**Salesforce Student Admission Management System**

**Phase 1 (Problem Understanding & Industry Analysis):**

* **Problem Statement**

Many colleges and universities still rely on manual processes, paper forms, and spreadsheets to manage admissions. These methods lead to inefficiencies, errors, and delays in the admission lifecycle. Applicants lack transparency and real-time updates, while admission officers struggle with high volumes, manual evaluations, and fragmented tracking of applications, enrollments, and fees.

Administrators also face challenges with limited analytics, making it hard to forecast trends or optimize resources. Legacy solutions often fail to provide the customization and unified experience institutions need.

This project addresses these issues by offering a cloud-based, automated platform that streamlines applications, supports structured decision workflows, automates enrollment tracking, and delivers actionable insights through reports and dashboards.

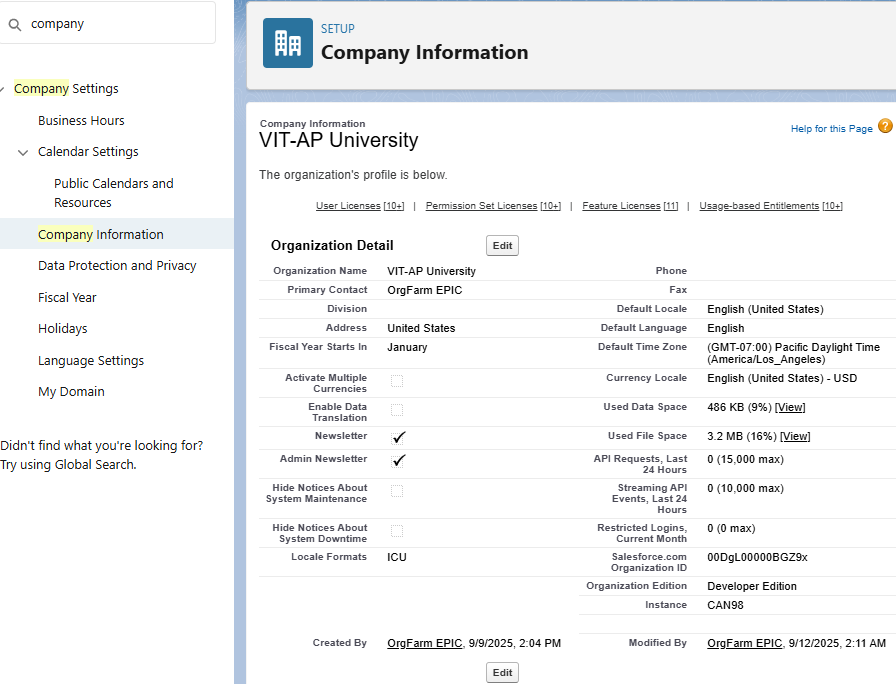
**Problem Understanding & Industry Analysis**

* **Requirement Gathering**  
  Collected requirements from the perspective of a university admissions team: online student application, review workflow, enrollment tracking, and reporting needs.
* **Stakeholder Analysis**  
  Identified stakeholders such as Admission Officers, Faculty Reviewers, Students/Applicants, and Administrators, along with their roles and expectations in the admission lifecycle.
* **Business Process Mapping**  
  Mapped the end-to-end admission process starting from student application submission → review & evaluation → decision publishing → enrollment confirmation.
* **Industry-specific Use Case Analysis**  
  Researched admission challenges faced by educational institutions: high application volumes, manual data entry, lack of real-time tracking, and reporting inefficiencies.

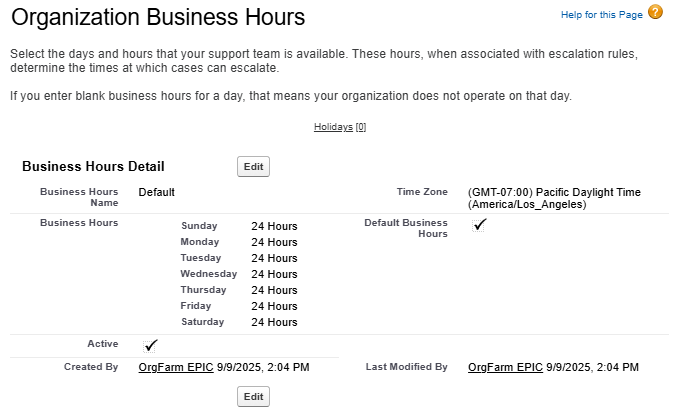
**Phase 2 (Org Setup & Configuration):**

In this phase, the Salesforce Developer Org was configured to align with the requirements of the Student Admission Management System. Key configuration activities included:

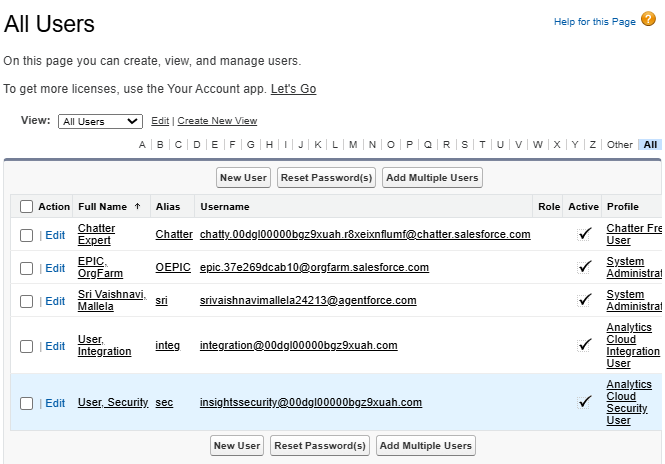
* **Salesforce Edition Selection**  
  A Salesforce **Developer Edition** was used to build and test the project, as it provides core CRM features, customization, and LWC development capabilities at no cost.
* **Company Profile Setup**  
  The company profile was configured with institution details such as **College/University name, locale, time zone, currency, and default language**, ensuring consistency across records.



