Salesforce Student Admission Management System

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.

Phase 1 (Problem Understanding & Industry Analysis):

Problem Understanding & Industry Analysis

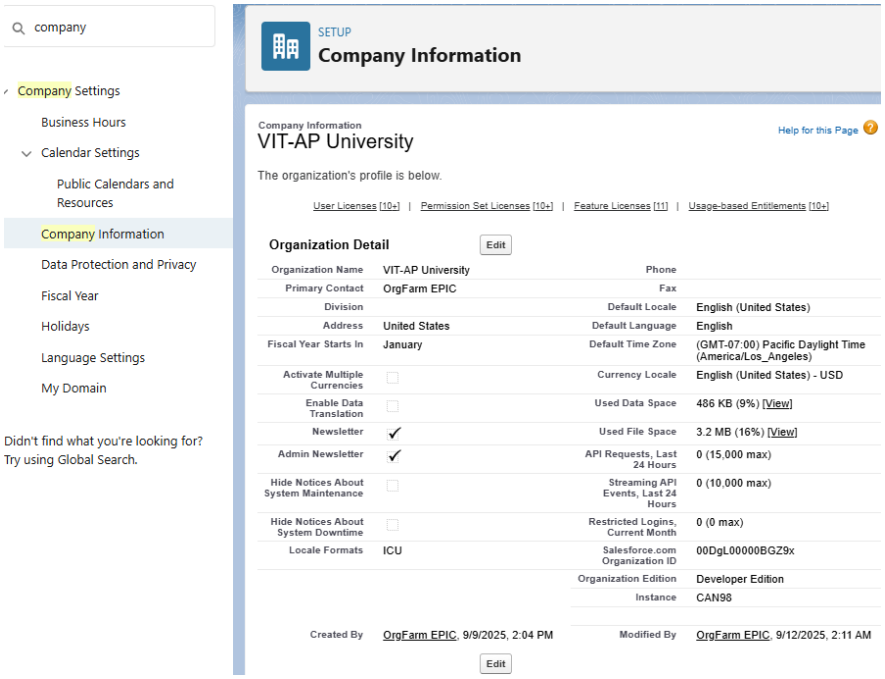
- **Requirement Gathering**
  Collected requirements from the perspective of a university admissions team: online student application, review workflow, enrollment tracking, and reporting needs.
- **Stakeholder Analysis**
  Identified stakeholders such as Admission Officers, Faculty Reviewers, Students/Applicants, 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):

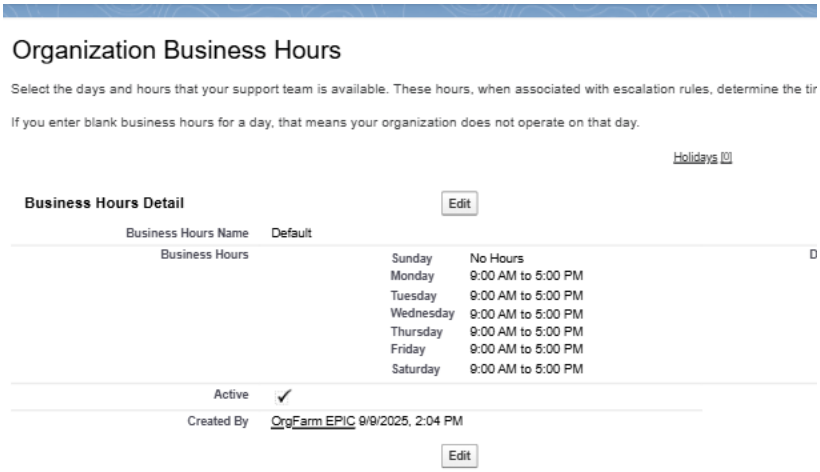
In this phase, the Salesforce Developer Org was configured to align with the requirements of the Student Admission Management System. Key configuration activities included:

- **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.

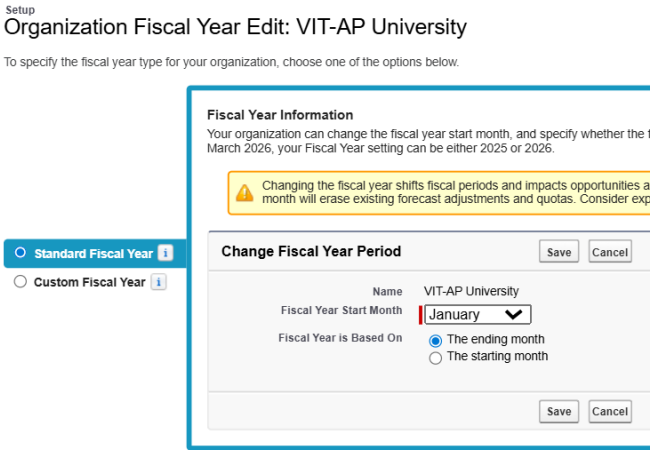


- **Business Hours & Holidays**
  Defined admission office working hours (9 to 5 and no working hours on sundays) and academic holidays can also be included to be used in workflows (like application submission SLAs or automated responses)



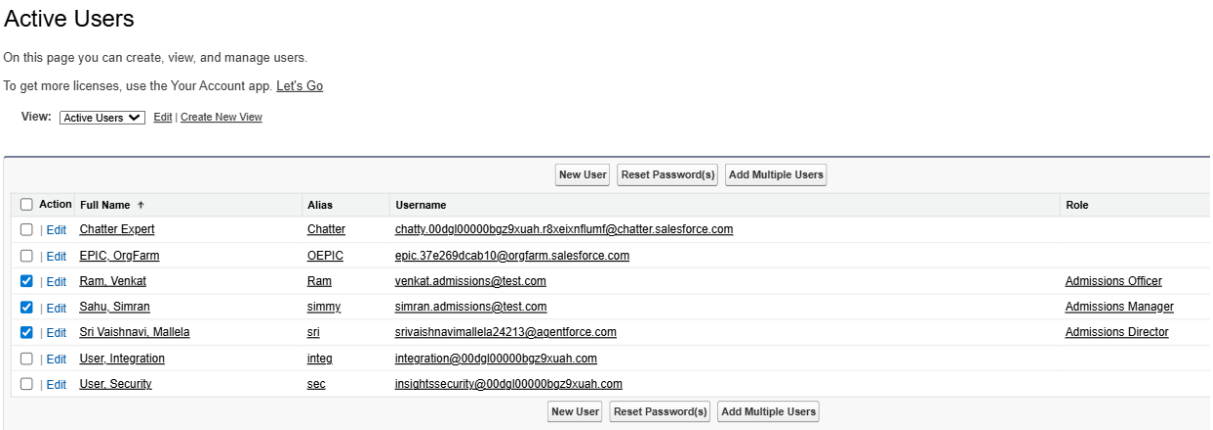
Fiscal Year Settings

Configured fiscal year settings to align with the **academic calendar** for reporting admission data semester/annual basis.



