

# Admissions-Elevate-CRM Project – Full Explanation

---

## Phase 1: Problem Understanding & Industry Analysis

- **Goal:** Understand what we're building and why.

### 1. Requirement Gathering

- Track prospective students from their first inquiry through to the final admission decision.
- Manage a formal, multi-stage application review process (e.g., Submitted, Under Review, Accepted).
- Automate key communications, such as sending an email to confirm an application has been received.
- Provide a central place to view all applicant data and related documents.

### 2. Stakeholder Analysis

- **Admissions Director (Manager):** Needs to monitor application volumes and team performance through reports and dashboards.
- **Admissions Officers (End Users):** Need an efficient system to manage their assigned applicants and review application details.
- **Faculty Reviewers (Specialized Users):** Need limited access to only see and provide feedback on applications assigned to them.

### 3. Business Process Mapping

- **Current Process:** A manual system using spreadsheets to track applicants and email to communicate. This is slow, prone to data entry errors, and provides no clear visibility into the admissions pipeline.
- **Future Process:** A prospective student (Lead) is captured from the university website. When they apply, an Application record is created. This record moves through a structured review process, automatically notifying the applicant of status changes.

### 4. Industry-specific Use Case Analysis

- In higher education, application deadlines are strict, and different academic programs often have unique requirements. The system must be able to manage these complexities and ensure all required documents are submitted and verified.

### 5. AppExchange Exploration

- Researched form-building tools like **FormAssembly** and document generation apps like **Conga**.

- **Decision:** To ensure the project is completely **cost-free**, all functionality will be built using native Salesforce features.

## Phase 2: Org Setup & Configuration

- **Goal:** Prepare the foundational Salesforce environment for the application.

### 1. Salesforce Editions

- A free **Developer Edition Org** has been created and is the active environment for all development.

### 2. Company Profile Setup

- The university's core information, including name, address, default time zone, and primary language, has been configured in Setup > Company Information.

The screenshot shows the Salesforce Setup interface for the 'Company Information' section. The organization is 'Gyan Ganga Institute of Technology and Sciences'. The page is divided into two main sections: 'Organization Detail' and 'System Settings'. The 'Organization Detail' section includes fields for Organization Name, Primary Contact, Division, Address, Phone, Fax, Default Locale, and Default Language. The 'System Settings' section includes fields for Fiscal Year, Currency, Time Zone, and various system options like 'Enable Data Translation', 'Newsletters', and 'API Requests'. The page is created by 'OrgFarm EPIC' on 7/17/2025, 9:28 AM and modified by 'Vidhi Yadav' on 9/14/2025, 7:03 AM.

Organization Detail	
Organization Name	Gyan Ganga Institute of Technology and Sciences
Primary Contact	OrgFarm EPIC
Division	
Address	Tikara Ghat, near Bangi Hills, Bangi Hills, Jabalpur, Madhya Pradesh 482008 Jabalpur 482008 Madhya Pradesh India
Phone	(545) 746-6941
Fax	
Default Locale	English (India)
Default Language	English
Fiscal Year Starts In	January
Activate Multiple Currencies	
Enable Data Translation	
Newsletters	
Admin Newsletters	
Hide Notices About System Maintenance	
Hide Notices About System Downtime	
Locale Formats	ICU
Default Time Zone	(GMT+05:30) India Standard Time (Asia/Kolkata)
Currency Locale	English (India) - INR
Used Data Space	456 KB (9%) [View]
Used File Space	58 KB (0%) [View]
API Requests, Last 24 Hours	0 (15,000 max)
Streaming API Events, Last 24 Hours	0 (10,000 max)
Restricted Logins, Current Month	0 (0 max)
Salesforce.com Organization ID	00DgK000007Zyvf
Organization Edition	Developer Edition
Instance	CAN96

Created By: OrgFarm EPIC 7/17/2025, 9:28 AM  
Modified By: Vidhi Yadav 9/14/2025, 7:03 AM

### 3. Business Hours & Holidays

- The admissions office's standard operating hours (e.g., 9:00 AM - 5:00 PM, Monday-Friday) and key university holidays have been set.

### 4. Fiscal Year Settings

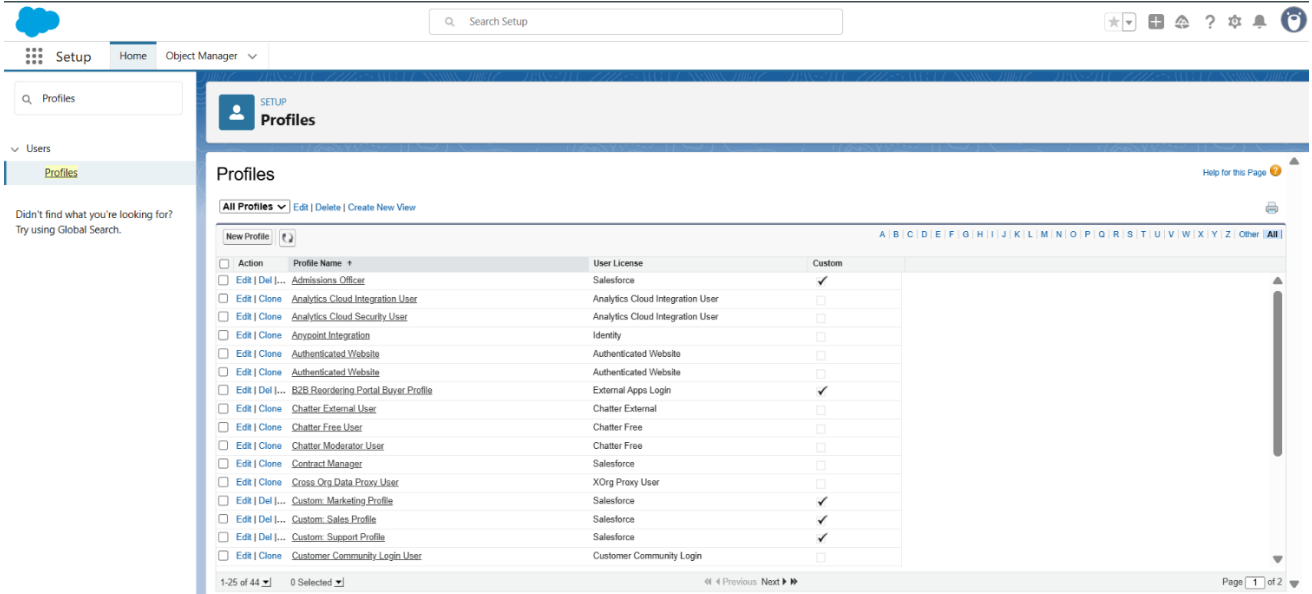
- A **Standard Fiscal Year** beginning in January has been configured for reporting purposes..

### 5. User Setup & Licenses

- Example user records for the primary roles (Admissions Director, Admissions Officer) have been created and assigned **Salesforce** user licenses..

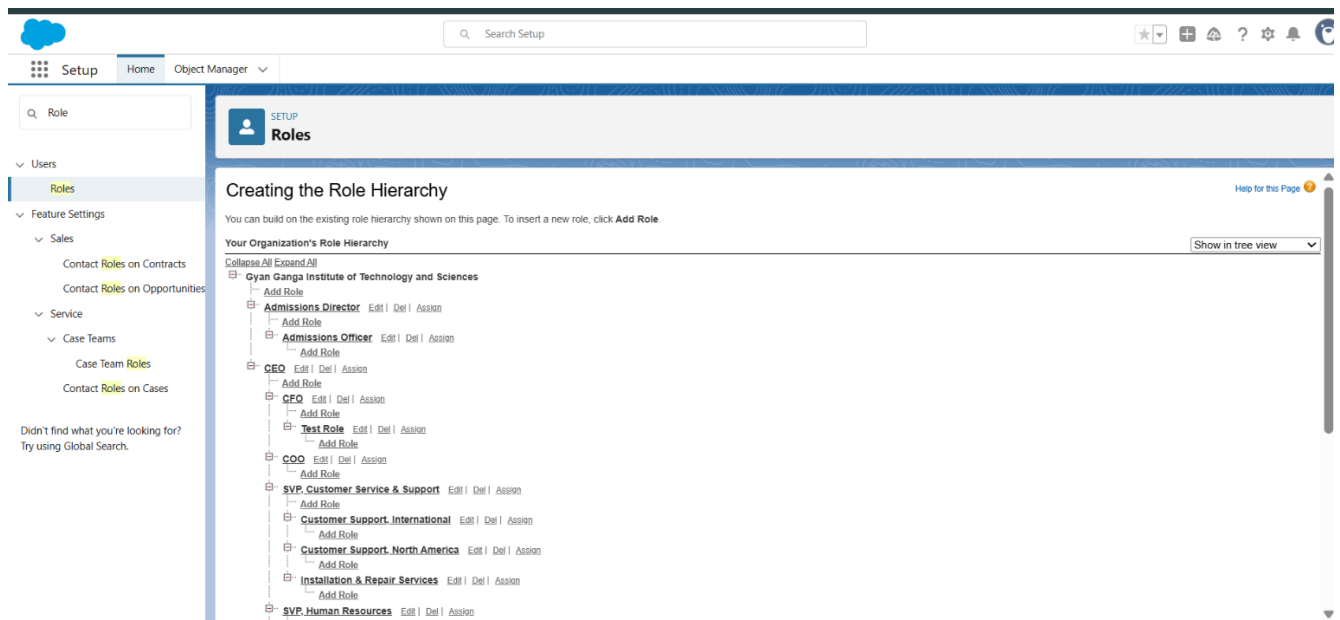
## 6. Profiles

- A custom profile named "**Admissions Officer**" has been cloned from a base profile to provide specific, limited access to the system. The System Administrator profile is assigned to the Director.



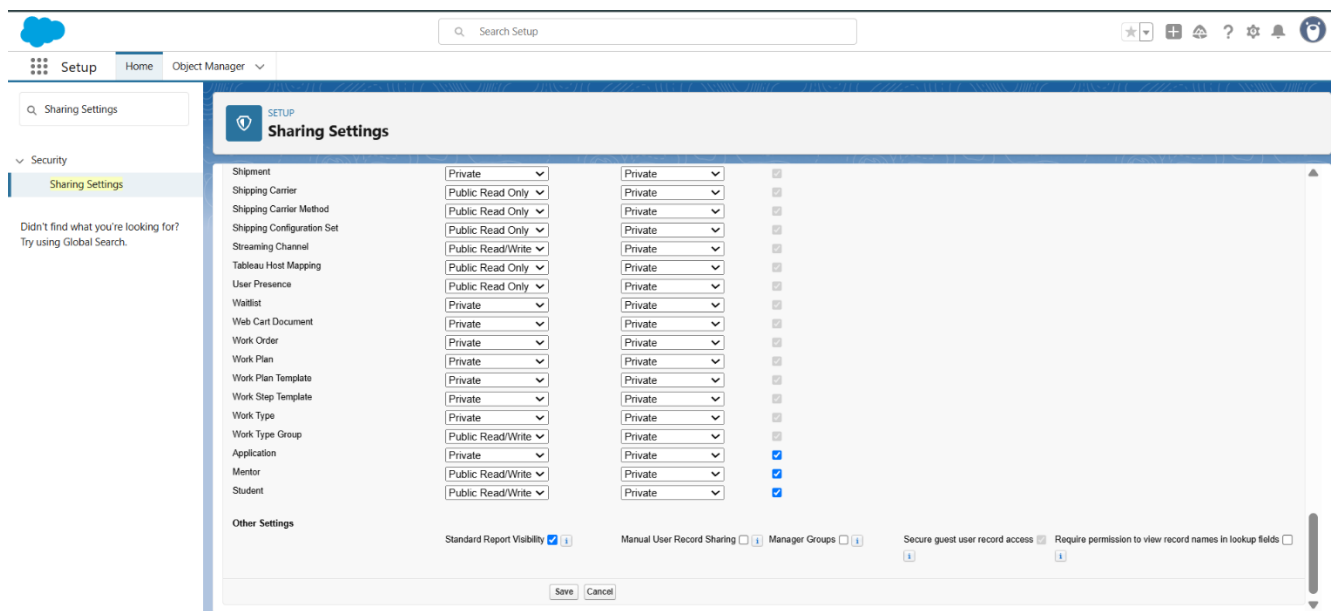
## 7. Roles

- A role hierarchy has been established with the Admissions Director at the top and Admissions Officers reporting to them to ensure proper data visibility for reporting.