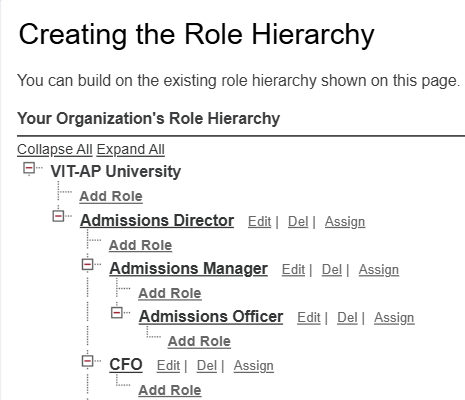
* **Business Hours & Holidays**  
  Defined admission office working hours and academic holidays to be used in workflows (like application submission SLAs or automated responses)



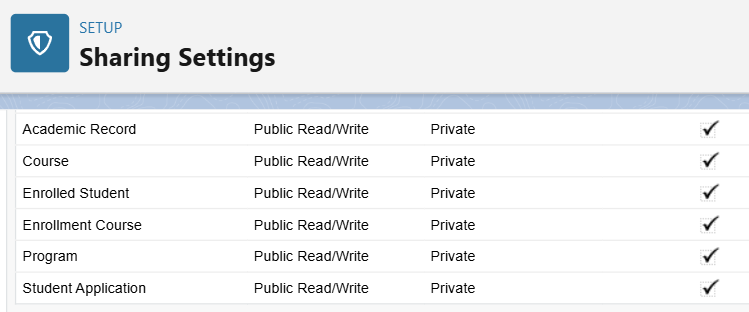
* **Fiscal Year Settings**  
  Configured fiscal year settings to align with the **academic calendar** for reporting admission data semester/annual basis.
* **User Setup & Licenses**  
  Created sample users (e.g., **Admission Officer, Faculty Reviewer, System Admin**) with appropriate Salesforce licenses for testing.



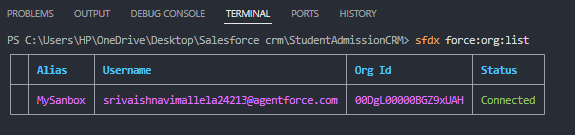
* **Profiles & Roles**  
  Configured **custom profiles** (e.g., Admission Officer, Student Counselor) and **role hierarchy** (University → Admissions Director → Admission Manager→ Admission Officer) to reflect the institution’s structure.



* **Permission Sets**  
  Built **permission sets** for granting extra access to specific users (e.g., granting access to dashboards or LWC components).
* **Organization-Wide Defaults (OWD)**  
  Set baseline record-level security:
  + Student Applications → **Private** (only owners & managers can see)
  + Courses → **Private** (since they’re institution-wide)
  + Enrolled Students → **Private** (sensitive student details)



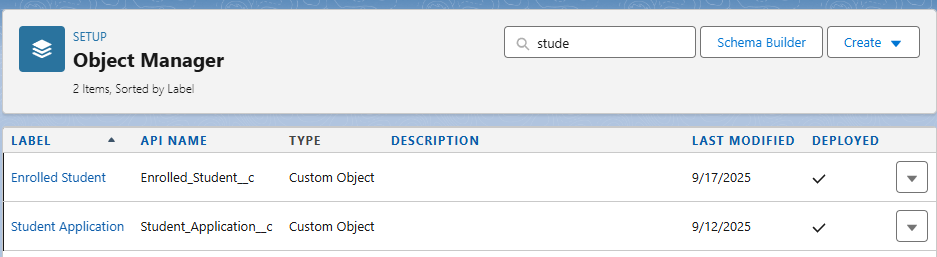
* **Sharing Rules**  
  Created sharing rules so **Admission Officers** can access applications submitted in their department, while **Admins** retain full access.
* **Login Access Policies**  
  Configured policies for secure login, including session timeout and trusted IP ranges.
* **Dev Org Setup**  
  Installed Salesforce CLI and linked VS Code to the Dev Org for **metadata deployment and retrieval**.

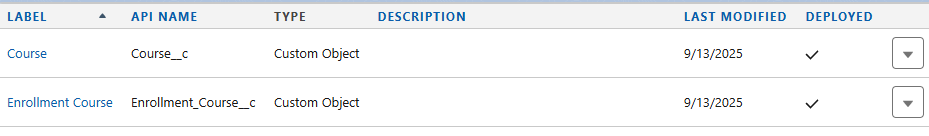


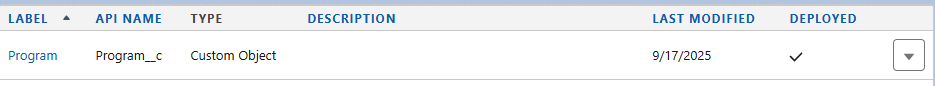
* **Sandbox Usage**  
  Though not mandatory in Developer Edition, sandbox strategies were discussed (e.g., UAT for testing, Full Sandbox for production-like data).
* **Deployment Basics**  
  Learned deployment methods:
  + **Change Sets** (Admin way)
  + **SFDX CLI & GitHub** (Developer way, used in this project)

**Phase 3 (Data Modelling & Relationships):**

* **Standard & Custom Objects:**
* **Standard objects:** User, Contact for student contacts and admission officers.
* **Custom objects:**
  + Student\_Application\_\_c – stores student application details (name, DOB, email, GPA, program applied)
  + Enrolled\_Student\_\_c – tracks enrolled students and links to Student Application
  + Course\_\_c – stores available coursese
  + Enrollment\_Course\_\_c – junction object linking students to courses
  + Program\_\_c – stores programs offered by the institution
  + Academic\_Records\_\_c – stores student grades and transcripts



