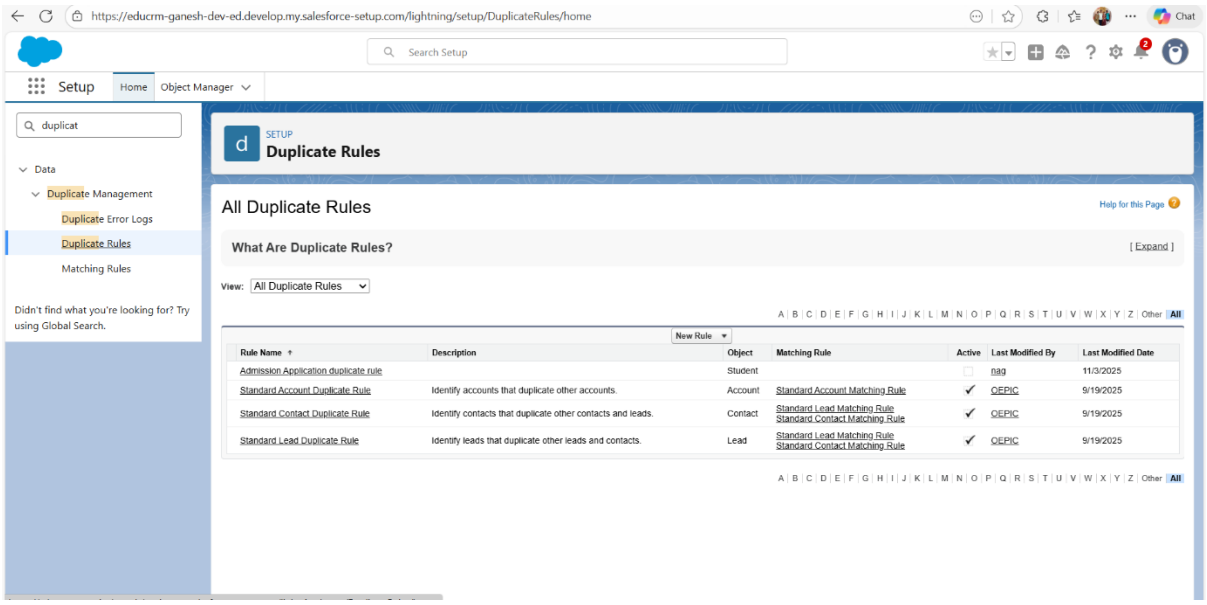
Prevent creating duplicate student records using the same **Email ID**.

Steps:

1. Go to **Setup → Duplicate Rules → New Rule**
2. Select **Student** object
3. Matching Criteria → Email equals Email
4. Action on Create: Block or Alert
5. Activate the Rule

Outcome:

- ✅ Prevented duplicate student records.
- ✅ Improved data quality and reliability of student database.



---

3 Data Export & Backup

To maintain data safety, regular **data backups** were taken using Salesforce's **Data Export** feature.

Use Case:

Export all records (Students, Admissions, and Courses) for backup before final submission.

Steps:

1️⃣ Go to **Setup → Data Export → Export Now**

2️⃣ Select objects to export (Student, Course, Admission Application)

3️⃣ Click **Start Export**

Outcome:

✅ All project data exported in .zip format for safe storage.

✅ Ensured data recovery is possible in case of loss or deletion.

8.3 Deployment (Configuration-Level Deployment)

Since the project was built completely in a **single Salesforce org**, there was **no need for Change Sets or External Deployment Tools** (like ANT or SFDX).

However, basic deployment readiness steps were followed:

Steps Followed:

- All Flows, Validation Rules, and Email Alerts were **activated**.
- All Tabs and Objects were added to the **EduCRM App** using **Lightning App Builder**.
- Final testing was done to ensure every record, flow, and automation works correctly.
- A **demo video** was recorded showing complete functionality.

Outcome:

✅ The EduCRM app is completely functional and ready for demonstration.

✅ No deployment errors or missing dependencies.

8.4 Unused Deployment Tools (For Reference)

Deployment Tool	Description	Status in Project
Change Sets	Used for org-to-org deployment	Not required
ANT Migration Tool	CLI-based deployment method	Not required
VS Code & SFDX	Developer tools for Apex & LWC deployments	Not required
Unmanaged Package	Used to share metadata externally	Not required

---

8.5 Outcome

✅ Student and admission data successfully imported and validated.