## 8. OWD (Org-Wide Defaults)

- The Organization-Wide Default for the custom Application object has been set to **Private** to ensure applicant data is secure and confidential by default.



## 9. Sharing Rules

- No sharing rules have been created yet. They will be added in a later phase if specific users need access to records they do not own.

## 10. Login Access Policies

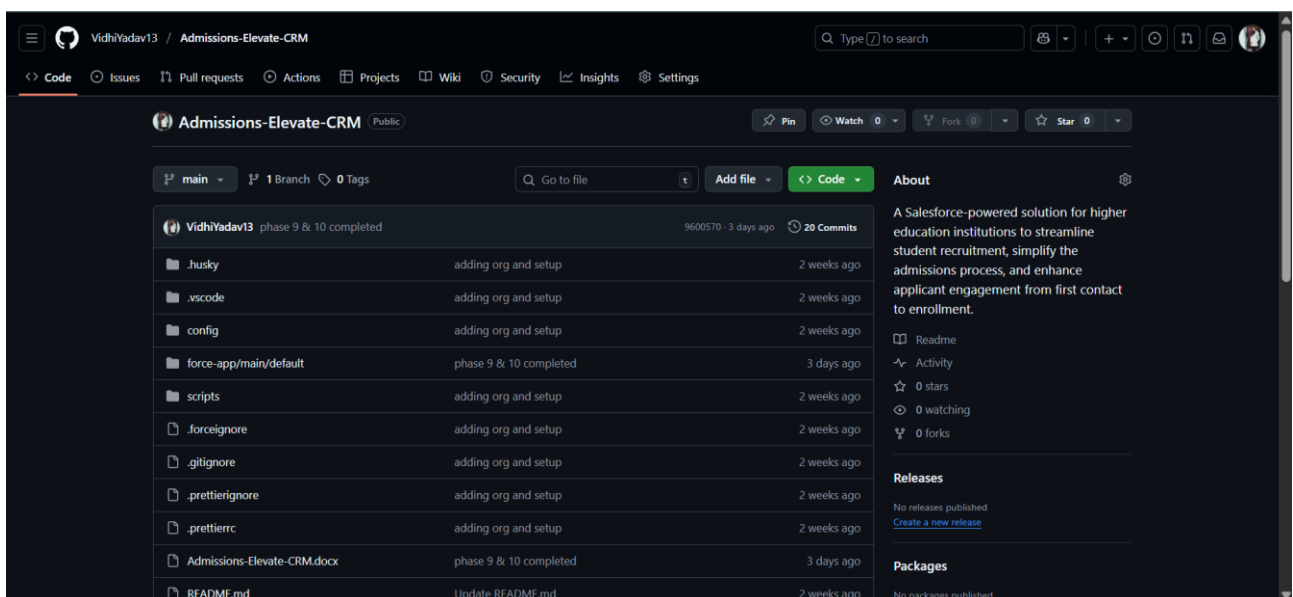
- The default login and password policies for the org have been reviewed and confirmed.

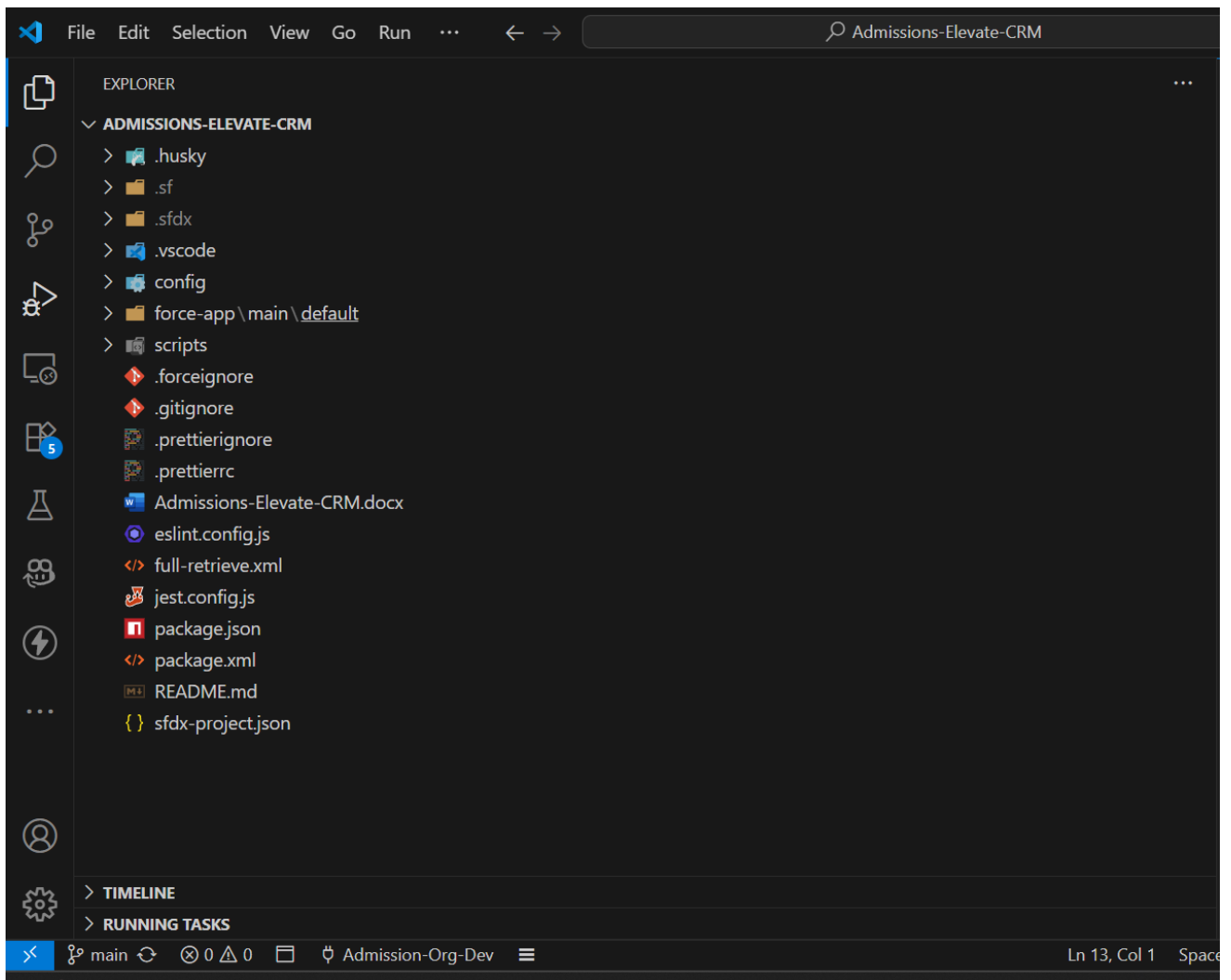
## 11. Dev Org Setup

- The Developer Org has been created and is fully configured.

## 12. Deployment Basics

- The Developer Org is connected to a VS Code project. All configurations from Phase 1 and 2 have been successfully retrieved and committed to a public GitHub repository for version control





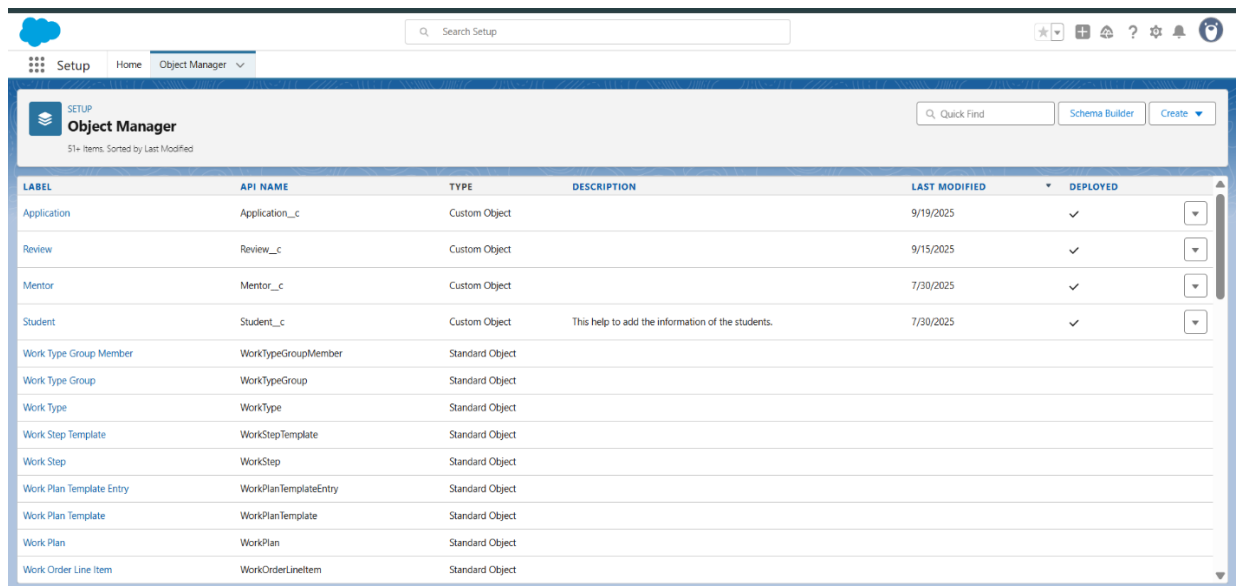
---

## Phase 3 : Data Modeling & Relationships

- **Goal:** Build the core data structure to track applications and reviews.

### 1. Standard & Custom Objects

- **Standard Objects:** We will use the standard Contact object to store information about the applicants/students. This allows us to leverage built-in features for managing person details.
- **Custom Objects:**
  - **Application (Application\_\_c):** This is the central object of the project. It will hold all the details related to a single student's application to the university.
  - **Review (Review\_\_c):** This object will store the feedback and recommendations submitted by faculty members for a specific application.



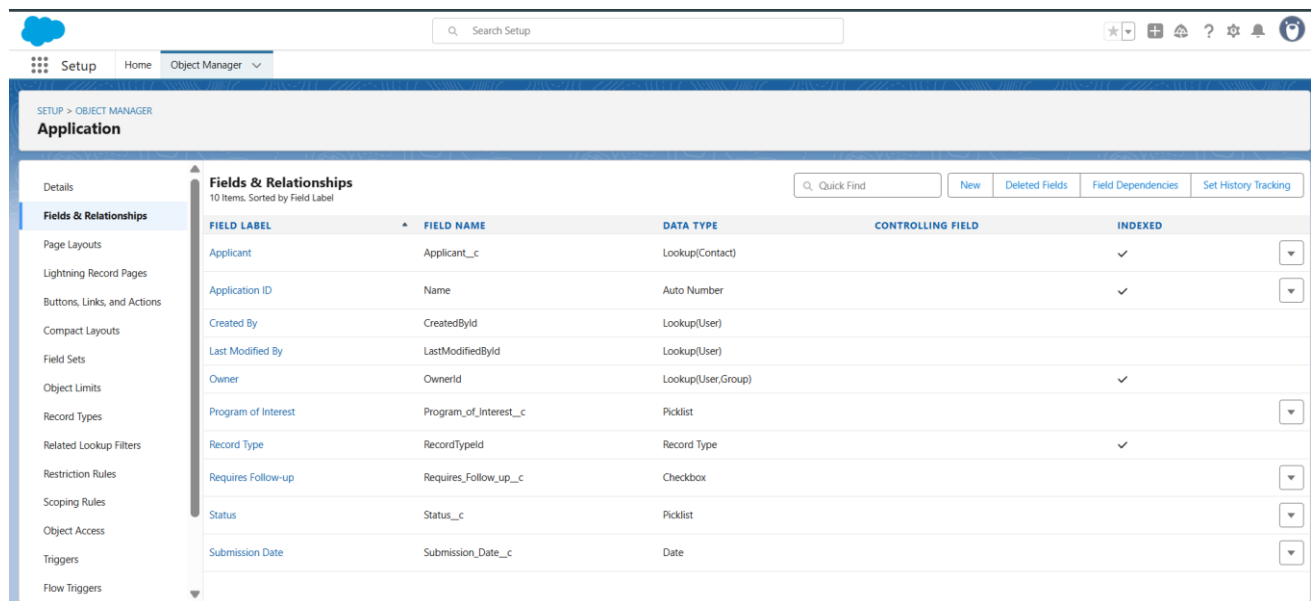
The screenshot shows the Salesforce Object Manager interface. At the top, there's a navigation bar with 'Setup', 'Home', and 'Object Manager'. Below this, a search bar and 'Quick Find', 'Schema Builder', and 'Create' buttons are visible. The main table lists objects with columns: LABEL, API NAME, TYPE, DESCRIPTION, LAST MODIFIED, and DEPLOYED. The objects listed are Application, Review, Mentor, Student, Work Type Group Member, Work Type Group, Work Type, Work Step Template, Work Step, Work Plan Template Entry, Work Plan Template, Work Plan, and Work Order Line Item.