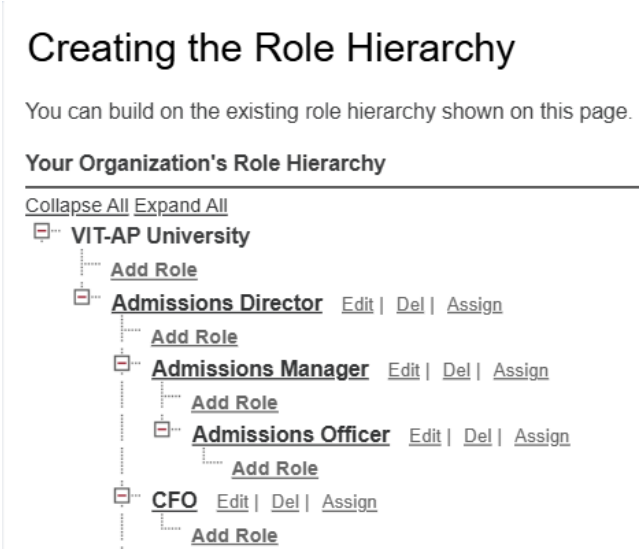
User Setup & Licenses

- Me (Mallela Sri Vaishnavi) →Admissions Director → Full access
- Simran Sahu (Admissions Manager) → Salesforce License
- Venkat Ram (Admissions Officer) → Salesforce License



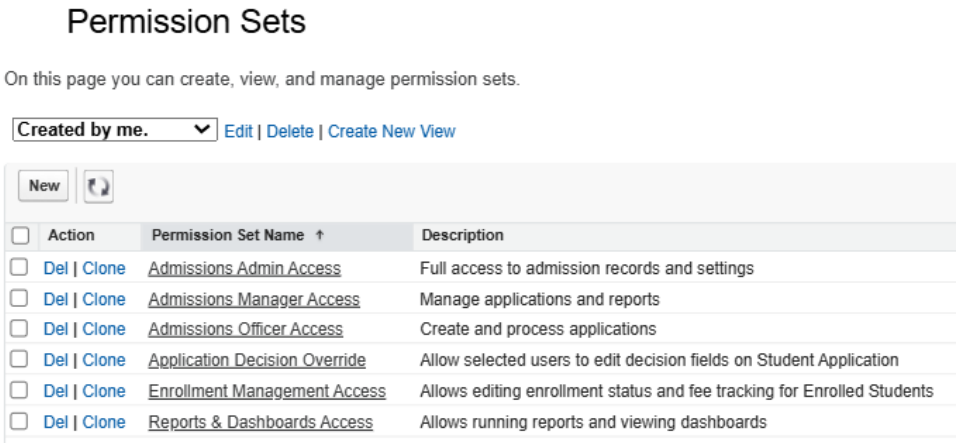
Profiles & Roles

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

- Permission Sets provide extra access without changing user profiles.



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)

SETUP

Sharing Settings

Academic Record	Public Read/Write	Private	✓
Course	Public Read/Write	Private	✓
Enrolled Student	Public Read/Write	Private	✓
Enrollment Course	Public Read/Write	Private	✓
Program	Public Read/Write	Private	✓
Student Application	Public Read/Write	Private	✓

Login Access Policies

- Enabled Administrators can Log in as Any User.
- Allows Admin to test features as Admin (Director), Manger, Officer.

Login Access Policies

Control which support organizations your users can grant login access to.

Changes Saved

Manage Support Options Save Cancel

Setting	Enabled
Administrators Can Log in as Any User	☑

Dev Org Setup

Installed Salesforce CLI and linked VS Code to the Dev Org for **metadata deployment and retrieval**.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS HISTORY

```
PS C:\Users\HP\OneDrive\Desktop\Salesforce crm\StudentAdmissionCRM> sfdx force:org:list
```

	Alias	Username	Org Id	Status
	MySanbox	srivaishnavimallela24213@agentforce.com	00DgL00000BGZ9xUAH	Connected

Deployment Basics

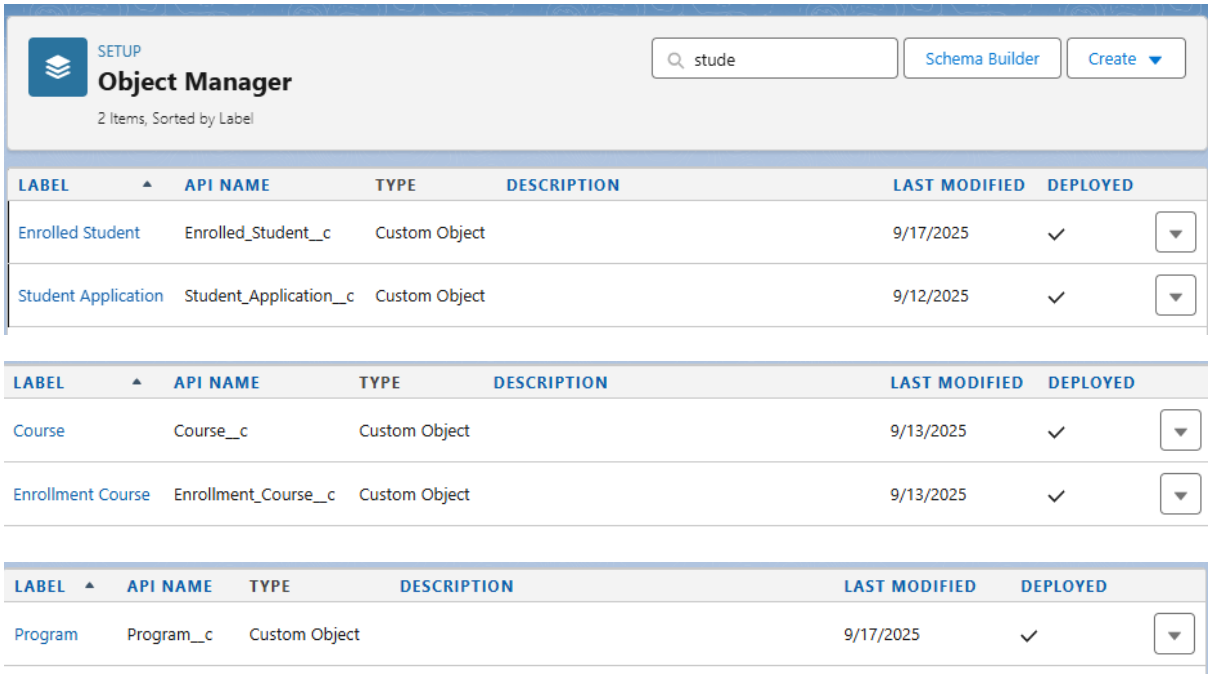
Learned deployment methods:

- **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 course
  - 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



LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Academic Record	Academic_Record__c	Custom Object		9/13/2025	✓

Fields:

- Defined fields for each object using appropriate data types as follows:

Student_Application__c:

Fields & Relationships
17 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE
Admission Officer	Admission_Officer__c	Lookup(User)
Applicant First Name	Applicant_First_Name__c	Text(80)
Applicant Full Name	Applicant_Full_Name__c	Formula (Text)
Applicant Last Name	Applicant_Last_Name__c	Text(80)
Application Number	Name	Auto Number
Created By	CreatedById	Lookup(User)
Date of Birth	Date_of_Birth__c	Date
Decision Comments	Decision_Comments__c	Long Text Area(32000)
Email	Email__c	Email
High School GPA	High_School_GPA__c	Number(3, 2)
Last Modified By	LastModifiedById	Lookup(User)
Notes	Notes__c	Long Text Area(32768)
Owner	OwnerId	Lookup(User,Group)
Phone	Phone__c	Phone
Program	Program__c	Lookup(Program)
Status	Status__c	Picklist
Submitted Date	Submitted_Date__c	Date/Time

Enrolled__Student__c:

Fields & Relationships

12 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE
Application	Application__c	Lookup(Student Application)
Created By	CreatedById	Lookup(User)
End Date	End_Date__c	Date
Enrollment Number	Name	Auto Number
Enrollment Status	Enrollment_Status__c	Picklist
Fees Paid	Fees_Paid__c	Currency(16, 2)
Last Modified By	LastModifiedById	Lookup(User)
Name	Name__c	Text(30)
Owner	OwnerId	Lookup(User,Group)
Payment Status	Payment_Status__c	Picklist
Start Date	Start_Date__c	Date
Student Contact	Student_Contact__c	Lookup(Contact)