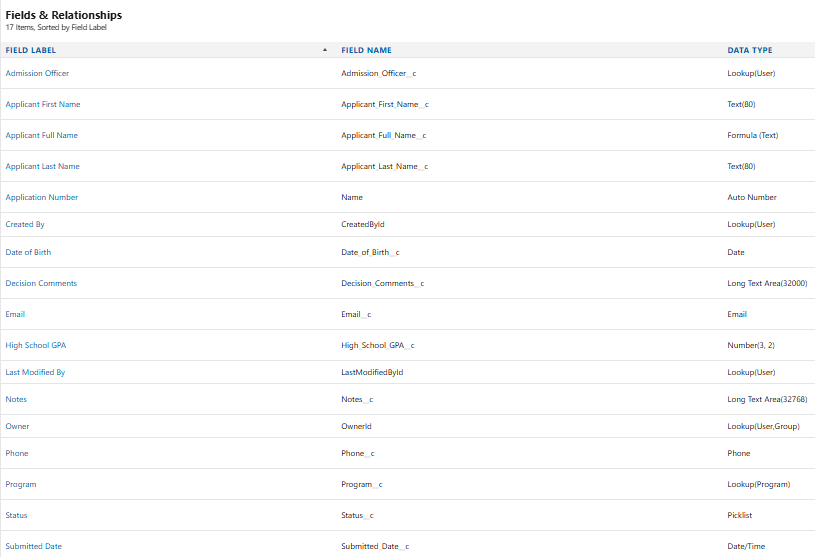




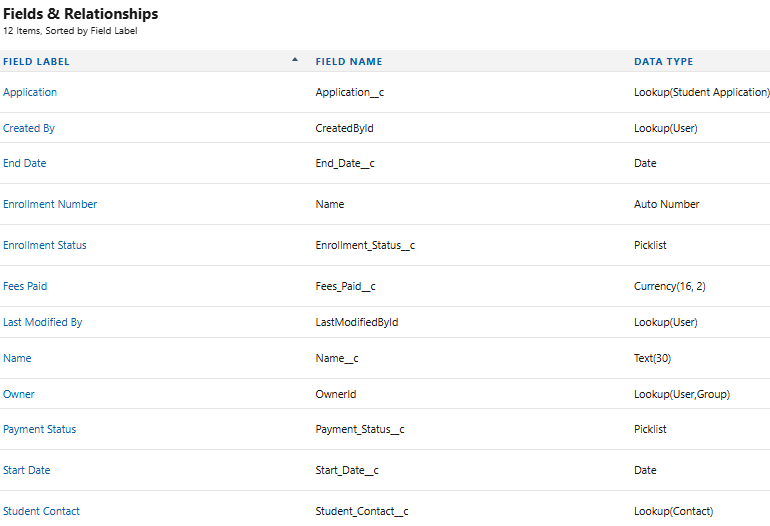


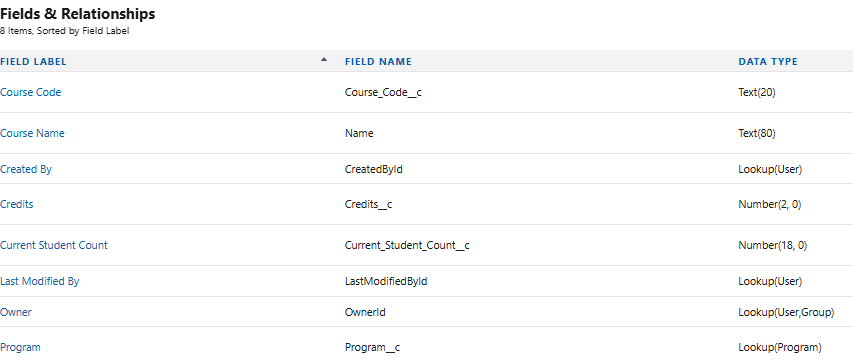
* **Fields:**
* Defined fields for each object:
  + Student\_Application\_\_c: First Name, Last Name, Email, Program (Lookup), GPA, Status, Submitted Date etc
  + Enrolled\_Student\_\_c: Student (Lookup), Application (Lookup, required), Enrollment Status, Start/End Dates, Fees Paid etc
  + Course\_\_c: Name, Course Code, Credits, Program (Lookup) etc
  + Program\_\_c: Name, Duration, Type etc
  + Academic\_Records\_\_c: Student (Lookup), Course (Lookup), Grade, Semester etc
* Used appropriate data types: Text, Number, Date, Lookup, Picklist, Auto Number