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Application	Application__c	Custom Object		9/19/2025	✓
Review	Review__c	Custom Object		9/15/2025	✓
Mentor	Mentor__c	Custom Object		7/30/2025	✓
Student	Student__c	Custom Object	This help to add the information of the students.	7/30/2025	✓
Work Type Group Member	WorkTypeGroupMember	Standard Object			
Work Type Group	WorkTypeGroup	Standard Object			
Work Type	WorkType	Standard Object			
Work Step Template	WorkStepTemplate	Standard Object			
Work Step	WorkStep	Standard Object			
Work Plan Template Entry	WorkPlanTemplateEntry	Standard Object			
Work Plan Template	WorkPlanTemplate	Standard Object			
Work Plan	WorkPlan	Standard Object			
Work Order Line Item	WorkOrderLineItem	Standard Object			

## 2. Fields

### ○ Application Object Fields:

- Status (Picklist): Tracks the application's current stage (e.g., Submitted, Under Review, Accepted, Rejected).
- Program of Interest (Picklist): The academic program the student is applying for (e.g., Computer Science, Business Administration).
- Submission Date (Date): The date the application was officially submitted.



The screenshot shows the 'Fields & Relationships' section for the 'Application' object in Salesforce. The left sidebar lists various configuration options like Details, Page Layouts, and Field Sets. The main table lists fields with columns: FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The fields listed are Applicant, Application ID, Created By, Last Modified By, Owner, Program of Interest, Record Type, Requires Follow-up, Status, and Submission Date.

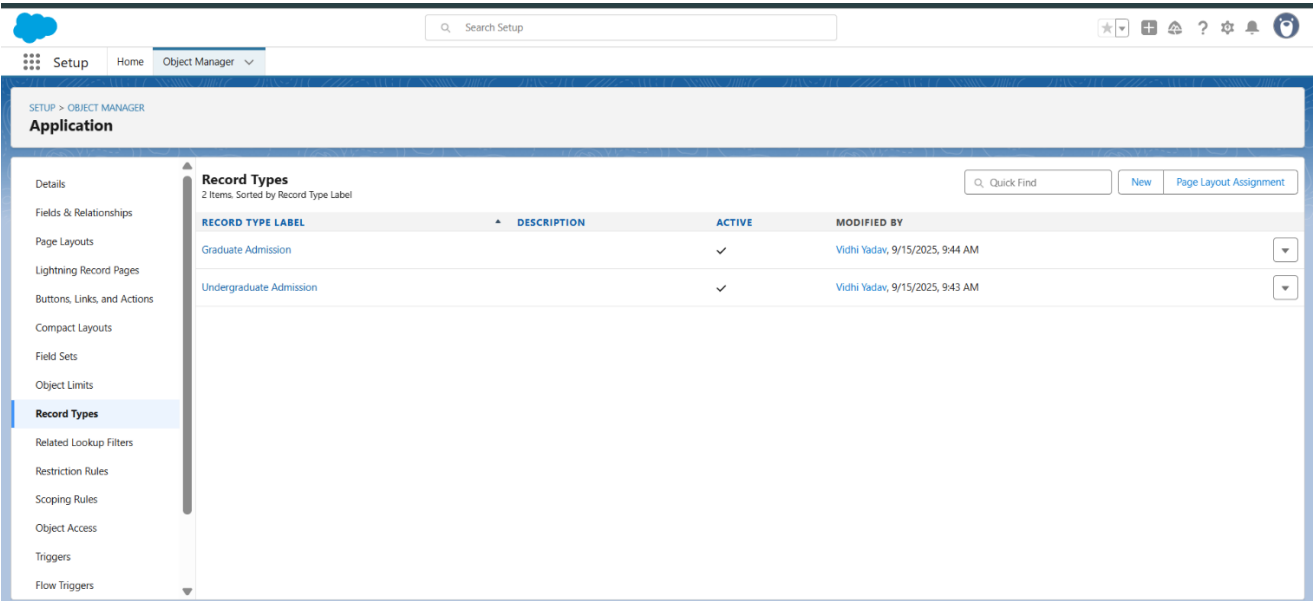
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Applicant	Applicant__c	Lookup(Contact)		✓
Application ID	Name	Auto Number		✓
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User/Group)		✓
Program of Interest	Program_of_Interest__c	Picklist		
Record Type	RecordTypeId	Record Type		✓
Requires Follow-up	Requires_Follow_up__c	Checkbox		
Status	Status__c	Picklist		
Submission Date	Submission_Date__c	Date		

### ○ Review Object Fields:

- Recommendation (Picklist): The reviewer's official recommendation (e.g., Strongly Recommend, Waitlist, Reject).
- Reviewer Comments (Long Text Area): Detailed feedback from the faculty reviewer.

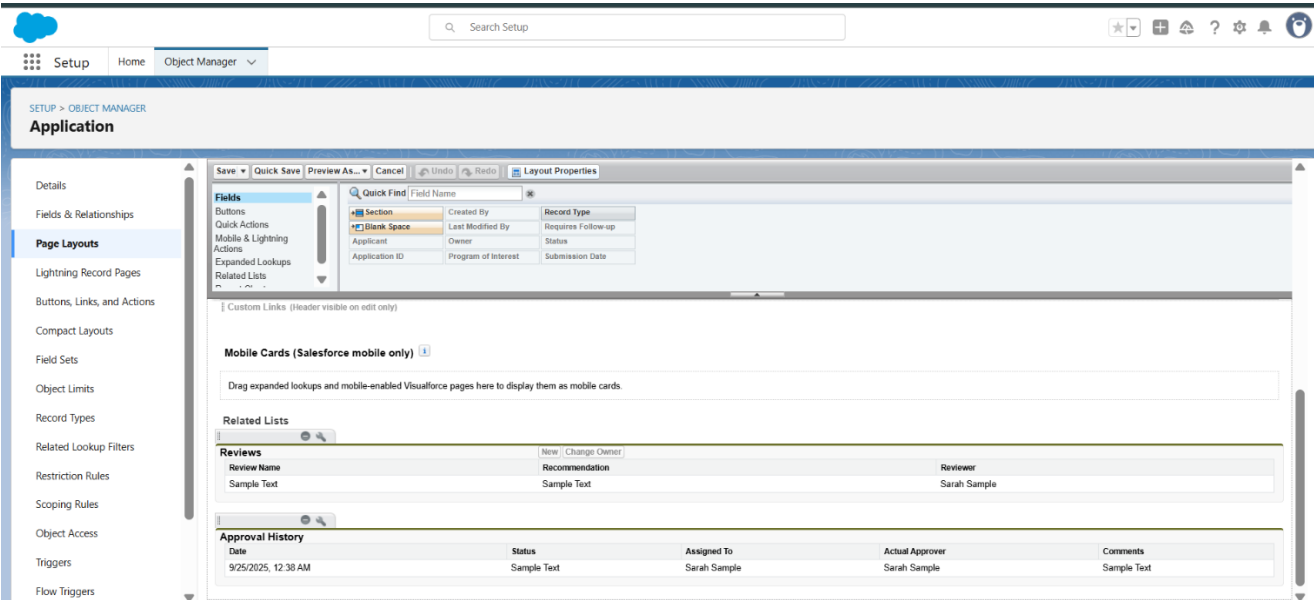
### 3. Record Types

- **Application Record Types:** Two record types have been created on the Application object: "Undergraduate Admission" and "Graduate Admission." This allows the university to have different page layouts, picklist values, and business processes for each application type in the future.



### 4. Page Layouts

- The Application record page has been customized to show the most relevant information at a glance.
- Crucially, the Reviews related list has been added to the Application page layout, so admissions officers can easily see all feedback associated with an application in one place.

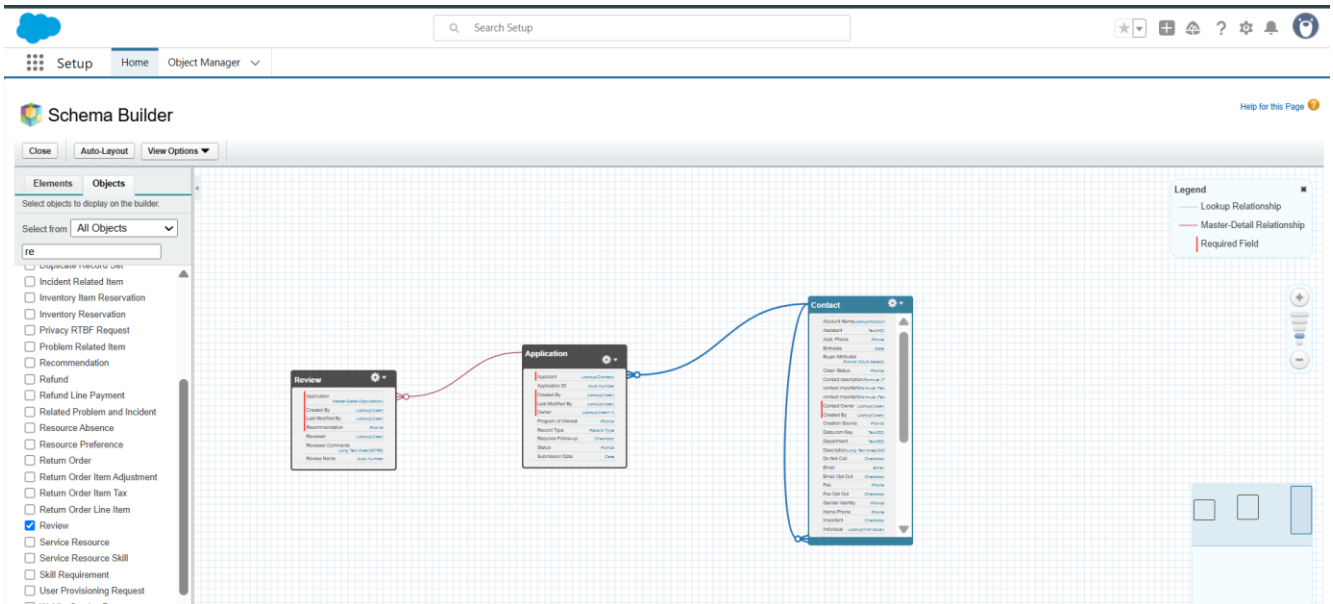


### 5. Compact Layouts

- The compact layout for the Application object has been configured to show the Applicant's Name, Status, and Program of Interest. This ensures key information is visible in list views and on mobile devices.

## 6. Schema Builder

- The Schema Builder tool was used to visualize the relationships between the Contact, Application, and Review objects, confirming the data model is structured correctly.



## 7. Lookup vs. Master-Detail vs. Hierarchical Relationships

- **Application to Contact:** This is a **Lookup Relationship**. An application is related to a contact (the applicant), but they are independent records.
- **Review to Application:** This is a **Master-Detail Relationship**. A Review is a direct child of an Application. If an application record is deleted, all of its associated review records are automatically deleted as well. This ensures data integrity.