Course__c:

Fields & Relationships

8 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE
Course Code	Course_Code__c	Text(20)
Course Name	Name	Text(80)
Created By	CreatedById	Lookup(User)
Credits	Credits__c	Number(2, 0)
Current Student Count	Current_Student_Count__c	Number(18, 0)
Last Modified By	LastModifiedById	Lookup(User)
Owner	OwnerId	Lookup(User,Group)
Program	Program__c	Lookup(Program)

Program__c:

Fields & Relationships
10 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE
Active	Active__c	Checkbox
Created By	CreatedById	Lookup(User)
Department	Department__c	Picklist
Description	Description__c	Long Text Area(32768
Duration (Months)	Duration__c	Number(3, 0)
Last Modified By	LastModifiedById	Lookup(User)
Owner	OwnerId	Lookup(User,Group)
Program Code	Program_Code__c	Text(20)
Program Name	Name	Text(80)
Tuition Fees	Tuition_Fees__c	Currency(16, 2)

Academic_Record__c:

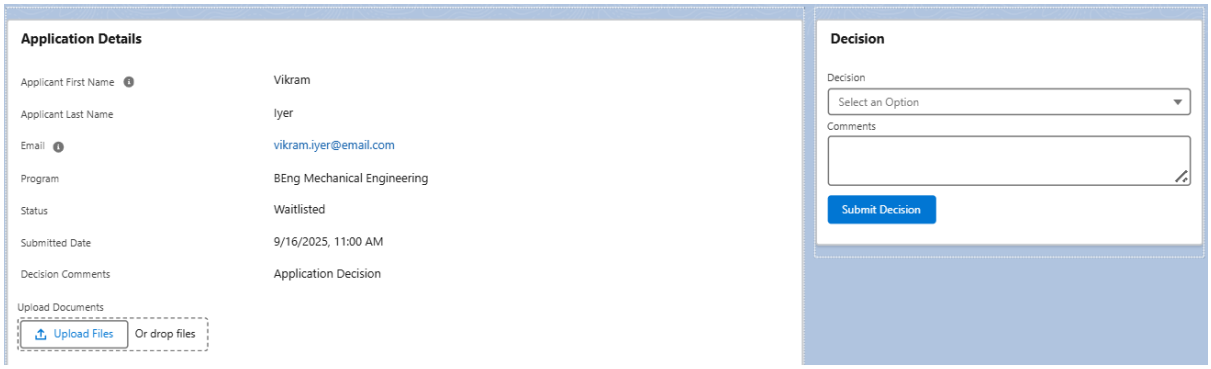
Fields & Relationships
9 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE
Academic Record Name	Name	Text(80)
Applicant	Applicant__c	Lookup(Student Application)
Created By	CreatedById	Lookup(User)
Degree	Degree__c	Text(255)
GPA	GPA__c	Number(3, 2)
Institution	Institution__c	Text(255)
Last Modified By	LastModifiedById	Lookup(User)
Owner	OwnerId	Lookup(User,Group)
Year Completed	Year_Completed__c	Number(4, 0)

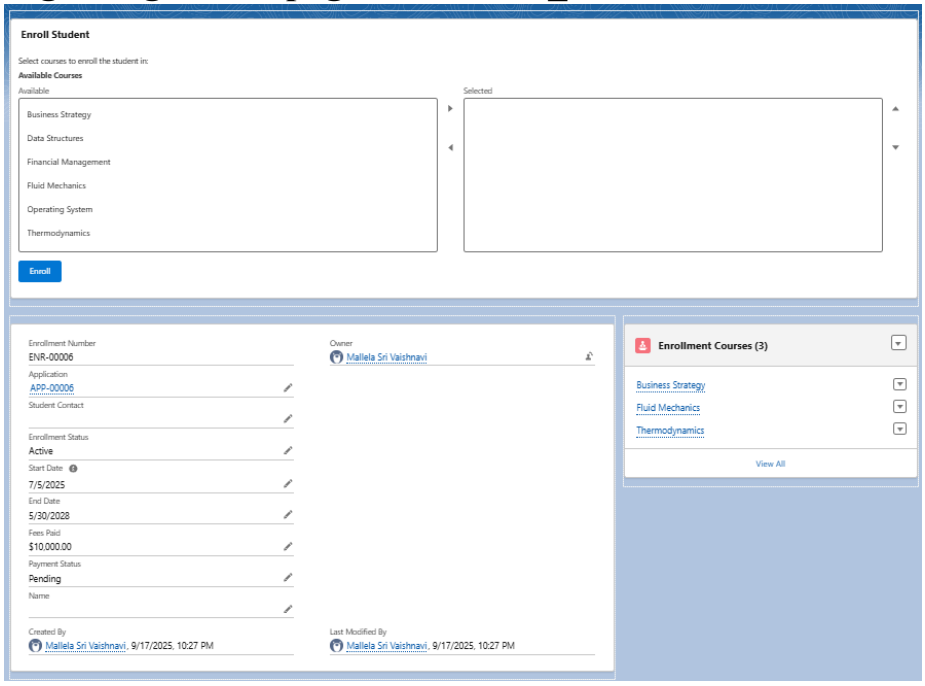
Record Types & Page Layouts:

- 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

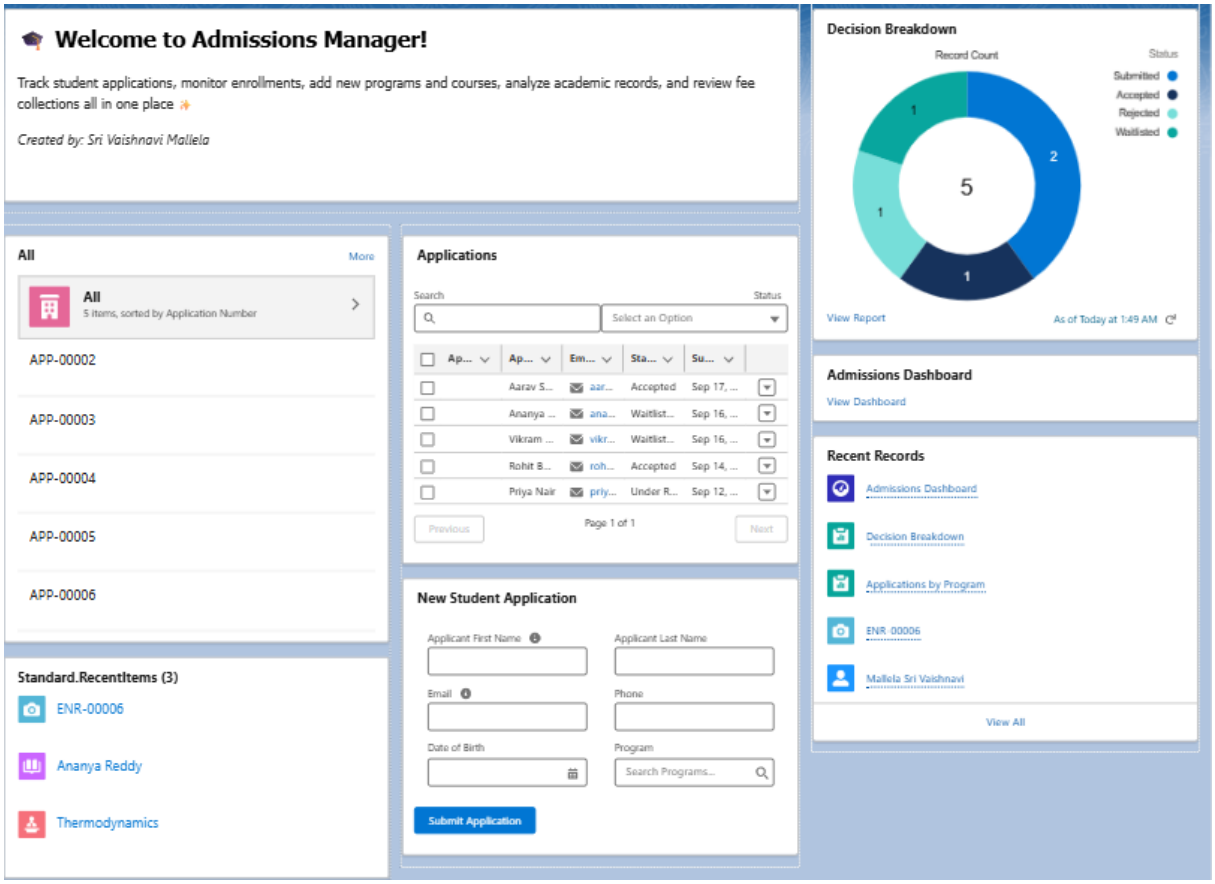
Lightning record page of Student_Application:



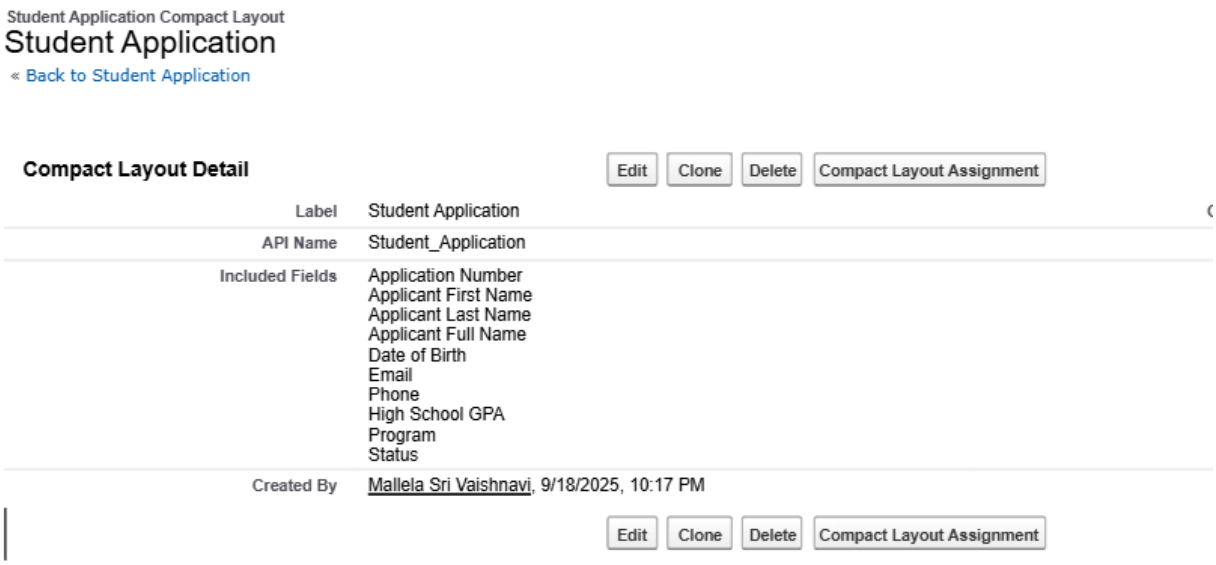
Lightning record page of Enrolled_Student:



Lightning record of Home_Page_Default:



Compact Layout for mobile:



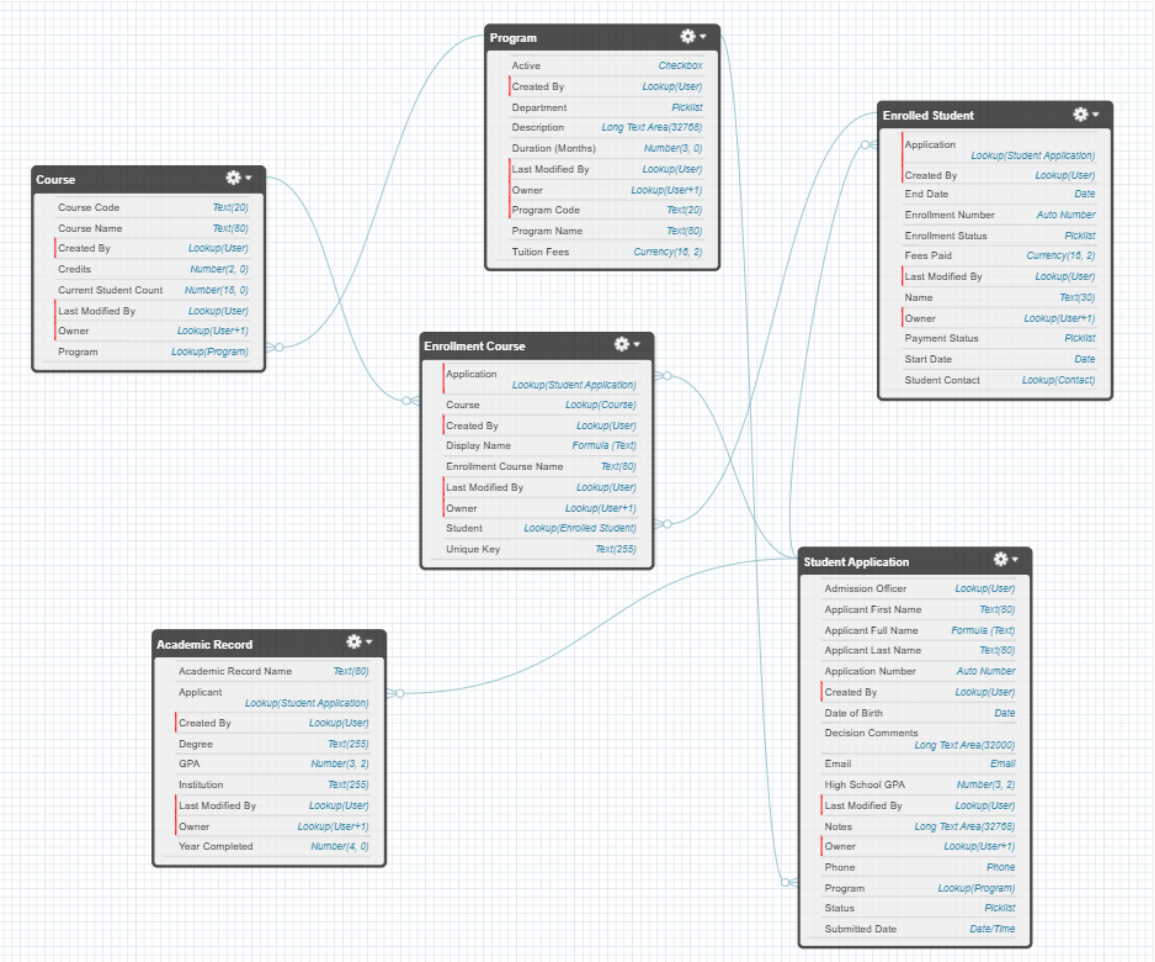
Relationships:

- 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**.