**Student\_Application\_\_c:**

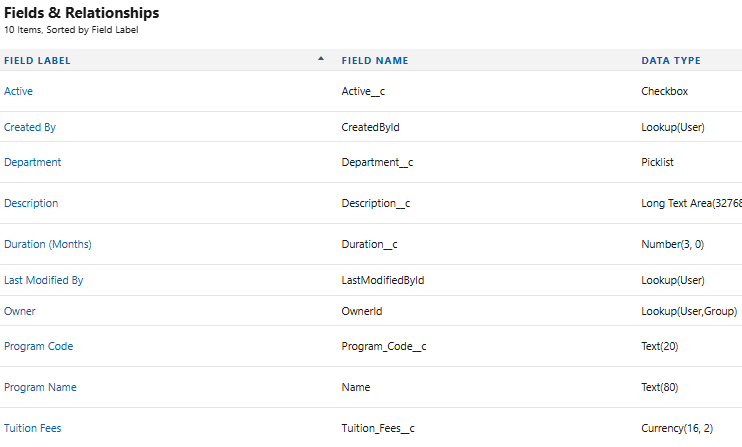


**Enrolled\_\_Student\_\_c:**

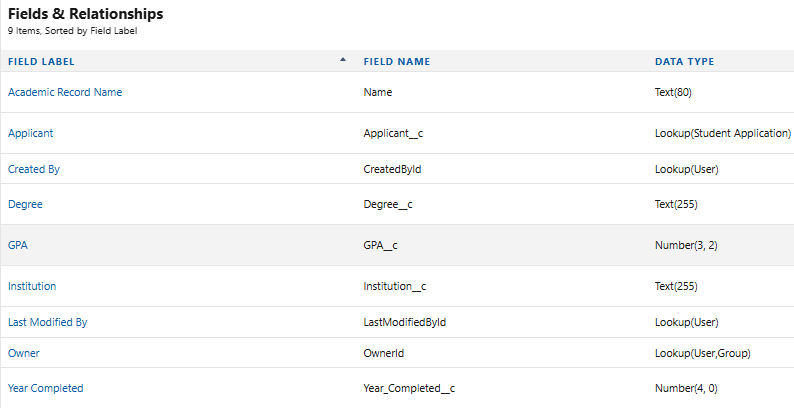


**Course\_\_c:**

**Program\_\_c:**

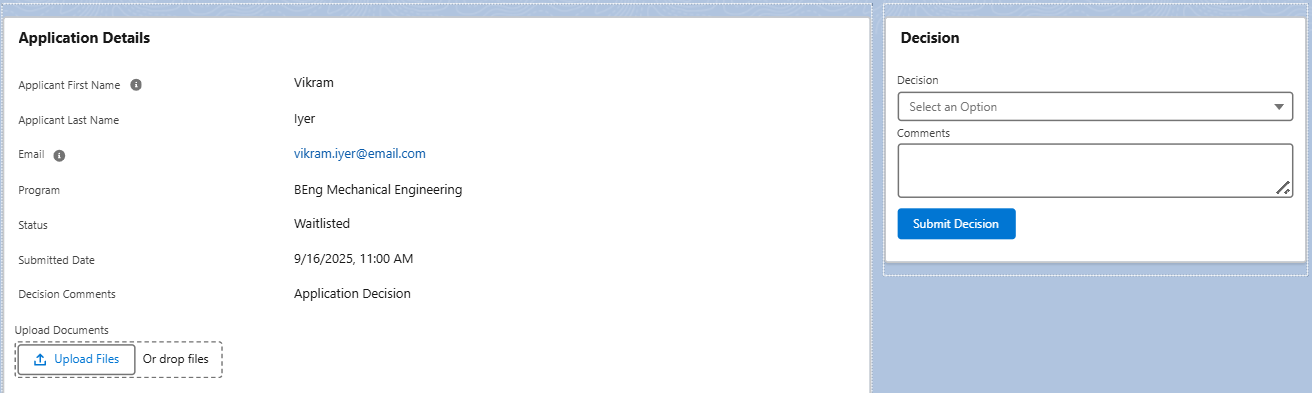


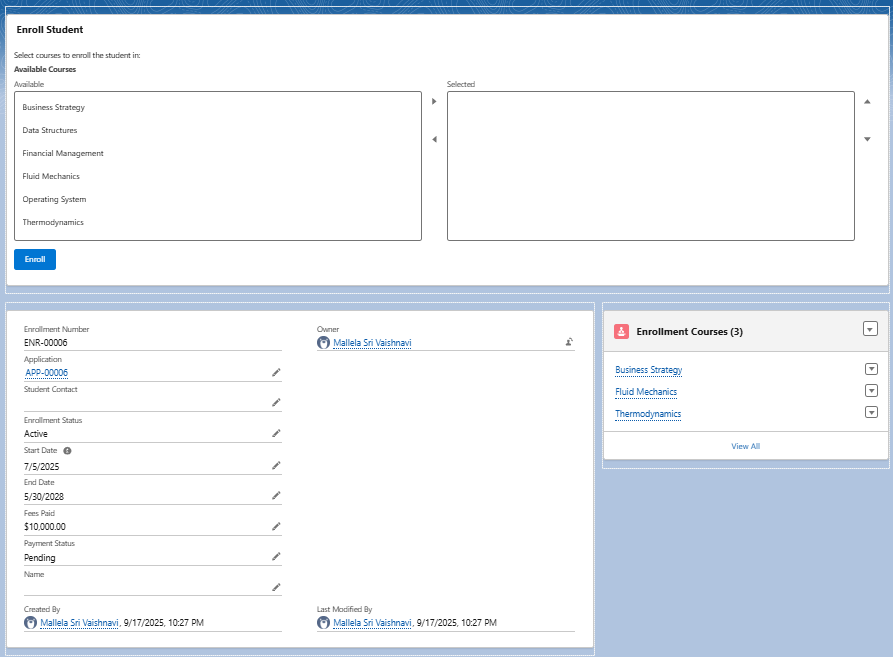
**Academic\_Record\_\_c:**



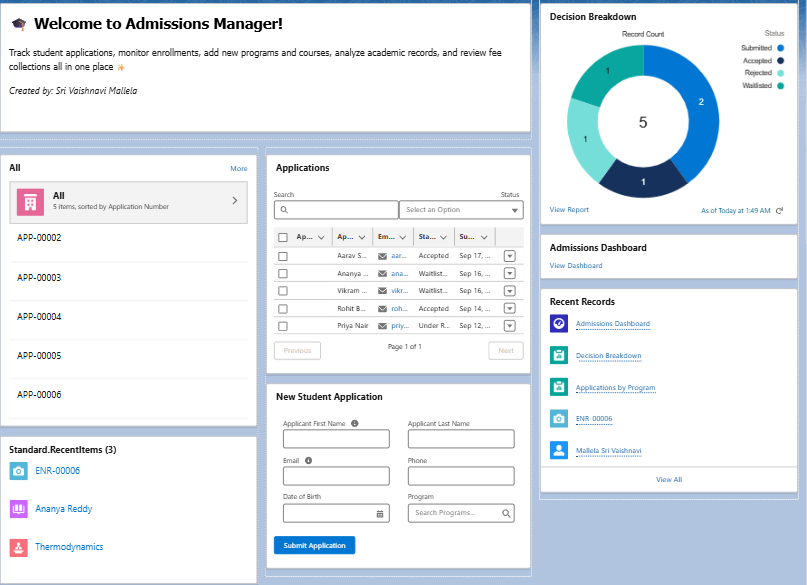
* **Record Types & Page Layouts:**
* Record types to differentiate program types or application types (optional).
* Customized **page layouts** for each object to show key fields:
  + Student\_Application→ applicant details, decision picklist, comments, uploaded documents
  + Enrolled\_Student → enrollment info, fees, selected courses
  + Home\_Page\_Default → Home Page layout of my App, consists of rich text, reports,recent records, list view, recent items, quicklinks.
* Compact layouts for mobile and Lightning pages for quick info access