## 8. Junction Objects

- A junction object is not required for the current data model. This would be needed if, for example, one review could be linked to multiple applications, which is not the case in our process.

## 9. External objects

- External objects are not in scope for this project. They could be used in a future phase to connect Salesforce with an external Student Information System (SIS) without migrating the data directly.

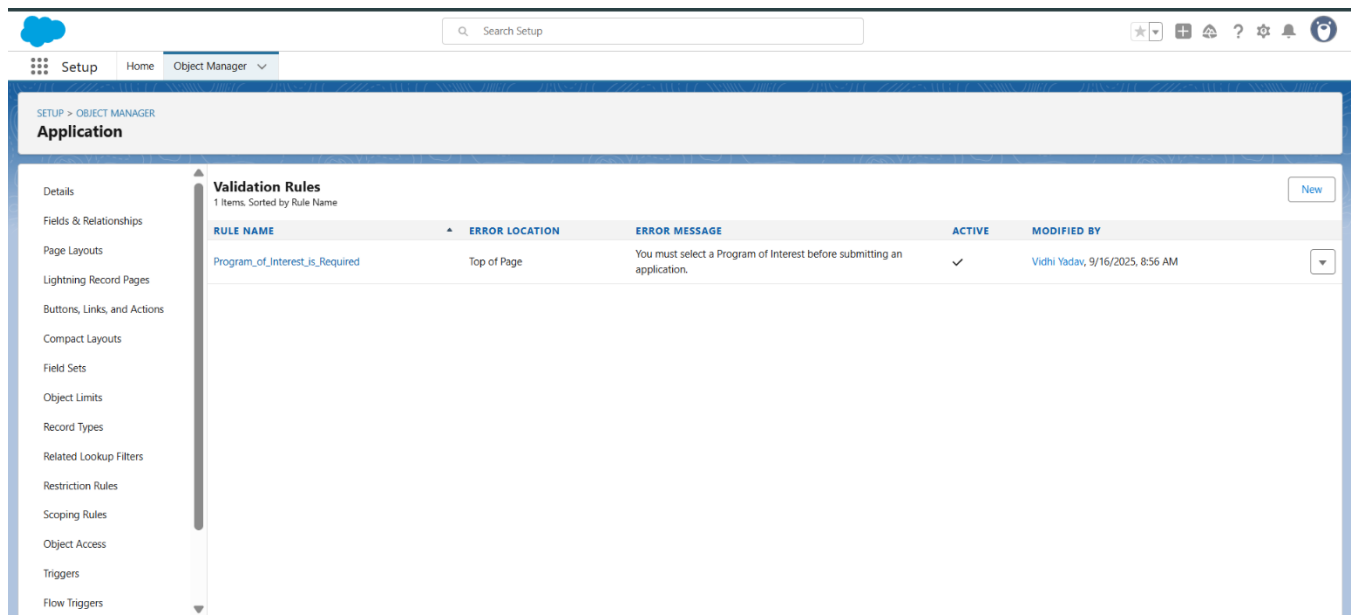


## Phase 4: Process Automation (Admin)

- **Goal:** Automate tasks to improve data quality, streamline the review process, and enhance communication with applicants.

### 1. Validation Rules

- **Example:** A rule on the Application object ensures that a record cannot be saved if the "Program of Interest" field is left blank. This guarantees all applications are properly categorized.



### 2. Workflow Rules (legacy)

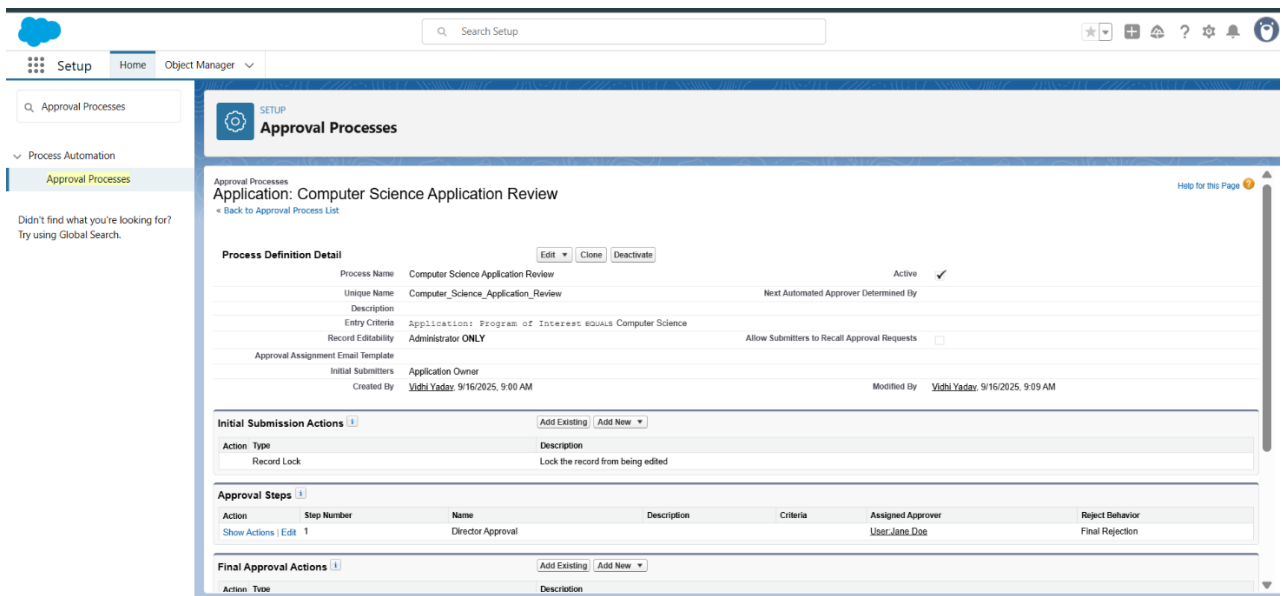
- This is a legacy automation tool. All new automations for this project are being built in Flow Builder for better performance and capabilities.

### 3. Process Builder (legacy)

- This is another legacy tool. It has been superseded by Flow Builder, which is used for all record-triggered automations in this project.

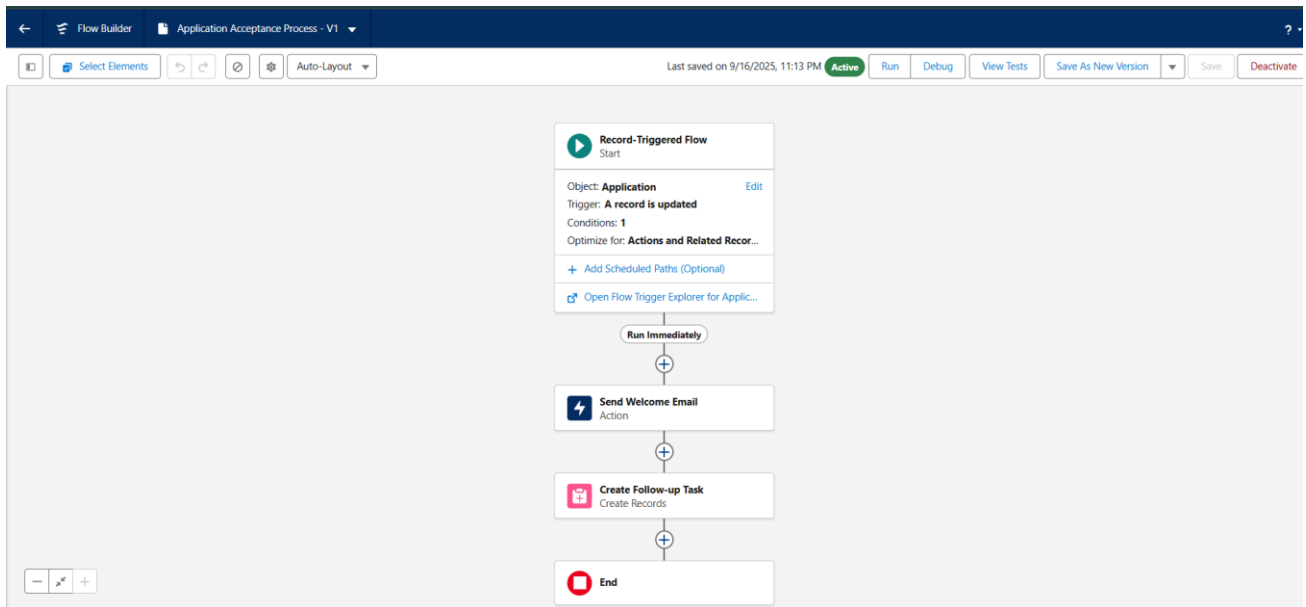
### 4. Approval Process

- **Example:** An approval process has been created for applications to the "Computer Science" program. When submitted, the application is automatically routed to the Admissions Director for formal review and approval before an acceptance decision can be made.



## 5. Flow Builder

- **Record-triggered Flow:** A flow named "Application Acceptance Process" runs automatically when an Application's status is updated to "Accepted." This single flow handles multiple actions.



## 6. Email Alerts

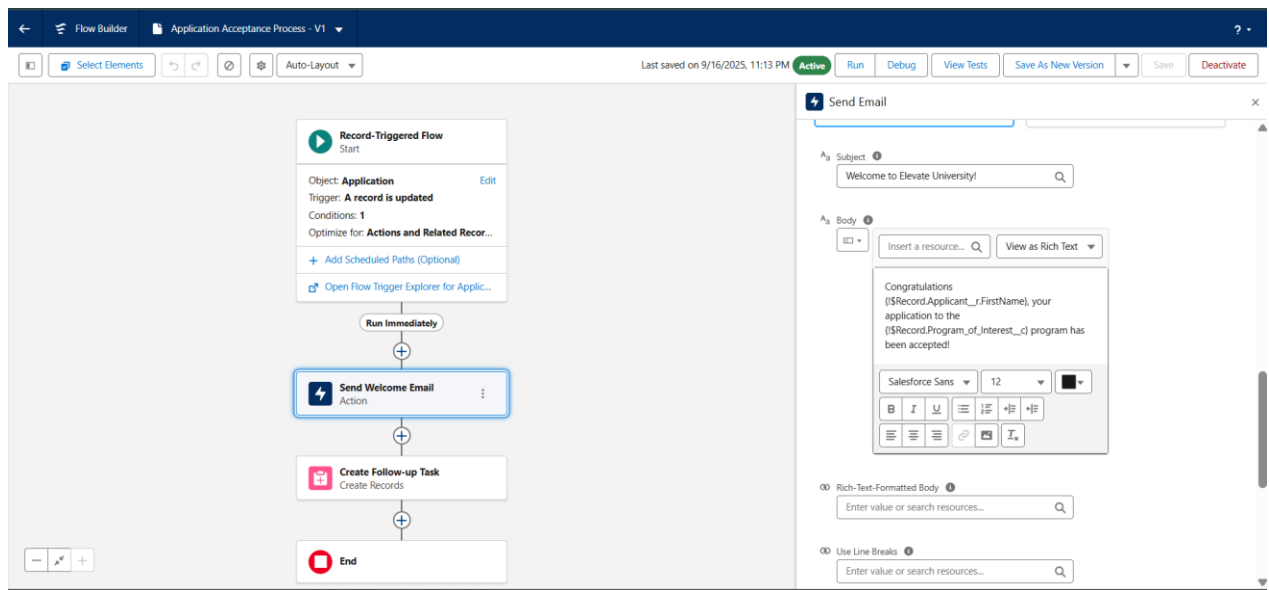
- The "Application Acceptance Process" flow includes an action to automatically send a personalized "Welcome to Elevate University!" email to the applicant using the email address on their Contact record.

## 7. Field Updates

- As part of the **Computer Science Application** Review approval process, upon final approval, a field update action automatically changes the application's Status to "Internally Approved," signaling to the admissions officer that they can proceed.

## 8. Tasks

- The "Application Acceptance Process" flow automatically creates a follow-up Task for the application owner (the Admissions Officer). The task is assigned with a subject of "Send enrollment package to new student" and is due 7 days in the future.