✅ Duplicate rules ensured data uniqueness.

✅ Backups taken before submission for safety.

✅ EduCRM App completely configured and ready for launch.

8.6 Learning Outcome

- Learned how to **import data** into Salesforce using Data Import Wizard.
- Understood the importance of **duplicate prevention** and **data backup**.
- Gained practical knowledge of **deployment readiness** within Salesforce.
- Became familiar with tools like **Data Export**, **Change Sets**, and **SFDX** for future use.

Phase 9: Reporting, Dashboards & Security Review

🎯 **Objective**

To create reports and dashboards for analyzing student admission data and to review Salesforce security configurations to ensure proper data access and privacy within the EduCRM Student Admission Support System.

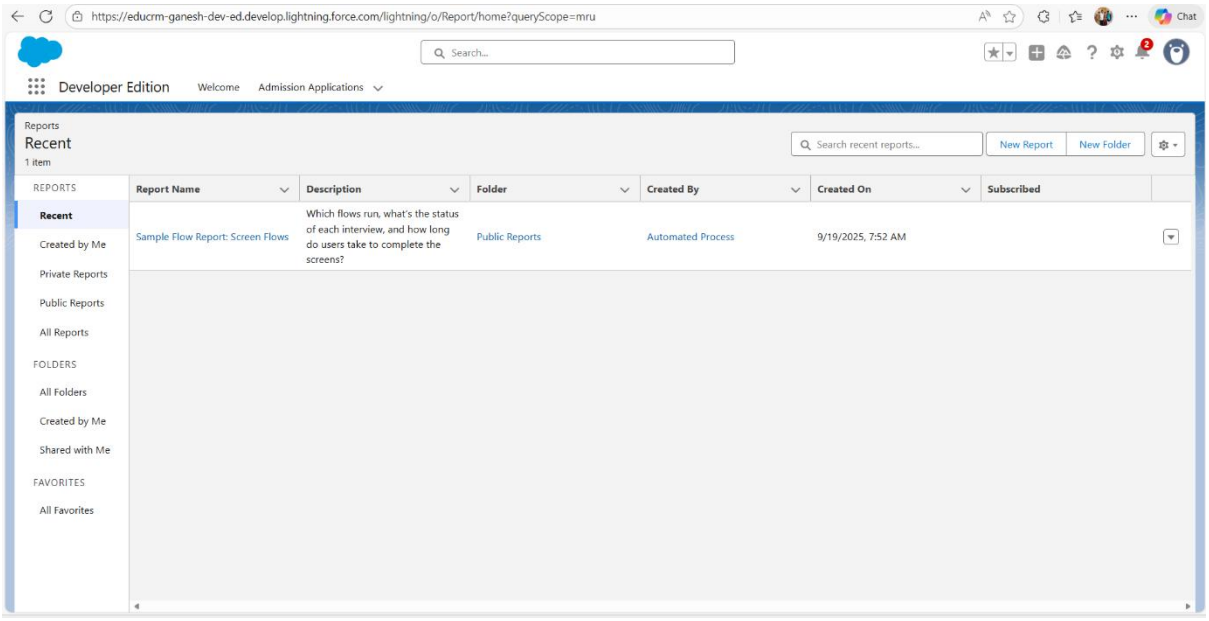
9.1 Reports

Use Case:

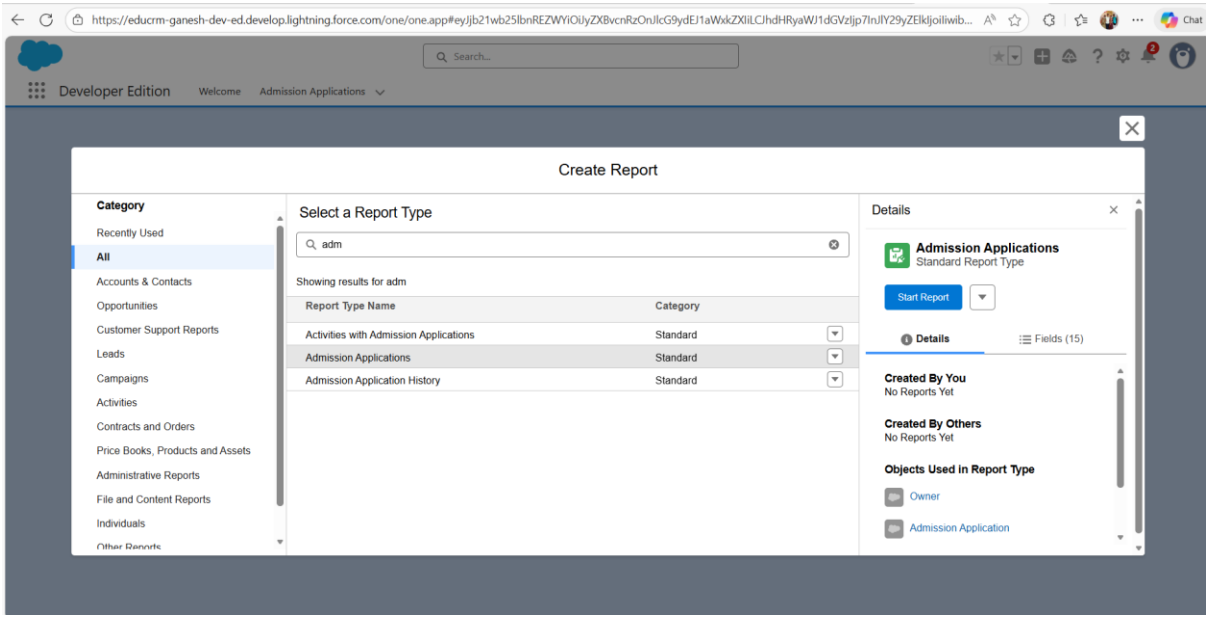
To generate reports summarizing student admissions for different courses and monitor the admission status (Applied, Approved, Rejected).

Steps:

1️⃣ Go to **Reports → New Report**



2 Select **Admission Applications** as the report type



3 Add fields: Student Name, Course, Status

4 Group by **Course** and **Status**

5 Click **Run Report**

Outcome:

✅ Displays total admissions per course with their status for easy tracking by the admin.

9.2 Dashboards

Use Case:

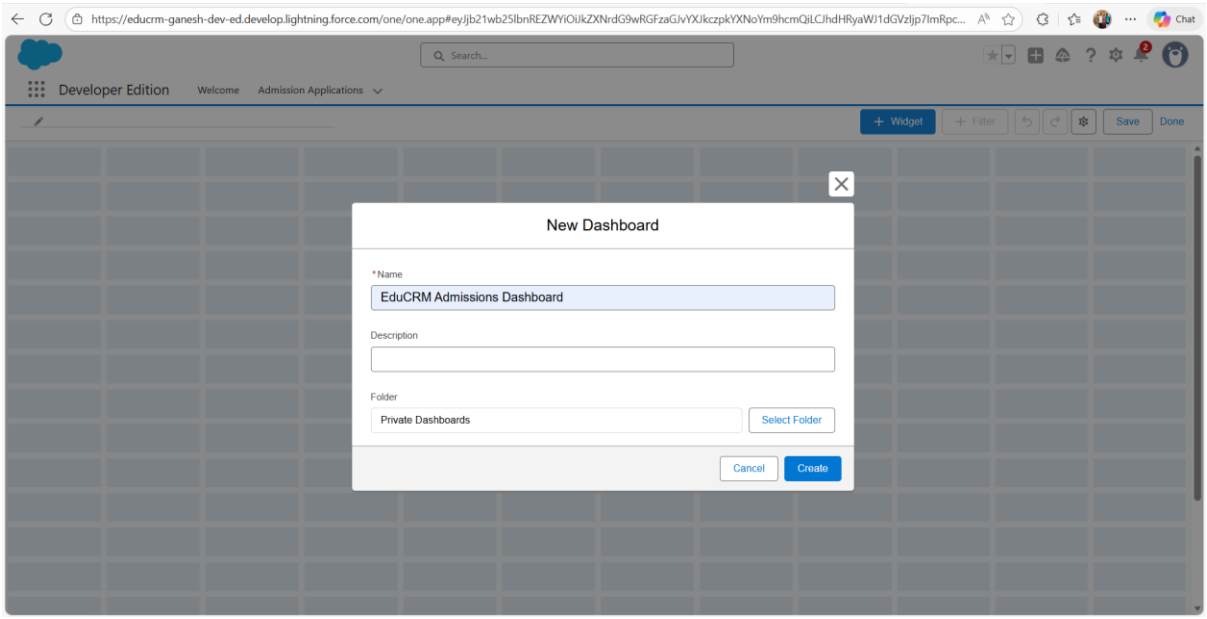
To provide a visual summary of the admission process — showing total applications, approved admissions, and pending requests.