Validation Rules
1 Items, Sorted by Rule Name

New

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Validate_GPA_Range	GPA	GPA must be between 0.0 and 4.0	✓	Mallela Sri Vaishnavi, 9/13/2025, 3:19 AM

Validation Rules
3 Items, Sorted by Rule Name

New

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Require_Decision_Comments_On_Reject	Top of Page	Please add decision comments when rejecting an application.	✓	Mallela Sri Vaishnavi, 9/12/2025, 4:38 AM
Require_Program_When_Accepted	Status	Program must be selected before marking application as Accepted.	✓	Mallela Sri Vaishnavi, 9/15/2025, 1:56 AM
Showsubmitteddate	Top of Page	Enrollment Date must be entered before marking the application as Accepted.	✓	Mallela Sri Vaishnavi, 9/15/2025, 1:58 AM

Validation Rules
1 Items, Sorted by Rule Name

New

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
EndDate_on_or_after_StartDate	End Date	End Date must be the same as or after Start Date.	✓	Mallela Sri Vaishnavi, 9/12/2025, 11:28 PM

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**

SETUP
Workflow Rules

Workflow Rule
Set Enrollment Confirmed

Help for this Page

Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with workflow rules — and more! Salesforce plans to retire workflow rules and recommends building automation in Flow Builder.
Tell Me More | Migrate your workflow rules to flows

Workflow Rule Detail Edit Clone Deactivate

Rule Name	Set Enrollment Confirmed	Object	Enrolled Student
Active	✓	Evaluation Criteria	Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
Description			
Rule Criteria	Enrolled Student: Fees Paid NOT EQUAL TO 0		
Created By	Mallela Sri Vaishnavi, 9/19/2025, 12:28 AM	Modified By	Mallela Sri Vaishnavi, 9/19/2025, 12:29 AM

Workflow Actions Edit

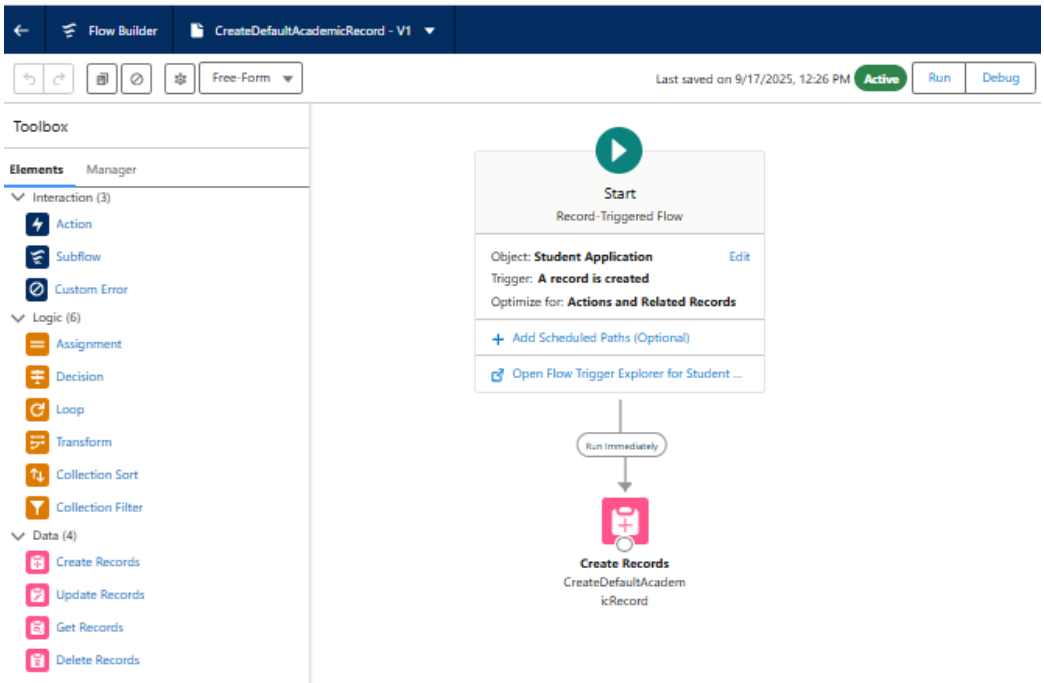
Immediate Workflow Actions

Type	Description
Field Update	Enrollment Status Confirmed

Flow Builder:

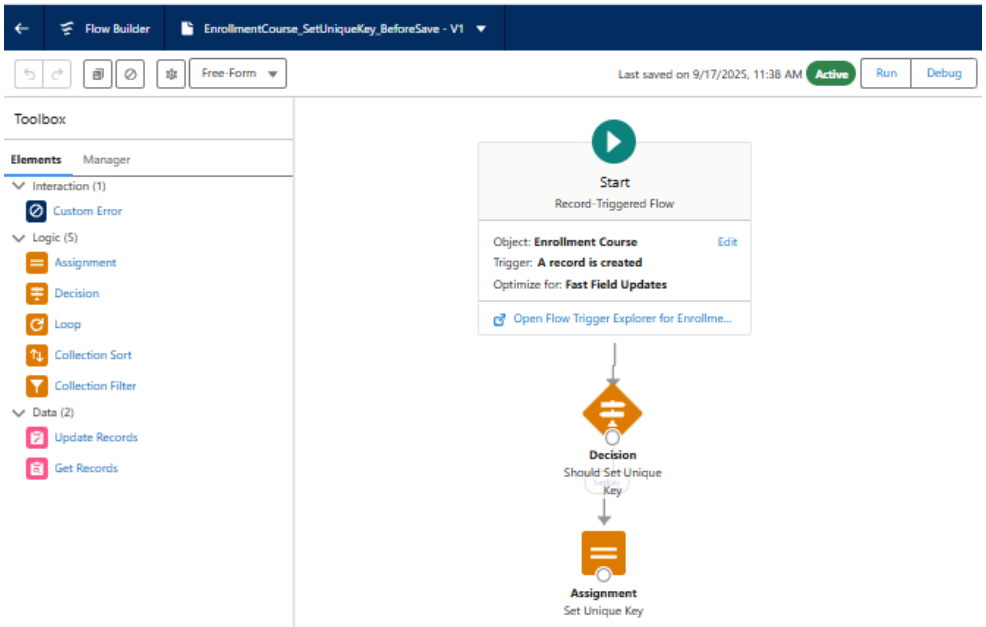
1) CreateDefaultAcademicRecord (Autolaunched Flow)

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



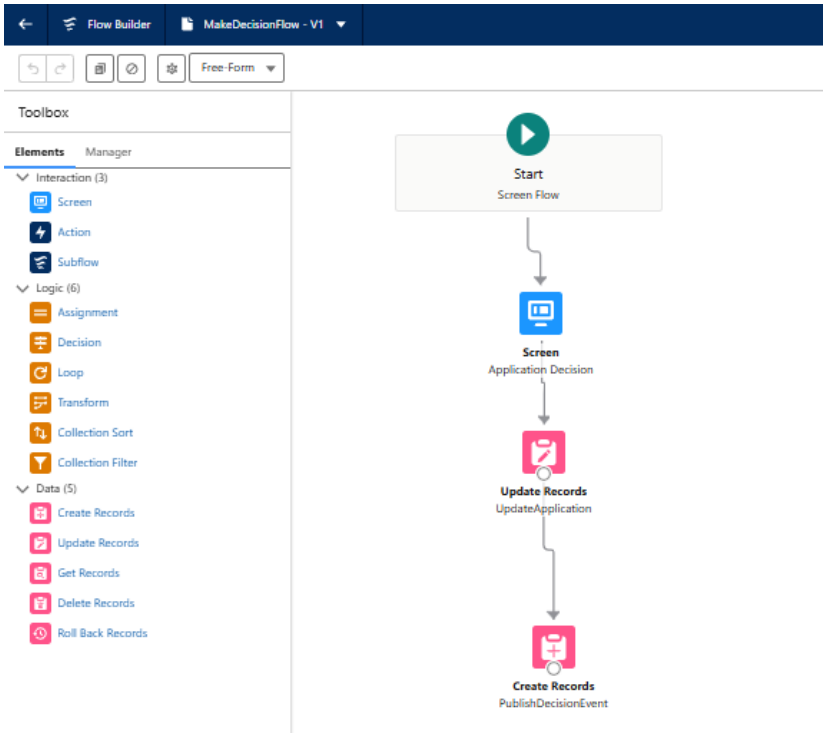
2) EnrollmentCourse_SetUniqueKey_BeforeSave (Autolaunched Flow)