## 9. Custom Notifications

- A custom in-app notification could be added in the future to alert the Admissions Director on their Salesforce homepage whenever a new high-priority application is submitted for their approval.

# Phase 5: Apex Programming (Developer)

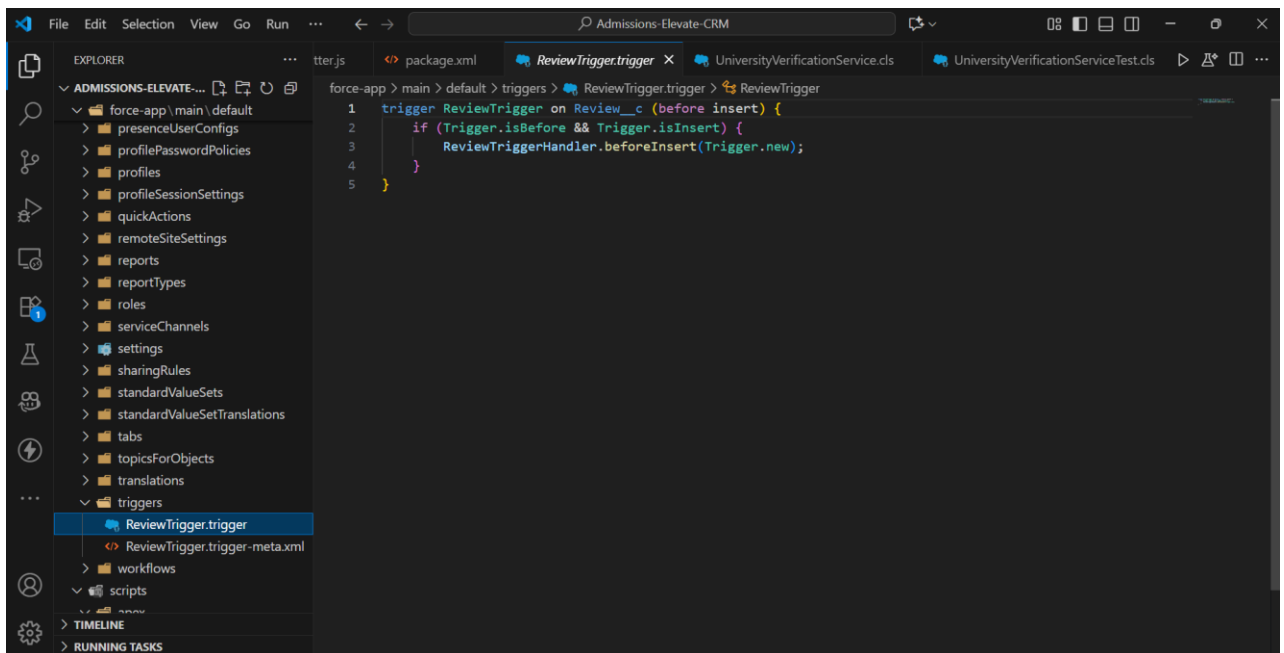
- **Goal:** Add advanced logic to enforce business rules and automate complex processes.

## 1. Classes & Objects

- Created a ReviewTriggerHandler class to hold the logic for our trigger, following best practices.
- Created a ReviewController class to expose a method for our Lightning Web Component to call.

## 2. Apex Triggers

- Implemented a ReviewTrigger that fires before insert on the Review\_\_c object. Its purpose is to call the handler class to validate the parent Application's status.



### 3. Trigger Design Pattern

- Used a handler pattern to keep the trigger file itself clean and simple. All complex logic is delegated to the ReviewTriggerHandler class, making the code more organized and easier to maintain.

### 4. SOQL & SOSL

- **SOQL:** Used a SOQL query within the trigger handler to efficiently fetch the Status\_\_c field from all related parent Application\_\_c records in a single query. The scheduled job also uses SOQL to find stale applications.

### 5. Collections: List, Set, Map

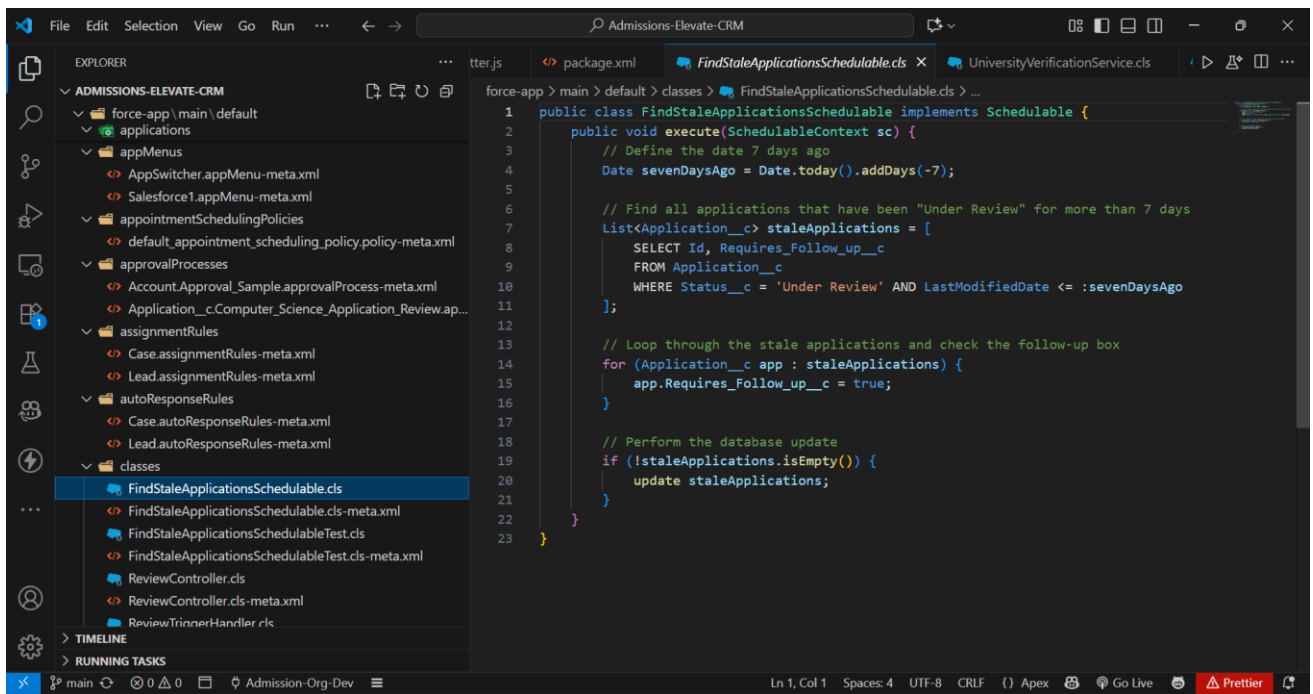
- Used a Map<Id, Application\_\_c> in the trigger handler to easily and efficiently look up the parent application for each incoming review. The scheduled class uses a List<Application\_\_c> to hold the records that need to be updated.

### 6. Control Statements

- Used if statements and for loops to iterate through the new Review\_\_c records and apply the business logic, adding an error if the parent application's status is not "Under Review."

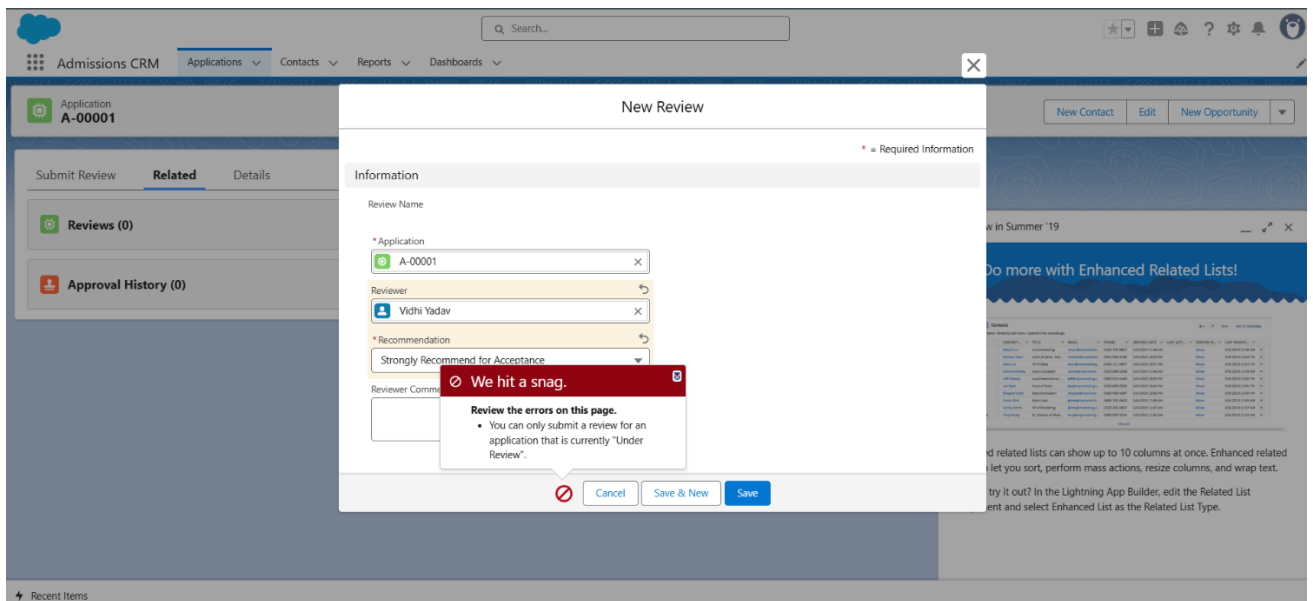
### 7. Scheduled Apex

- Created a scheduled Apex class, FindStaleApplicationsSchedulable, that runs every night to find Application\_\_c records that have been "Under Review" for more than 7 days and flags them with a checkbox.



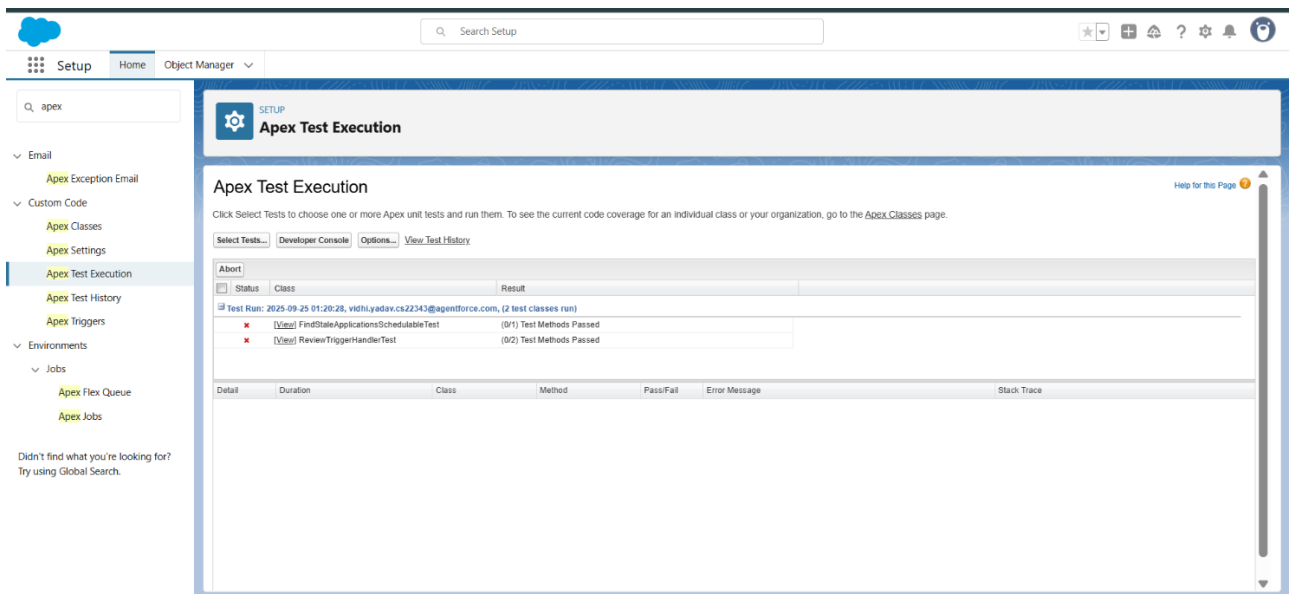
## 8. Exception Handling

- The trigger uses the .addError() method to gracefully prevent the record from being saved and display a user-friendly error message if the business rule is violated.