**Lightning record page of Student\_Application:**

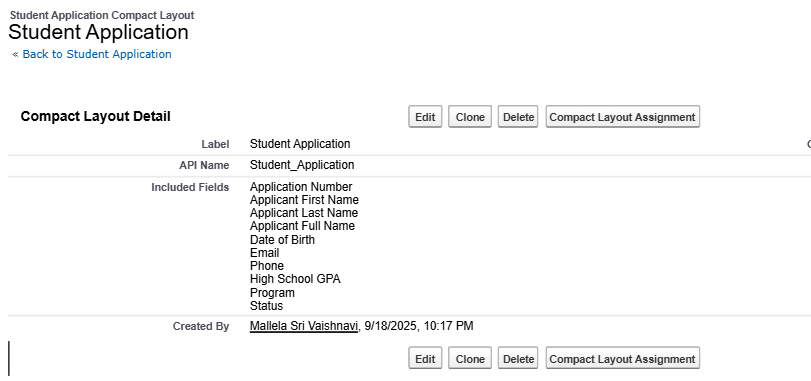


**Lightning record page of Enrolled\_Student:**

**Lightning record of Home\_Page\_Default:**

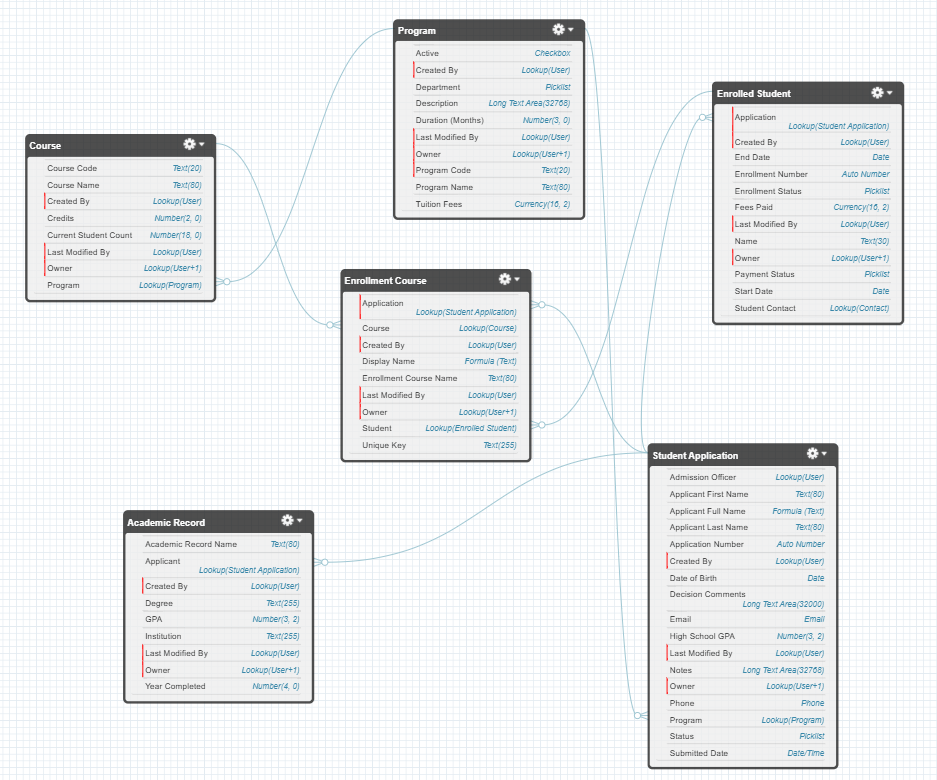


**Compact Layout for mobile:**



* **Relationships:**
* Lookup and Master-Detail relationships:
  + Enrolled\_Student\_\_c → Student\_Application\_\_c : Lookup (required)
  + Enrollment\_Course\_\_c → Enrolled\_Student\_\_c : Lookup
  + Enrollment\_Course\_\_c → Course\_\_c : Lookup
  + Academic\_Records\_\_c → Student\_\_c : Lookup
  + Academic\_Records\_\_c → Course\_\_c : Lookup
* **Junction Objects:**
* Enrollment\_Course\_\_c enables many-to-many tracking between students and courses.

**Schema Builder :**

****

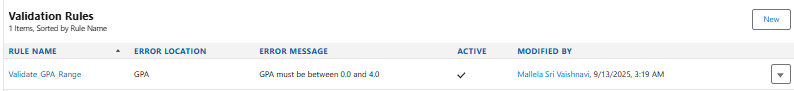
**Phase 4 (Process Automation (Admin)):**

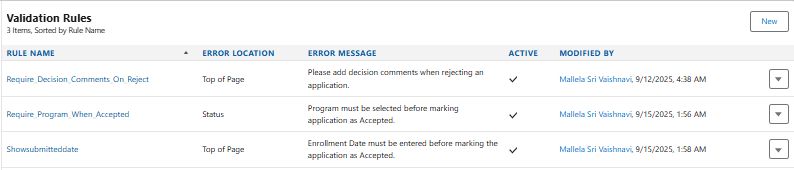
**Objective:** Automate repetitive administrative tasks in the admission lifecycle to ensure accuracy, save time, and provide a seamless applicant experience.

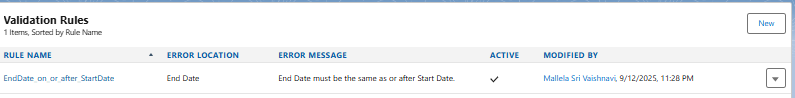
**Validation Rules:**

Ensured data quality and consistency in admission workflow.

* Implemented rules in **Student\_Application\_\_c**, **Academic\_Record\_\_c**, **Enrollment\_Course** object:
  1. **Require\_Decision\_Comments\_On\_Reject** – Prevents rejecting an application unless decision comments are added.
     + Example: If Admission Officer tries to reject without filling **Comments\_\_c**, an error appears.
  2. **Require\_Program\_When\_Accepted** – Forces the officer to select a **Program\_\_c** before marking the application as Accepted.
     + Example: Officer cannot approve without assigning a program.
  3. **Showsubmitteddate** – Ensures **Enrollment\_Date\_\_c** is entered before marking the application as Accepted.
     + Example: Students cannot be accepted without enrollment date filled.
  4. **Validate\_GPA\_Range** – Ensures GPA entered is between **0.0 and 4.0**.
     + Example: Prevents saving record if GPA = 4.5 or GPA = -1.
  5. **EndDate\_on\_or\_after\_StartDate --** Used in **Courses / Programs** to validate date ranges.
* Example: Prevents creating a course/program with an **End Date** earlier than **Start Date**.