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



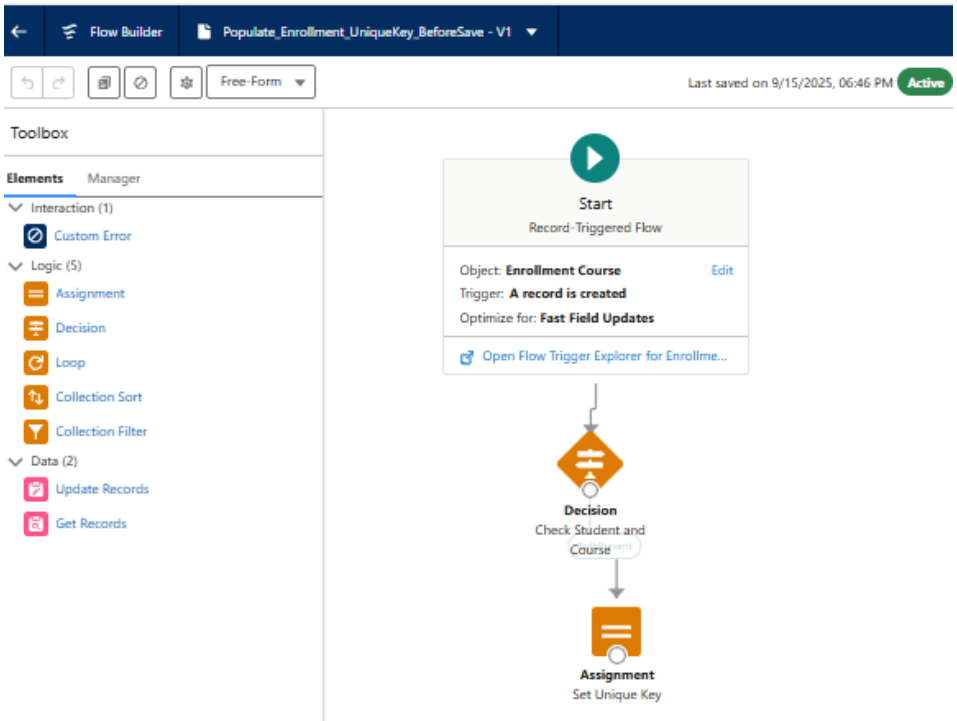
3) MakeDecisionFlow (Screen Flow)

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



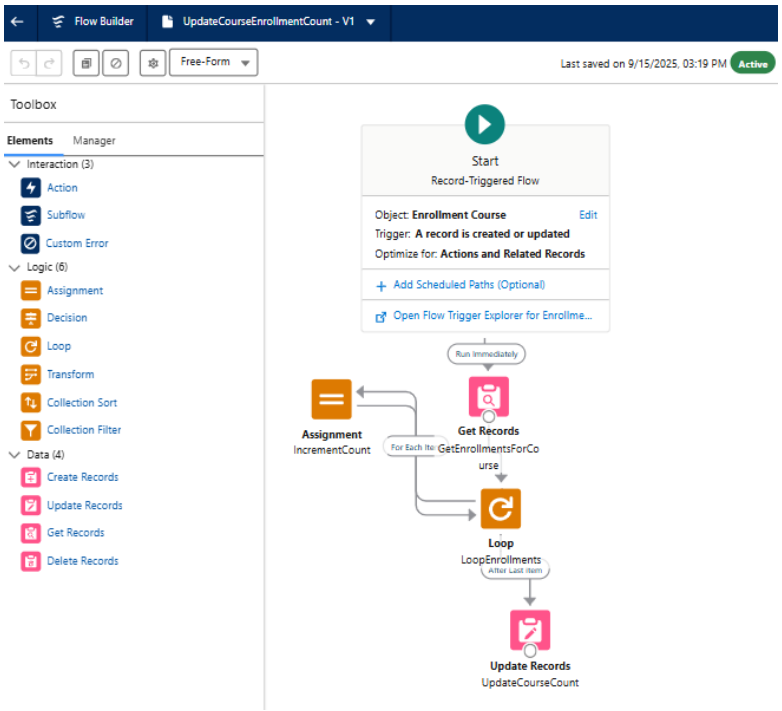
4) Populate_Enrollment_UniqueKey_BeforeSave (Autolaunched Flow)

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



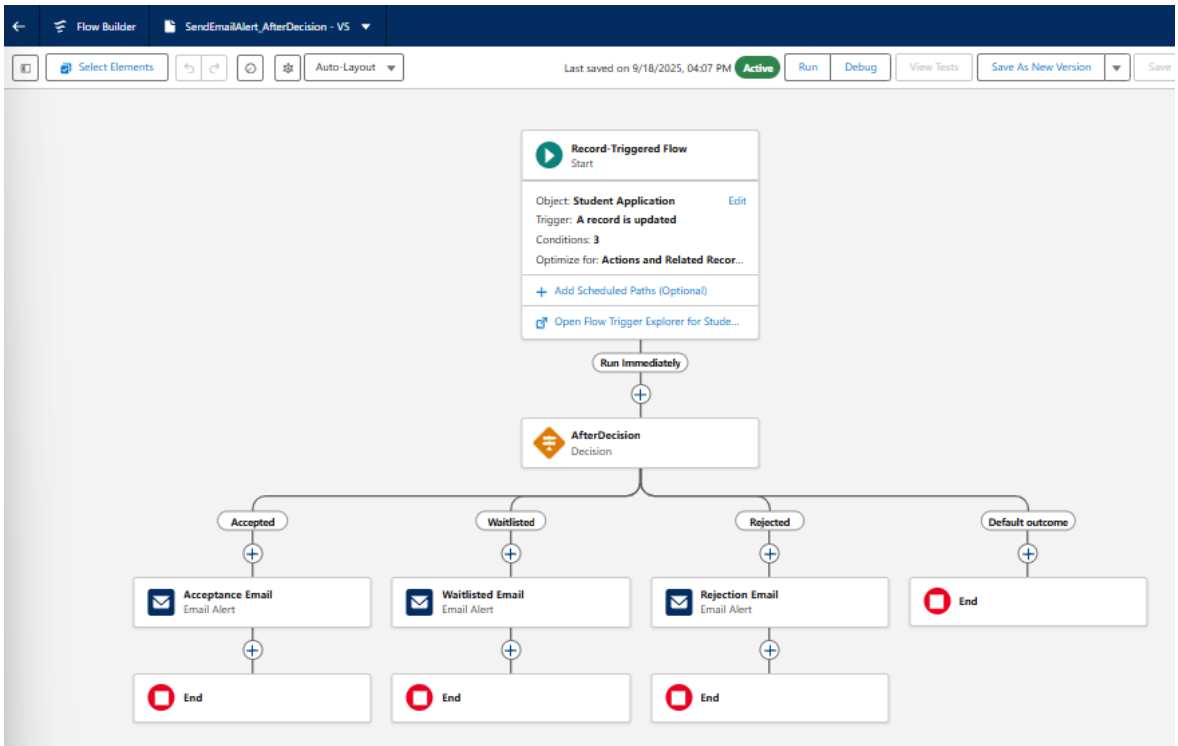
5) UpdateCourseEnrollmentCount (Autolaunched Flow)

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



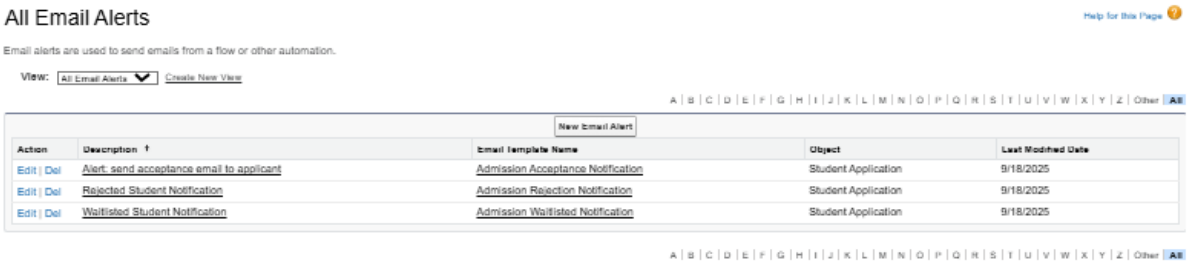
6) SendEmailAlert_AfterDecision (Autolaunched Flow)

- 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.