## 9. Test Classes

- Created dedicated test classes (ReviewTriggerHandlerTest, FindStateApplicationsSchedulableTest) for all Apex code to ensure it functions as expected and meets Salesforce's code coverage requirements for deployment.

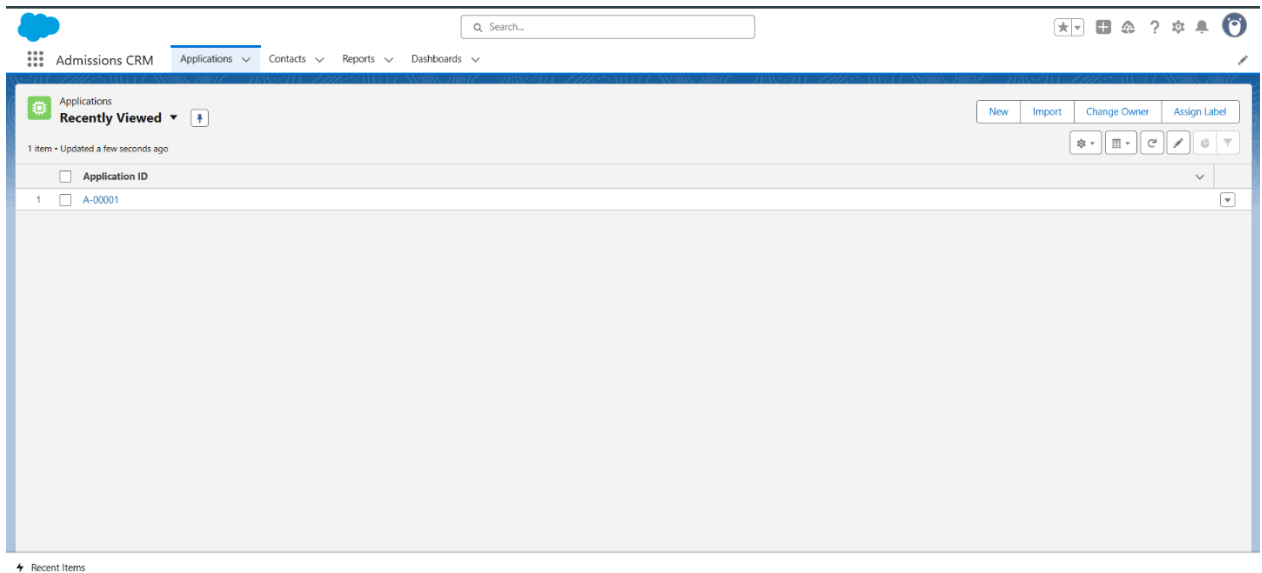


## Phase 6: User Interface Development

- **Goal:** Make the application user-friendly and efficient for the admissions team.

### 1. Lightning App Builder

- Used the Lightning App Builder to create a dedicated, branded app named "**Admissions CRM**".



### 2. Record Pages

- Edited the default Application record page to add a new custom tab, creating a dedicated space for the review submission component.

### 3. Tabs

- Added the standard **Applications** and **Contacts** tabs to the navigation menu of the "Admissions CRM" app.

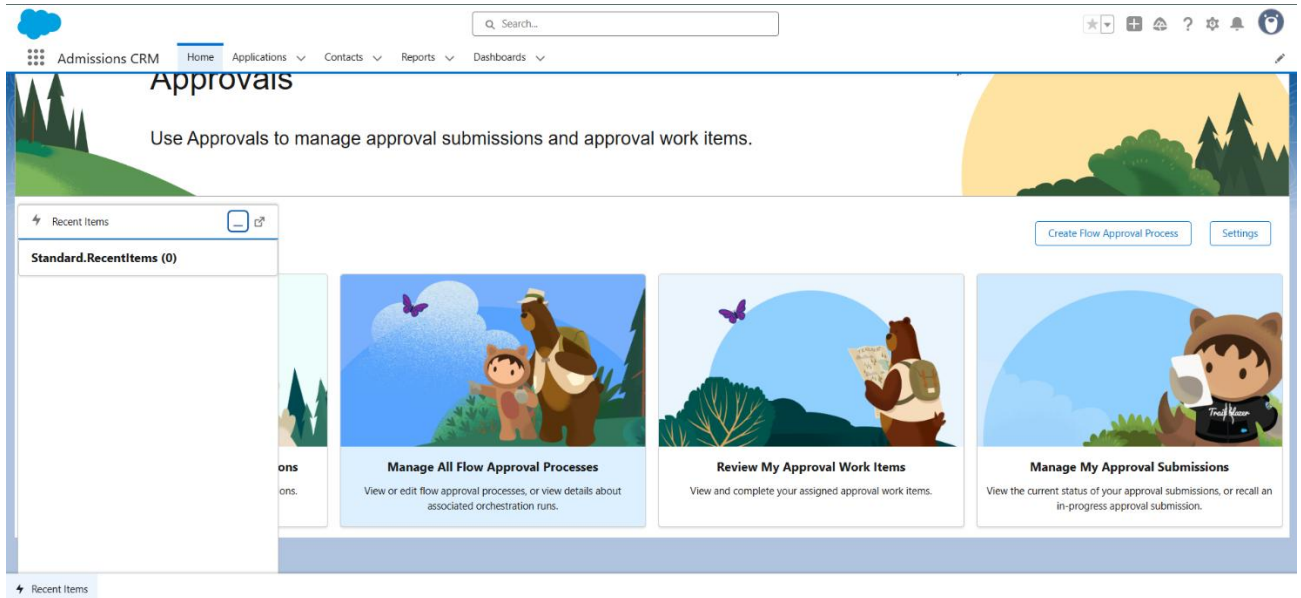
- Created a custom **"Submit Review"** tab on the Application record page to house our LWC.

#### 4. Home Page Layouts

- The home page layout has not been customized yet but could be enhanced in a later phase with dashboards showing key admissions metrics.

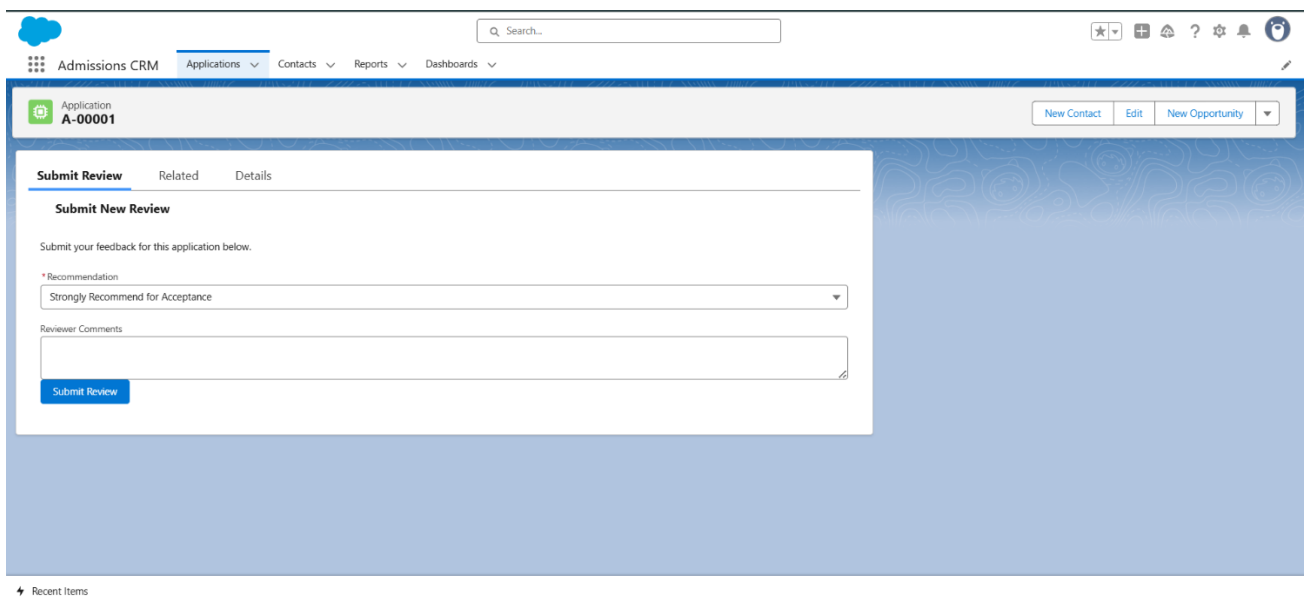
#### 5. Utility Bar

- Added the standard **Recent Items** component to the app's utility bar for quick access to recently viewed records.



#### 6. LWC (Lightning Web Components)

- Built a custom Lightning Web Component named **reviewSubmitter**. This component provides a clean and simple form for users to enter their review feedback.

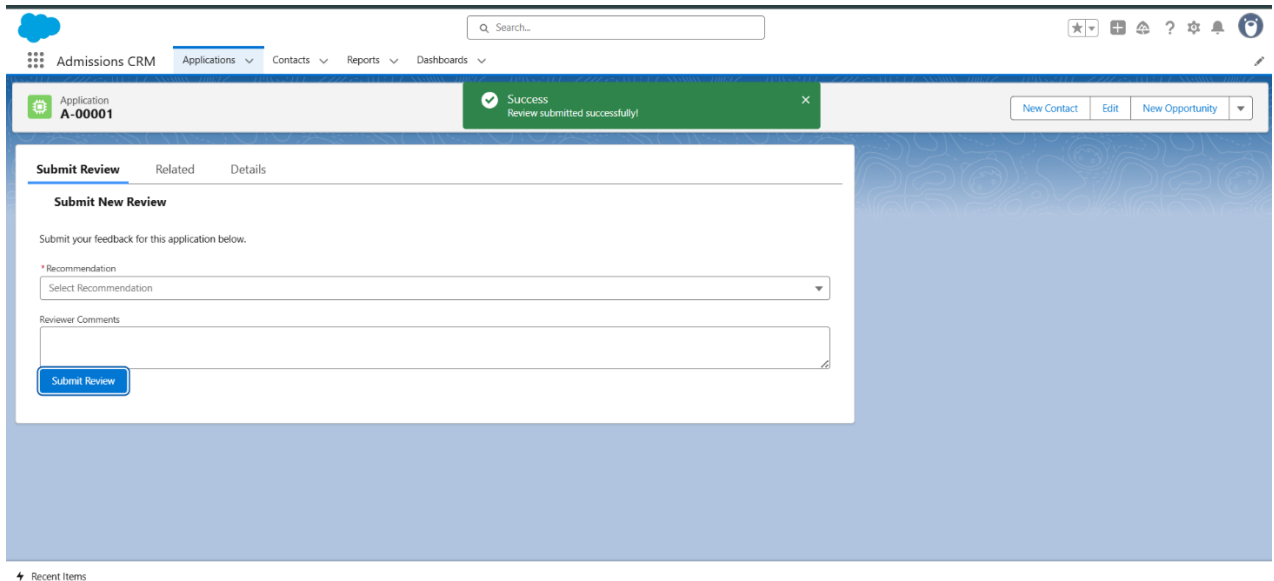


## 7. Apex with LWC

- The reviewSubmitter component's JavaScript imports the submitReview method from the ReviewController Apex class, allowing the frontend to securely interact with the backend.

## 8. Events in LWC

- The component uses the ShowToastEvent to display success or error notifications to the user after they attempt to submit a review. It also uses standard onchange and onclick events to handle user input.



## 9. Imperative Apex Calls

- The handleSubmit function in the component's JavaScript makes an **imperative call** to the submitReview Apex method. This is triggered when the user clicks the "Submit Review" button.

## 10. Navigation Service

- The navigation service is not used in this component, but a future enhancement could be to automatically navigate the user back to the "Related" tab after a successful submission.