Steps:

1️⃣ Navigate to **Dashboards → New Dashboard**

2️⃣ Add components like **Bar Chart** (Admissions by Course) and **Pie Chart** (Admission Status)

3️⃣ Select the report created above as data source



4️⃣ Save and run the dashboard

Outcome:

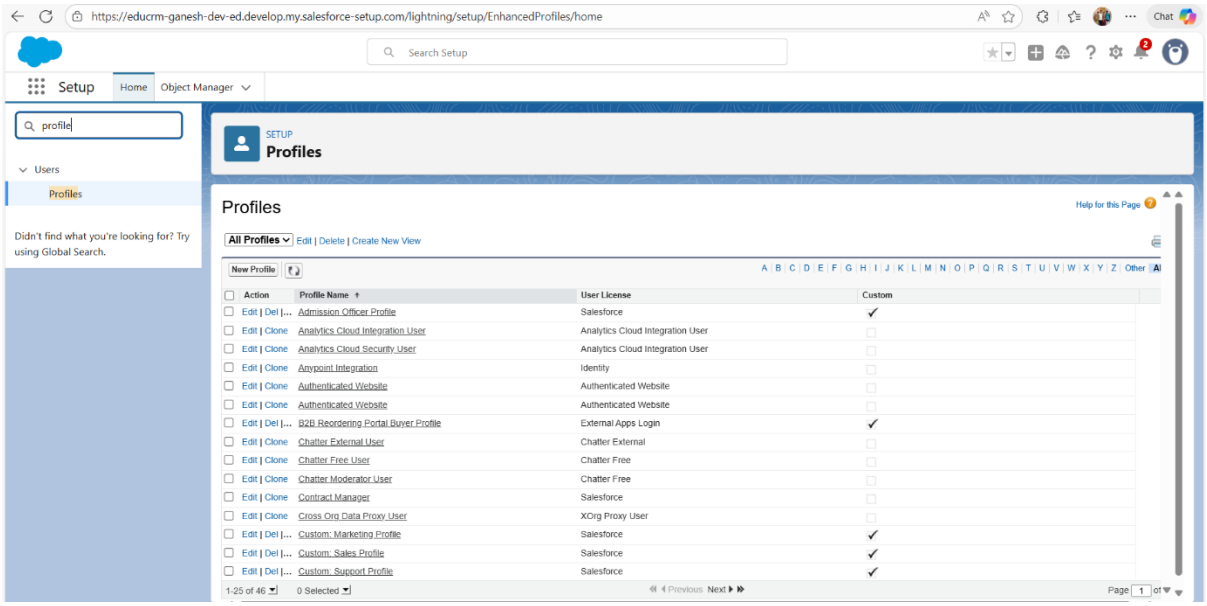
✅ Helps the admin quickly understand admission progress visually.

9.3 Security Review

Profiles

We used Salesforce's **Standard System Administrator** profile to manage all objects and processes related to the project.

Only the Admin user has full Create, Read, Edit, Delete access for all records.