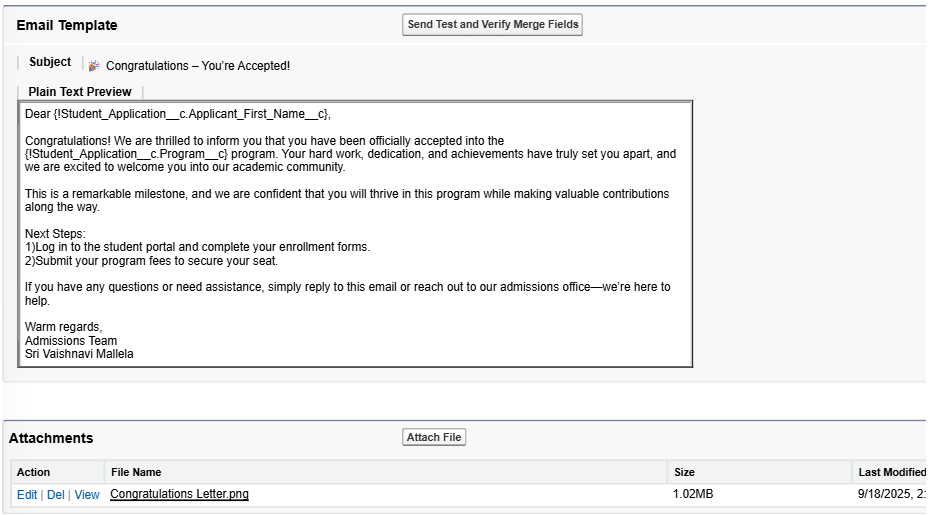


Action	Description	Email Template Name	Object	Last Modified Date
Edit | Del	Alert: send acceptance email to applicant	Admission Acceptance Notification	Student Application	9/18/2025
Edit | Del	Rejected Student Notification	Admission Rejection Notification	Student Application	9/18/2025
Edit | Del	Waitlisted Student Notification	Admission Waitlisted Notification	Student Application	9/18/2025

Implemented Templates

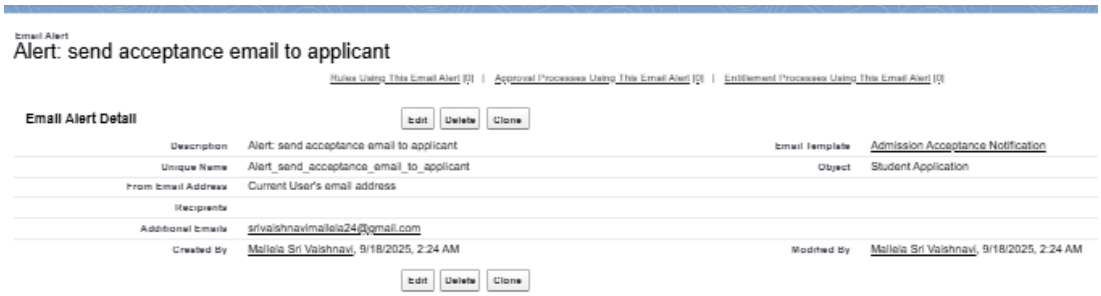
- **Admission Acceptance Notification**
  - **Type:** Text
  - **Purpose:** Sent automatically when an applicant is marked **Accepted**

Email Template:

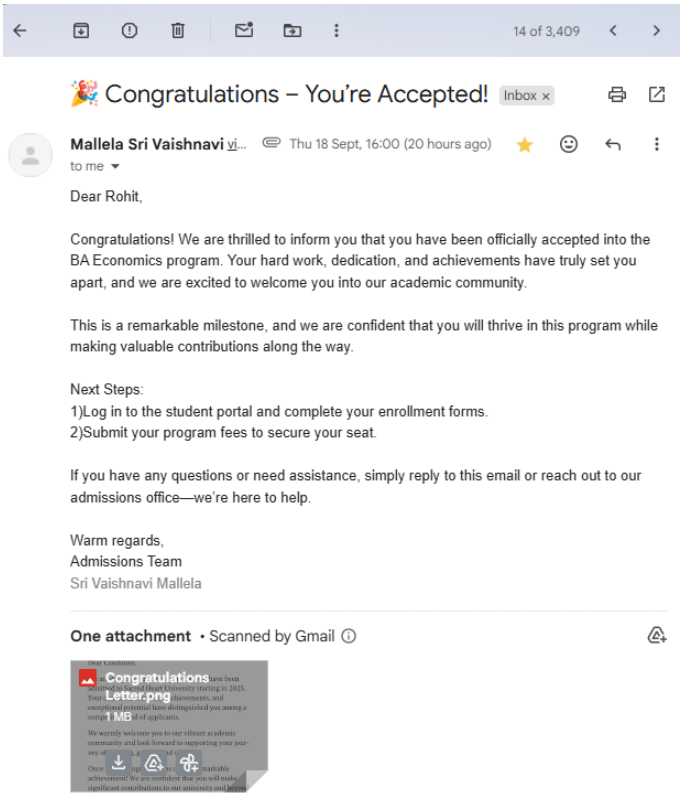


Action	File Name	Size	Last Modified
Edit | Del | View	Congratulations Letter.png	1.02MB	9/18/2025, 2:

Email Alert:



Email Notification:



14 of 3,409

🎉 Congratulations – You're Accepted! Inbox ×

Mallela Sri Vaishnavi vi... Thu 18 Sept, 16:00 (20 hours ago)
to me

Dear Rohit,

Congratulations! We are thrilled to inform you that you have been officially accepted into the BA Economics program. Your hard work, dedication, and achievements have truly set you apart, and we are excited to welcome you into our academic community.

This is a remarkable milestone, and we are confident that you will thrive in this program while making valuable contributions along the way.

Next Steps:
1)Log in to the student portal and complete your enrollment forms.
2)Submit your program fees to secure your seat.

If you have any questions or need assistance, simply reply to this email or reach out to our admissions office—we're here to help.