---

# Phase 7: Integration & External Access

- **Goal:** Connect the application to outside systems to bring in external data.

### 1. Named Credentials

- Named Credentials are not used in this phase but are a best practice for securely storing the URL and authentication details for an API endpoint.

### 2. Web Services (REST/SOAP)



- A **REST callout** is used to connect to a free, public web service that provides a list of universities. This allows us to verify an applicant's email domain.

### 3. Callouts

- An Apex class, `UniversityVerificationService`, was created to perform the HTTP callout. This class sends a request to the external university API and processes the JSON response.

```

1 public class UniversityVerificationService {
2
3     // This method will be called to start the verification process
4     @future(callout=true)
5     public static void verifyUniversityEmail(Id contactId) {
6         // Find the contact record to get their email
7         Contact applicant = (SELECT Id, Email FROM Contact WHERE Id = :contactId LIMIT 1);
8
9         if (String.isBlank(applicant.Email)) {
10             System.debug('Contact has no email address.');
```

### 4. Remote Site Settings

- To allow Salesforce to make the callout, the API's base URL (`http://universities.hipolabs.com`) was added to the **Remote Site Settings**. This is a mandatory security step.

The screenshot shows the Salesforce Setup interface. The left sidebar has a search bar and a navigation menu with 'Setup', 'Home', and 'Object Manager'. The main content area is titled 'Remote Site Settings' and contains a table of 'All Remote Sites'.

Action	Remote Site Name	Namespace Prefix	Remote Site URL	Active	Created By	Created Date	Last Modified By	Last Modified Date
Edit   Del	ApexDevNet	-	http://www.apexdevnet.com	✓	EPIC_OrgFarm	7/17/2025, 9:28 AM	EPIC_OrgFarm	7/17/2025, 9:28 AM
Edit   Del	UniversityAPI	-	http://universities.hipolabs.com	✓	Yadav_Vidhi	9/18/2025, 7:09 AM	Yadav_Vidhi	9/18/2025, 7:09 AM

## 5. Other Integration Tools (Conceptual Knowledge)

- **Platform Events, Change Data Capture, Salesforce Connect:** These are more advanced integration tools not required for this project. They are used for real-time event-driven integrations and connecting to external databases.

## 6. API Limits

- All integrations are subject to Salesforce's API call limits. The current integration is designed to be run on-demand to stay well within the free developer org limits.

---

# Phase 8: Data Management & Deployment

- **Goal:** Manage the application's data and understand deployment methodologies.

### 1. Data Import Wizard

- The Data Import Wizard was used to perform a simple import of 4-5 sample **Contact** records to act as our applicants. This was done by preparing an applicants.csv file.

### 2. Data Loader

- The desktop Data Loader application was used for a more complex data load. An applications.csv file was prepared, which included the Salesforce IDs of the Contact records. This allowed us to insert new **Application** records and automatically link them to the correct applicants.

The screenshot shows the 'Mapping Dialog' window in Salesforce. It has a title bar with a Salesforce logo and a close button. The main area contains the instruction 'Match the Salesforce object fields to your CSV column headers.' Below this are two buttons: 'Clear Mapping' and 'Auto-Match Fields to Columns'. A search bar with a magnifying glass icon is present. Below the search bar is a table with three columns: 'Field Name/Relationship', 'Field Label', and 'Field Data Type'. The table contains the following rows:

Field Name/Relationship	Field Label	Field Data Type
Owner-Id	Owner ID	Lookup (Group, User)
RecordType-Id	Record Type ID	Lookup (RecordType)
Requires_Follow_up__c	Requires Follow-up	boolean
Submission Date__c	Submission Date	date

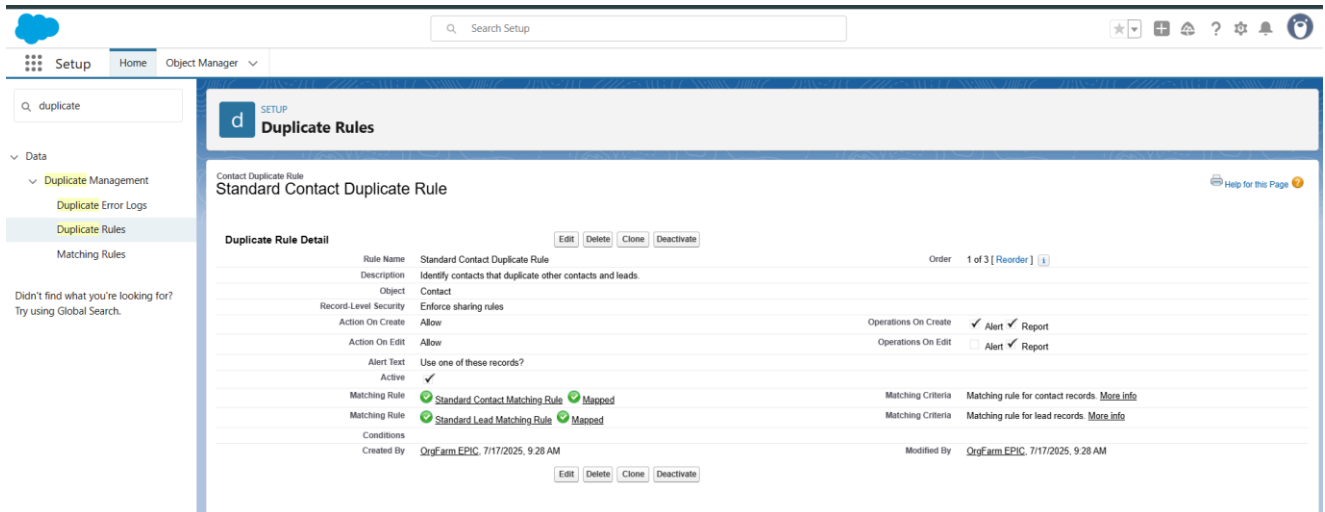
Below the table is a section with the instruction 'Drag the Salesforce object fields down to the column mapping.' and a green arrow icon. Below this is another table with two columns: 'CSV Column Header' and 'Field Name/Relationship'. The table contains the following rows:

CSV Column Header	Field Name/Relationship
Applicant__c	Applicant__c
Program_of_Interest__c	Program_of_Interest__c
Status__c	Status__c

At the bottom of the dialog are three buttons: 'OK', 'Save Mapping', and 'Cancel'.

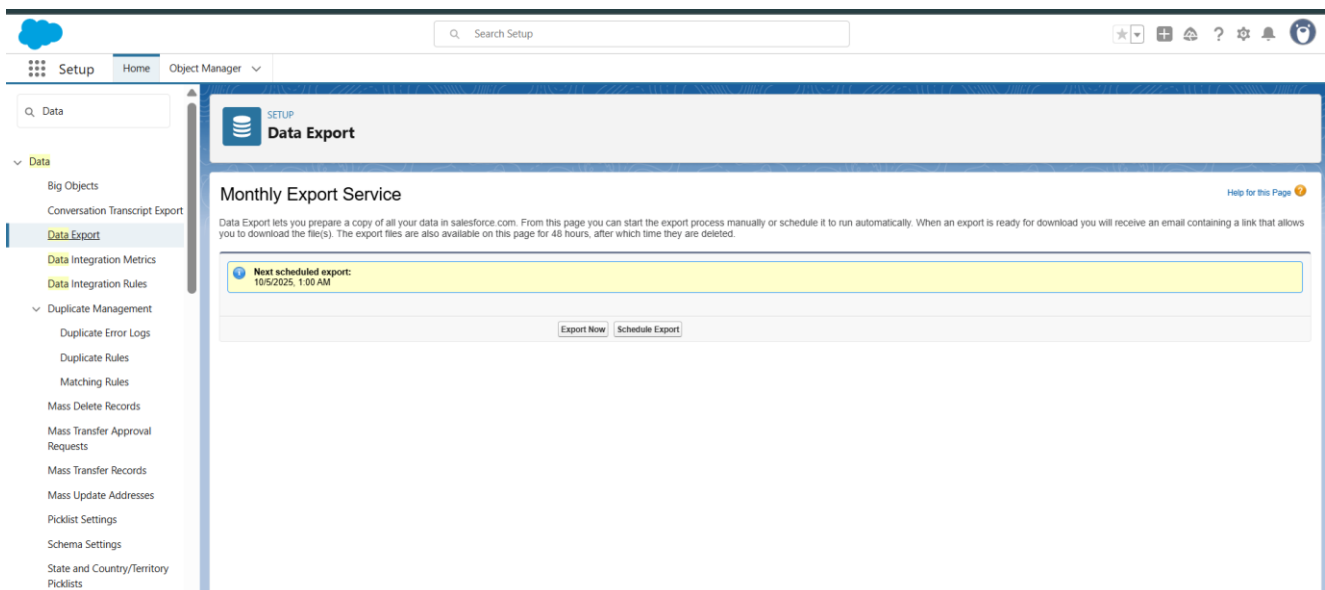
### 3. Duplicate Rules

- A **Duplicate Rule** was created and activated on the Contact object. It uses the standard matching rule for the Email field to block the creation of a new applicant record if an existing contact already has the same email address. This is critical for maintaining clean data.



### 4. Data Export & Backup

- The native **Data Export** service was configured to perform an automated, weekly backup of all the project's key data, including Contacts, Applications, and Reviews.



### 5. VS Code & SFDX

- This is the primary method used for all development and deployment in this project. All metadata components (Apex classes, custom fields, etc.) created in these phases have been successfully retrieved from the org to a local VS Code project and pushed to a GitHub repository for version control.

### 6. Other Deployment Tools (Conceptual Knowledge)

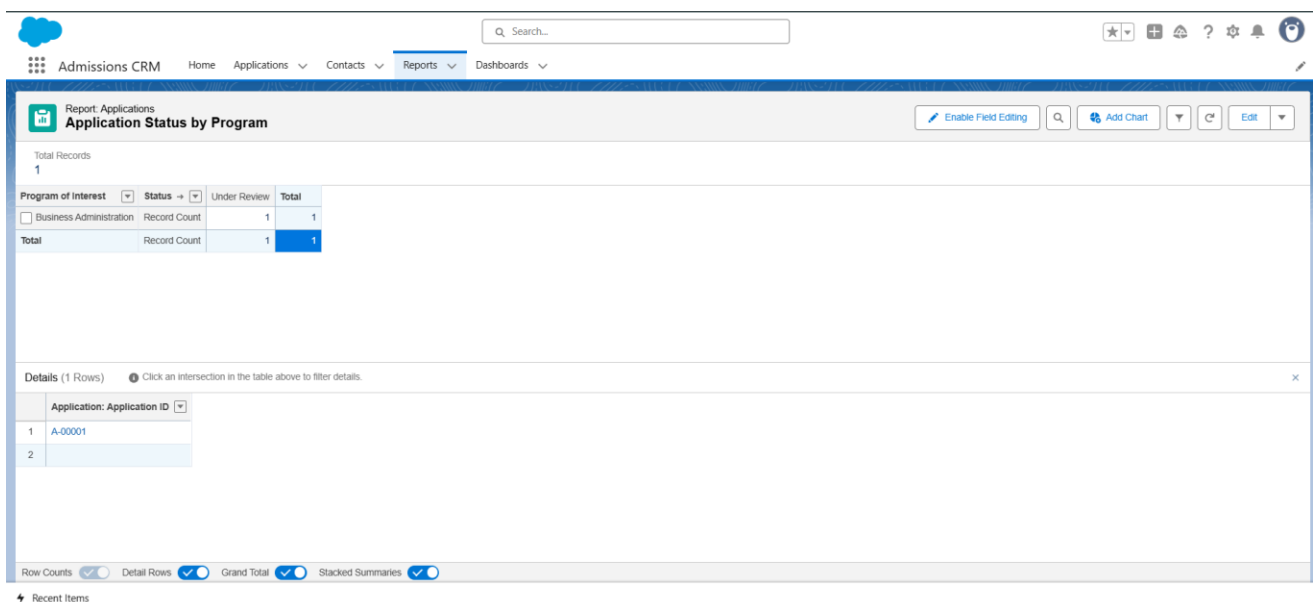
- **Change Sets:** A UI-based tool for moving metadata between connected orgs (e.g., from a Sandbox to Production).
- **ANT Migration Tool:** An older command-line deployment tool that uses XML files. It has largely been replaced by SFDX for modern development.