****

****

****

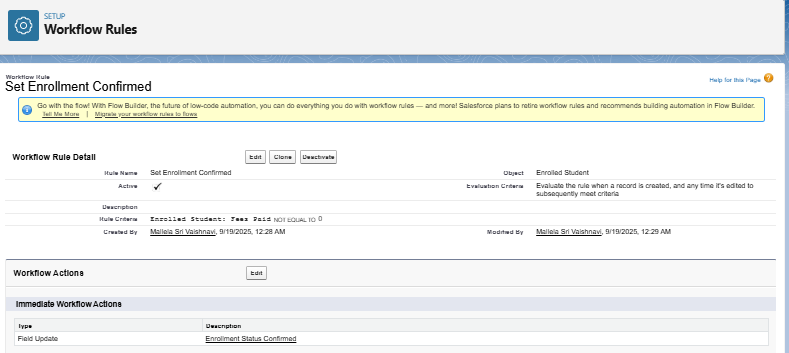
**Workflow Rules:**

**Set Enrollment Status to Confirmed**

* **Objective**: Automatically update a student’s enrollment status once the admission fee is paid.
* **Object**: Enrolled\_Student\_\_c
* **Evaluation Criteria**: Rule runs when a record is created, and every time it’s edited to meet the condition.
* **Rule Criteria**:
  + Fee\_Paid\_\_c = TRUE

**Field Update Action: Enrollment Status Confirmed**

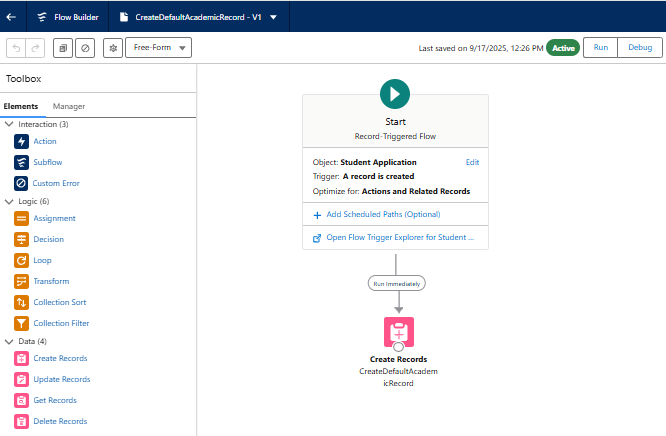
* **Action Name**: Enrollment Status Confirmed
* **Field to Update**: Enrollment\_Status\_\_c
* **New Value**: **Confirmed**



**Flow Builder:**

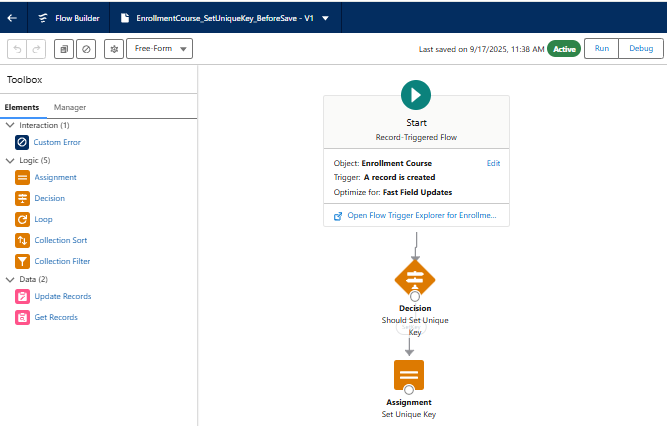
1) **CreateDefaultAcademicRecord (Autolaunched Flow)**

* Automatically creates a default academic record when a new student application is submitted.



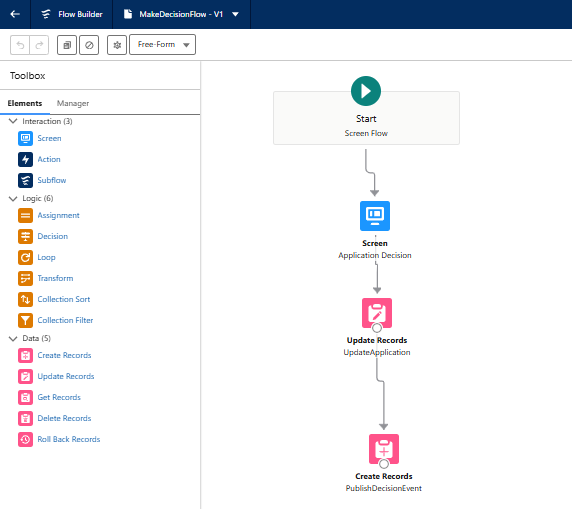
2) **EnrollmentCourse\_SetUniqueKey\_BeforeSave (Autolaunched Flow)**

* Generates a unique key for each enrollment-course record before save.
* Ensures data integrity and prevents duplicates.



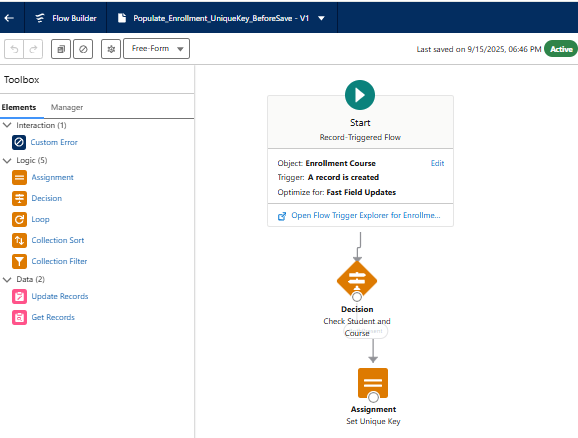
3) **MakeDecisionFlow (Screen Flow)**

* Guided screen flow for admission officers to mark applications as Accepted, Rejected, or Waitlisted.
* Collects decision comments and updates the record.