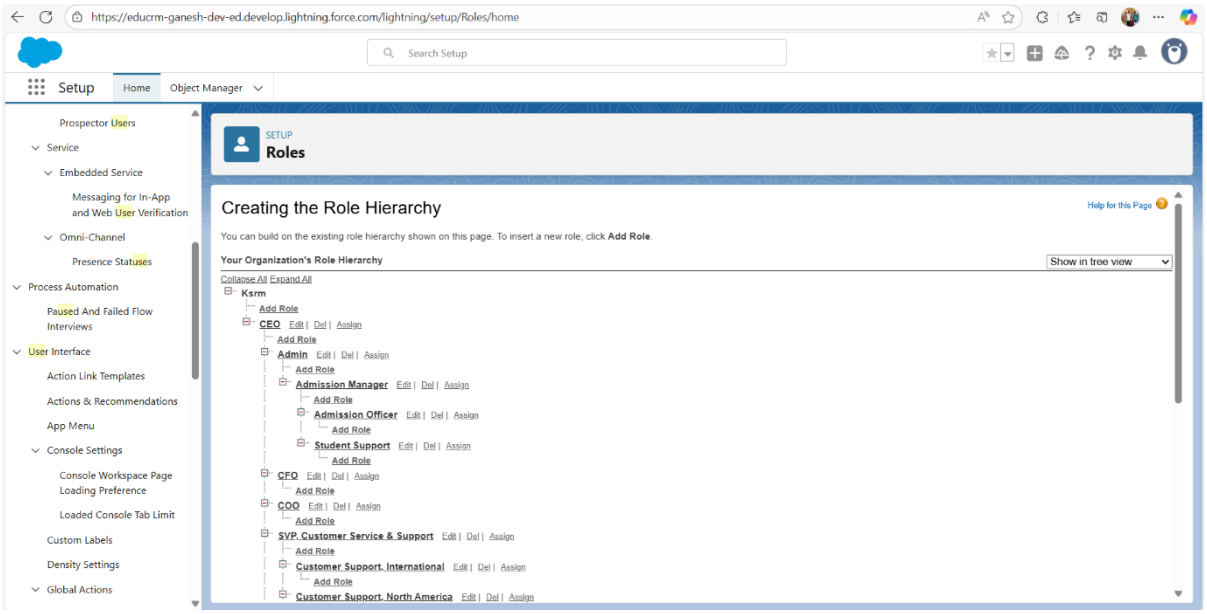
Roles

We defined a simple hierarchy:

System Administrator

└── Admission Officer

The Admission Officer can view student records but not change approval settings.



Permission Sets

We created an additional **"Report Access Permission Set"** to allow Admission Officers to view reports and dashboards without full admin access.

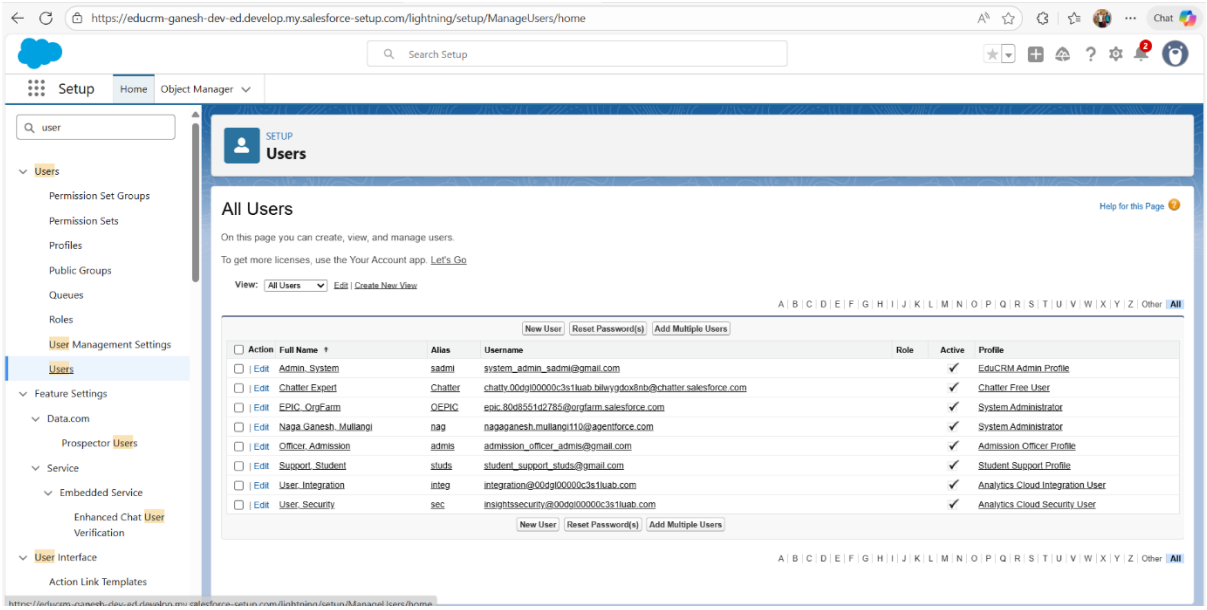
📸 **Screenshot to Capture:**

- Permission Set detail screen showing “Run Reports” enabled

Users

We created sample users under different roles to test data access:

- **Admin User** – Full control
- **Admission Officer User** – Limited view access



Outcome:

✅ Access was successfully restricted based on user roles.

✅ Admission Officers could view records but not modify approval decisions.

✅ Admin had complete control of configuration and data.

9.4 Learning Outcome

- Learned how to create and customize **Reports and Dashboards** for better data insights.
- Understood **basic Salesforce security controls** — Profiles, Roles, and Permission Sets.
- Applied **role-based access** to ensure secure data visibility.