## Phase 9: Reporting, Dashboards & Security Review

➤ **Goal:** Monitor business & secure data.

### 1. Reports

- **Application Status by Program:** A matrix report that shows a count of all applications, grouped by the academic program they are for and their current status (e.g., Submitted, Under Review, Accepted).



Report: Applications  
**Application Status by Program**

Total Records: 1

Program of Interest	Status →	Under Review	Total
<input type="checkbox"/> Business Administration	Record Count	1	1
<b>Total</b>	Record Count	1	1

Details (1 Rows) Click an intersection in the table above to filter details.

Application: Application ID
1 A-00001
2

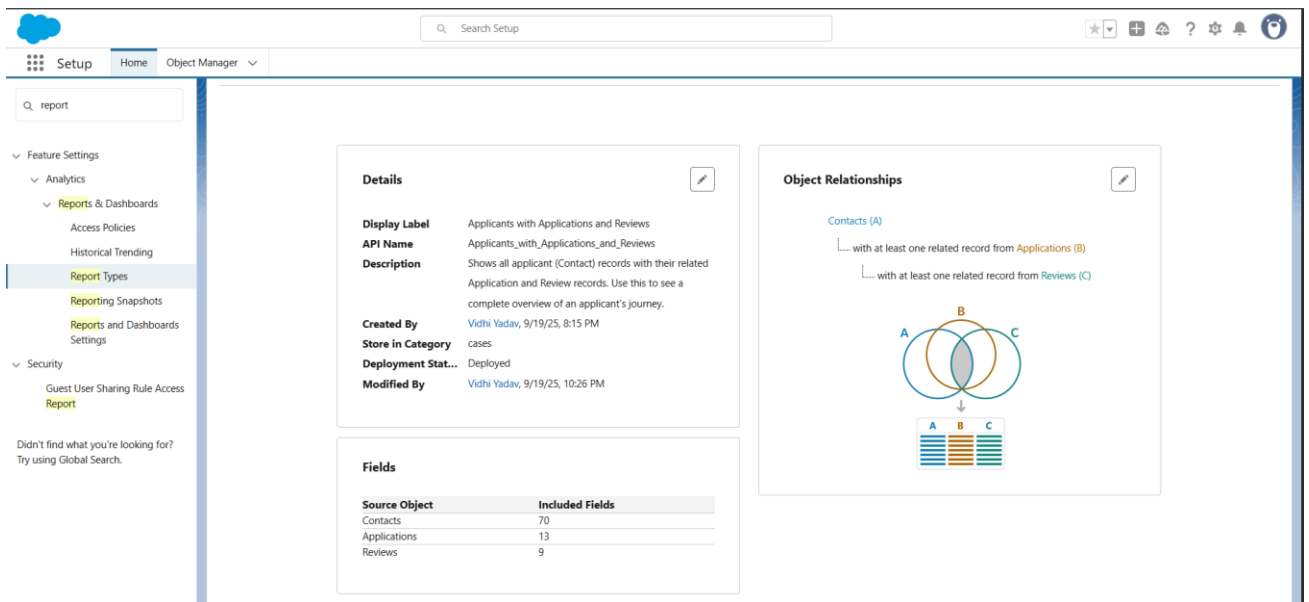
Row Counts ☐ Detail Rows ☒ Grand Total ☒ Stacked Summaries ☒

Recent Items

- **All Applicant Info Report:** A detailed report showing a complete overview of each applicant, including their contact details, the status of their application, and the recommendations from any submitted reviews.

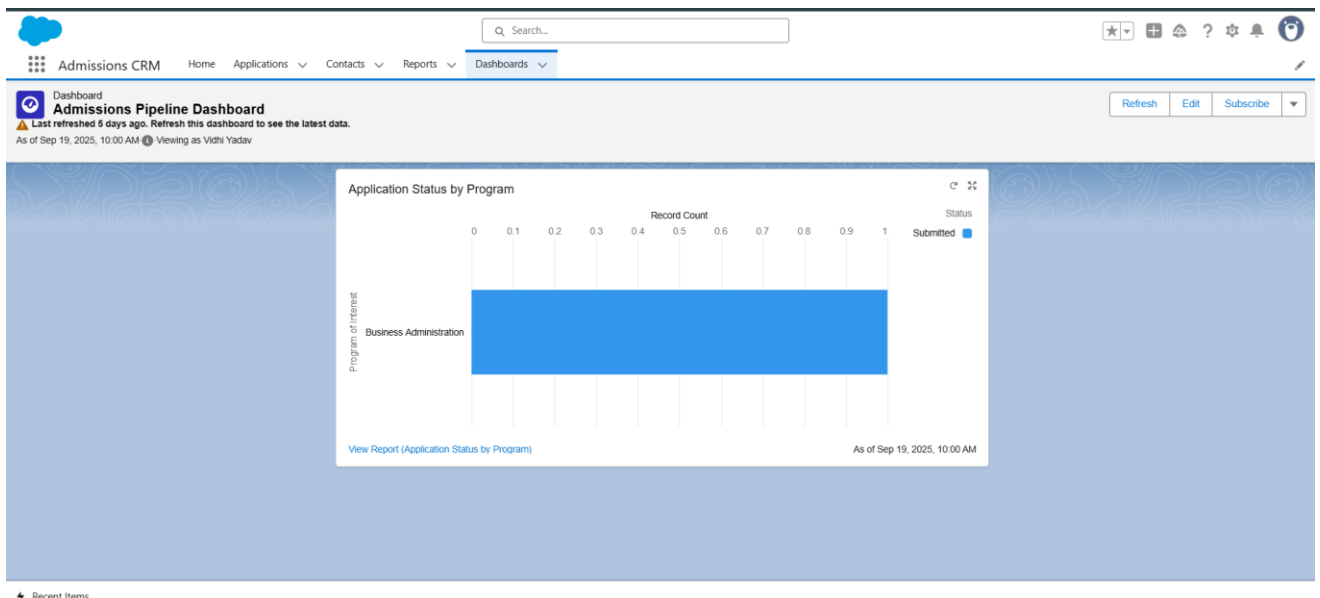
### 2. Report Types

- **Custom Report Type:** A custom report type named "Applicants with Applications and Reviews" was created. It links the Contact, Application, and Review objects, allowing for comprehensive reports that pull data from all three objects simultaneously.



### 3. Dashboards

- **Admissions Pipeline Dashboard:** A central dashboard was created to provide a high-level, visual overview of the admissions process. It features a stacked bar chart component based on the "Application Status by Program" report, allowing the Admissions Director to see the pipeline at a glance.



### 4. Dynamic Dashboards

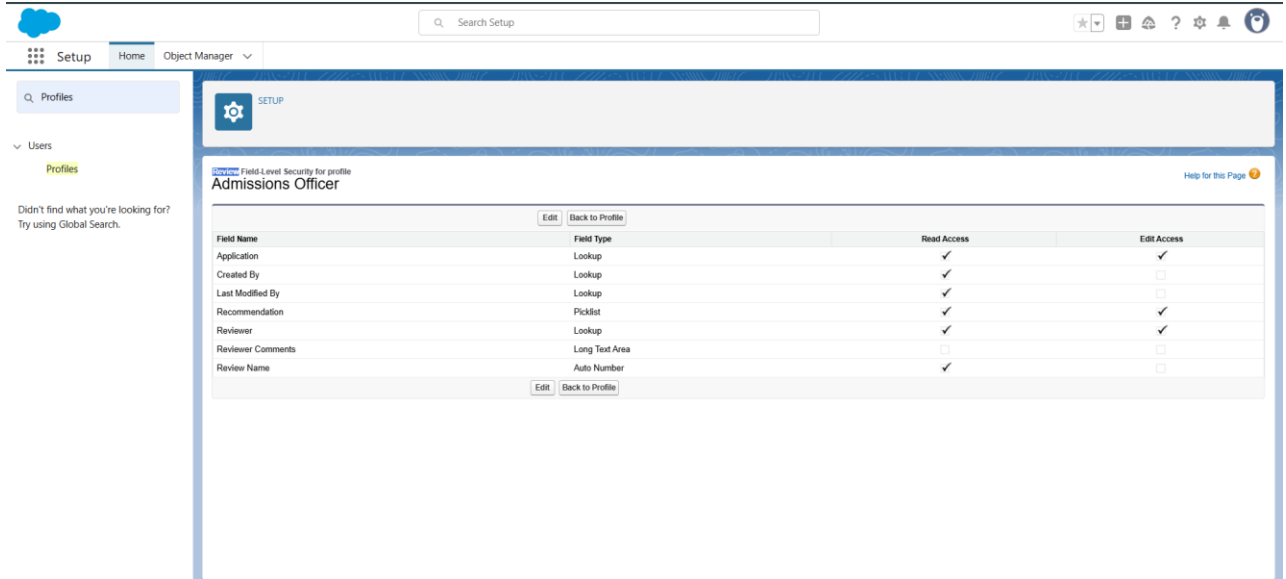
- This feature has not been implemented but is a potential future enhancement. A dynamic dashboard could be configured so that when an Admissions Officer views it, they only see data related to the applications they own.

### 5. Sharing Settings

- The Organization-Wide Default for the Application and Contact objects has been confirmed as **Private**, ensuring applicant data is secure by default.

### 6. Field Level Security

- Field Level Security was configured on the Admissions Officer profile to hide the Reviewer Comments field on the Review object. This ensures that sensitive feedback from faculty is only visible to authorized users like the Admissions Director.



## 7. Session Settings

- The default session settings were reviewed. For a real-world implementation, the standard 2-hour timeout would be reduced to 1 hour to enhance the security of sensitive student data.

## 8. Login IP Ranges

- The ability to set Login IP Ranges on a profile was reviewed. This feature can be used to restrict access to the university's physical network, preventing users from logging in from unauthorized locations.

## 9. Audit Trail

- The Setup Audit Trail was reviewed. It provides a complete log of all administrative and configuration changes made to the org, which is critical for security and compliance audits.

# Phase 10: Final Presentation & Demo Day

- **Goal:** Wrap it up like a real project delivery.

## 1. Pitch Presentation

- A slide deck was prepared outlining the project's journey:
  - **Problem:** The inefficiencies and errors of a manual, spreadsheet-based admissions process.
  - **Solution:** The Admissions-Elevate-CRM, a centralized and automated solution.

- **Benefits:** Increased efficiency, better data quality, improved applicant experience, and real-time insights.

## 2. Demo Walkthrough

- A live demo script was prepared to showcase a "day in the life" of an admissions officer:
  1. Start on the **Dashboard** to get a high-level overview.
  2. Open an **Application** record.
  3. Use the custom **reviewSubmitter LWC** to submit a new review.
  4. Show the newly created Review record in the related list.
  5. Demonstrate submitting a "Computer Science" application to the **Approval Process**.
  6. Show the result of the "Application Acceptance Process" **Flow** by changing an application's status to "Accepted," which triggers a welcome email and a follow-up task.

## 3. Handoff Documentation

- A simple user guide was created, explaining how to perform key tasks like submitting a review and managing the approval process. An admin guide was also drafted, outlining the key custom objects (Application\_\_c, Review\_\_c) and automations.

## 4. Portfolio Project Showcase

- A professional summary was written for portfolio use:
    - **Title:** Salesforce University Admissions CRM
    - **Description:** "Designed and developed a comprehensive CRM on the Salesforce platform to digitize and streamline the university admissions process. The solution features a secure data model for applicant information, automated review and approval processes, and a custom Lightning Web Component for faculty feedback, all visualized through a central reporting dashboard."
    - **Skills:** Salesforce Administration, Apex, Lightning Web Components (LWC), SOQL, Process Automation (Flow, Approval Process), Data Modeling, Git, SFDX.
    - **Link:** <https://github.com/VidhiYadav13/Admissions-Elevate-CRM>
-