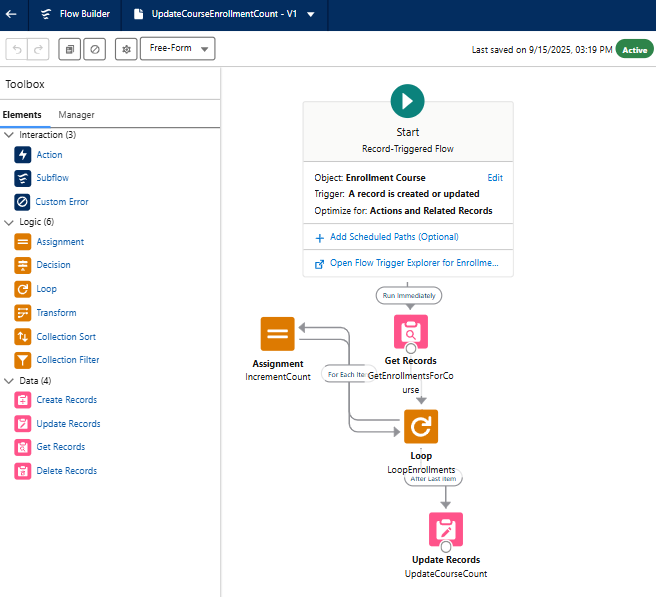
4) **Populate\_Enrollment\_UniqueKey\_BeforeSave (Autolaunched Flow)**

* Populates a unique identifier on enrollment records at save time.



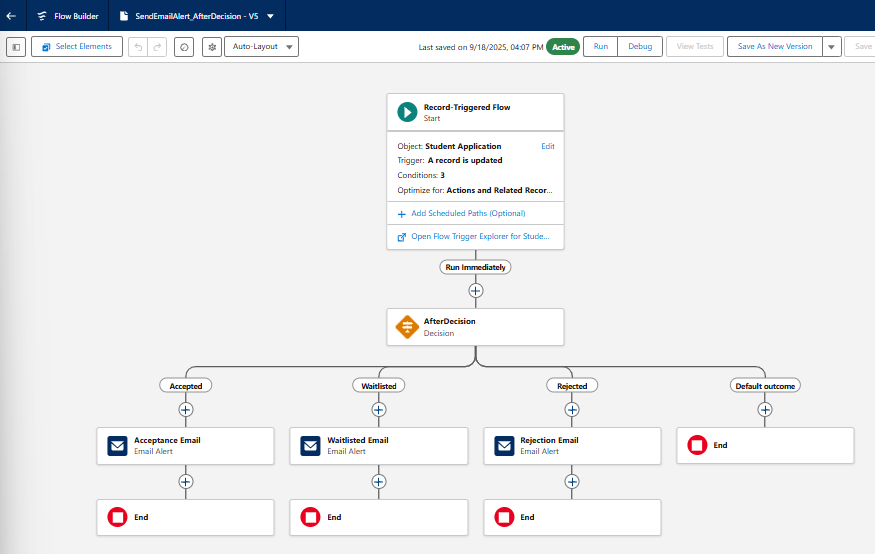
5) **UpdateCourseEnrollmentCount (Autolaunched Flow)**

* Updates the number of enrolled students whenever enrollment changes.
* Useful for reporting and capacity management.



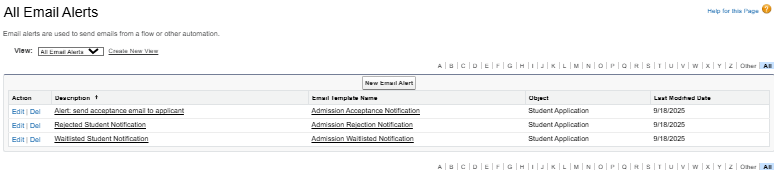
6) **SendEmailAlert\_AfterDecision (Autolaunched Flow)**

* Sends an **acceptance / rejection / waitlist** email to applicants after a decision is made.



**Email Templates in Project:**

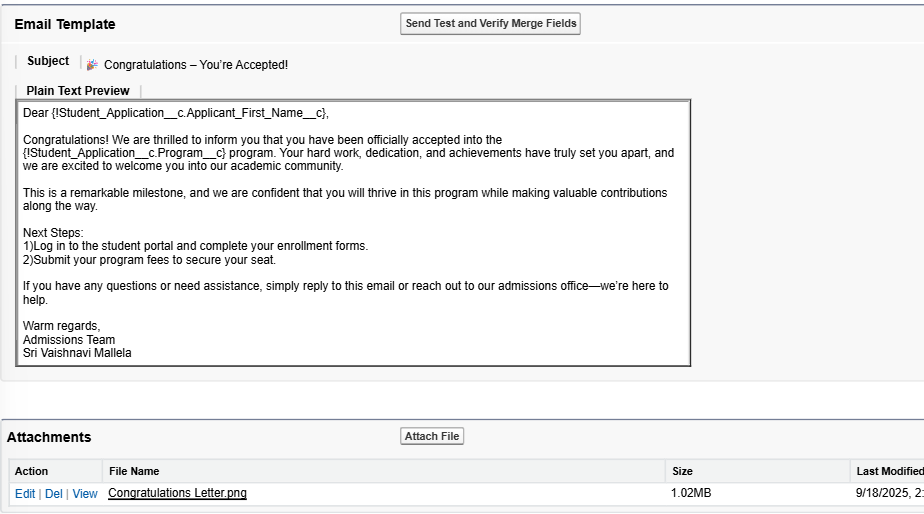
To ensure timely and consistent communication with applicants, **Email Templates** were configured and connected to Flows/Email Alerts.



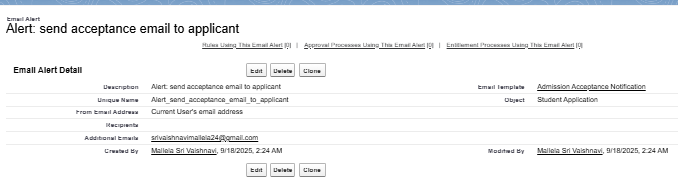
**Implemented Templates**

* **Admission Acceptance Notification**
  + **Type:** Text
  + **Purpose:** Sent automatically when an applicant is marked **Accepted**

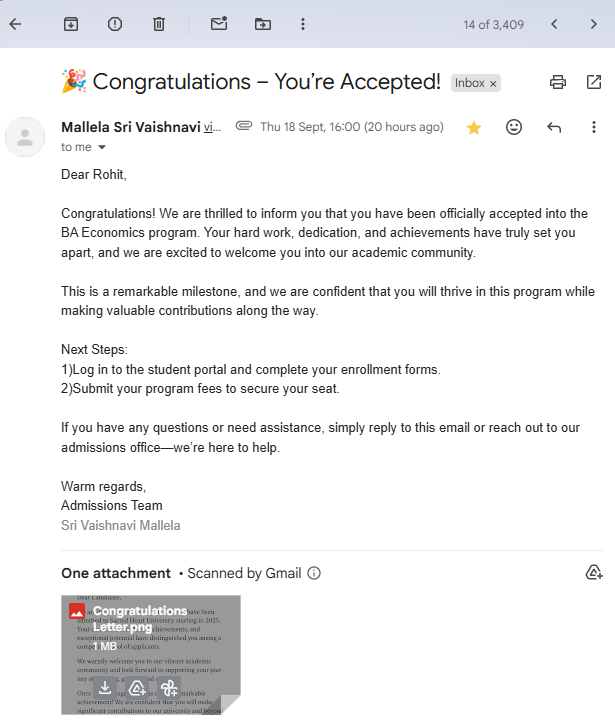
**Email Template:**



**Email Alert:**

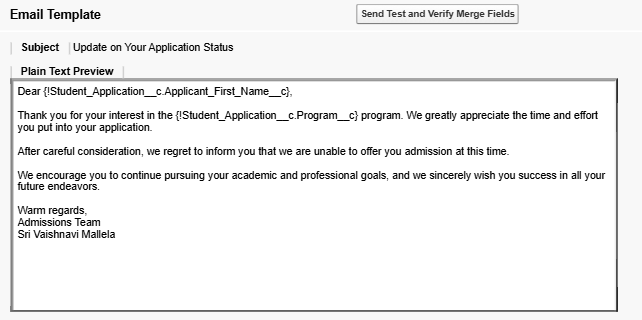


**Email Notification:**

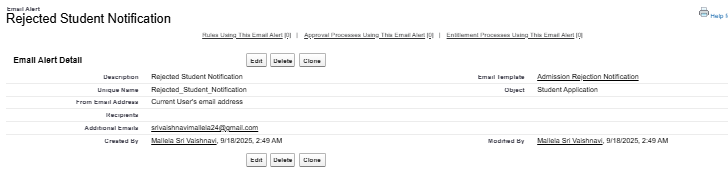


* **Admission Rejection Notification**
  + **Type:** Text
  + **Purpose:** Sent when an applicant’s status changes to **Rejected**.

**Email Template:**



**Email Alert:**

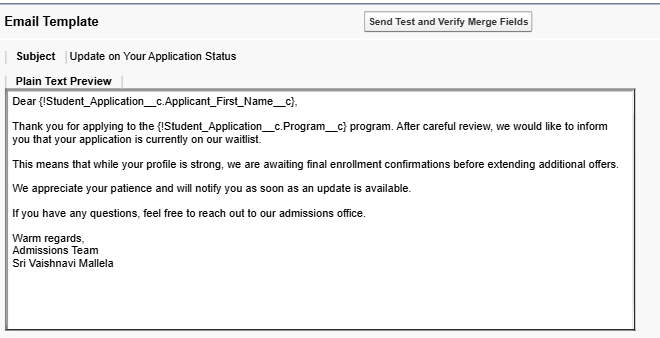


**Email Notification:**

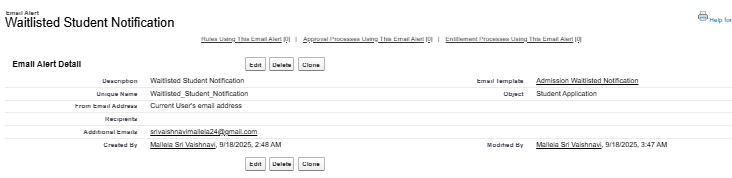


* **Admission Waitlisted Notification**
  + **Type:** Text
  + **Purpose:** Sent when an applicant is **Waitlisted**, giving transparency to students about their status.

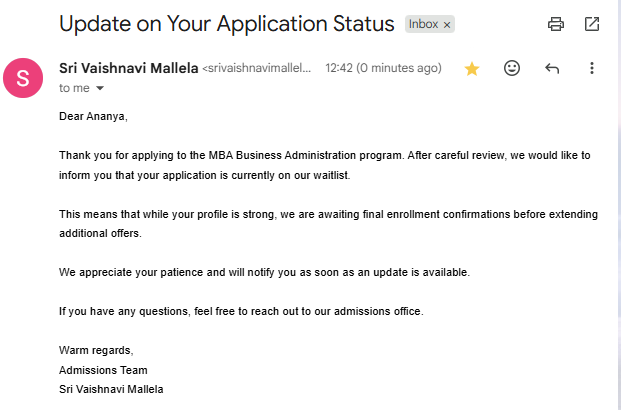
**Email Template:**



**Email Alert:**

****

**Email Notification:**

****

**Field Updates:**

* **Enrollment Status Confirmed**
* **Action Name**: **Enrollment Status Confirmed**
* **Object**: Enrolled\_Student\_\_c
* **Field to Update**: Enrollment\_Status\_\_c
* **Operation**: Update field value
* **New Value**: **Completed**

**Purpose:**

* Ensures that once the workflow condition (e.g., Fee Paid = true) is satisfied, the student’s enrollment record is automatically updated.
* Reduces manual intervention for the admissions team.