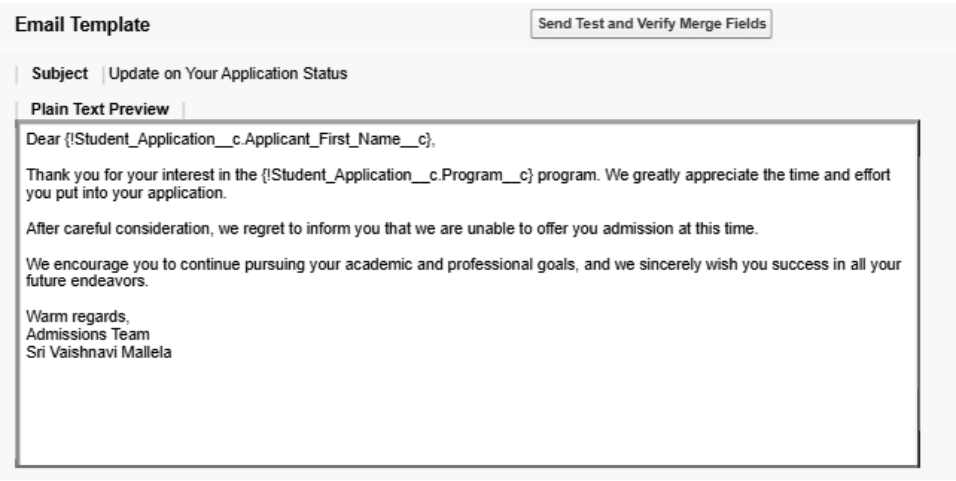
Warm regards,
Admissions Team
Sri Vaishnavi Mallela

One attachment • Scanned by Gmail

Congratulations Letter.png
1 MB

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

Email Template:



Email Template Send Test and Verify Merge Fields

Subject Update on Your Application Status

Plain Text Preview

Dear {!Student_Application__c.Applicant_First_Name__c},

Thank you for your interest in the {!Student_Application__c.Program__c} program. We greatly appreciate the time and effort you put into your application.

After careful consideration, we regret to inform you that we are unable to offer you admission at this time.

We encourage you to continue pursuing your academic and professional goals, and we sincerely wish you success in all your future endeavors.

Warm regards,
Admissions Team
Sri Vaishnavi Mallela

Email Alert:

Email Alert
Rejected Student Notification

Rules Using This Email Alert [0] | Approval Processes Using This Email Alert [0] | Entitlement Processes Using This Email Alert [0]

Email Alert Detail Edit Delete Clone

Description	Rejected Student Notification	Email Template	Admission Rejection Notification
Unique Name	Rejected_Student_Notification	Object	Student Application
From Email Address	Current User's email address		
Recipients			
Additional Emails	srivaishnavimallela24@gmail.com		
Created by	Mallela Sri Vaishnavi, 9/18/2025, 2:49 AM	Modified by	Mallela Sri Vaishnavi, 9/18/2025, 2:49 AM

Edit Delete Clone

Email Notification:

Update on Your Application Status Inbox ×

Mallela Sri Vaishnavi via 2gp... Thu 18 Sept, 16:12 (20 hours ago)
to me

Dear Vikram,

Thank you for your interest in the BEng Mechanical Engineering program. We greatly appreciate the time and effort you put into your application.

After careful consideration, we regret to inform you that we are unable to offer you admission at this time.

We encourage you to continue pursuing your academic and professional goals, and we sincerely wish you success in all your future endeavors.

Warm regards,
Admissions Team
Sri Vaishnavi Mallela

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

Email Template:

Email Template Send Test and Verify Merge Fields

Subject Update on Your Application Status

Plain Text Preview

Dear {!Student_Application__c.Applicant_First_Name__c},

Thank you for applying to the {!Student_Application__c.Program__c} program. After careful review, we would like to inform you that your application is currently on our waitlist.

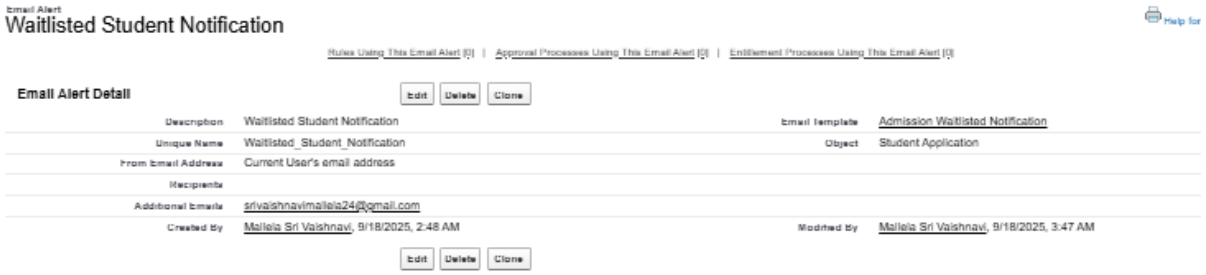
This means that while your profile is strong, we are awaiting final enrollment confirmations before extending additional offers.

We appreciate your patience and will notify you as soon as an update is available.

If you have any questions, feel free to reach out to our admissions office.

Warm regards,
Admissions Team
Sri Vaishnavi Mallela

Email Alert:



Email Alert
Waitlisted Student Notification

Help for

Rules Using This Email Alert (0) | Approval Processes Using This Email Alert (0) | Entitlement Processes Using This Email Alert (0)

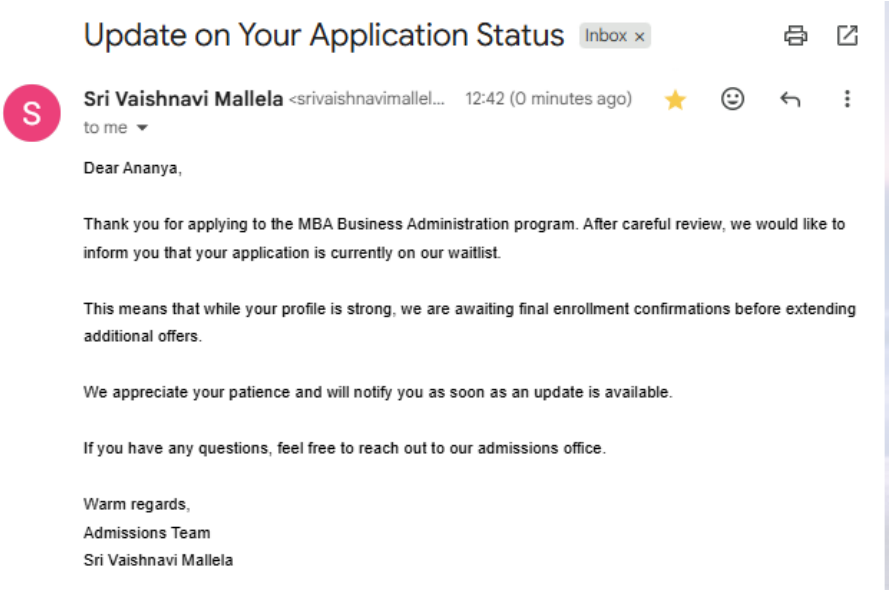
Email Alert Detail

Edit | Delete | Clone

Description	Waitlisted Student Notification	Email Template	Admission Waitlisted Notification
Unique Name	Waitlisted_Student_Notification	Object	Student Application
From Email Address	Current User's email address		
Recipients			
Additional Emails	srivaishnavimallela24@gmail.com		
Created By	Mallela Sri Vaishnavi, 9/18/2025, 2:48 AM	Modified By	Mallela Sri Vaishnavi, 9/18/2025, 3:47 AM

Edit | Delete | Clone

Email Notification:



Update on Your Application Status Inbox ×

Sri Vaishnavi Mallela <srivaishnavimallel... 12:42 (0 minutes ago)
to me

Dear Ananya,

Thank you for applying to the MBA Business Administration program. After careful review, we would like to inform you that your application is currently on our waitlist.

This means that while your profile is strong, we are awaiting final enrollment confirmations before extending additional offers.

We appreciate your patience and will notify you as soon as an update is available.

If you have any questions, feel free to reach out to our admissions office.

Warm regards,
Admissions Team
Sri Vaishnavi Mallela

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.

SETUP
Field Updates

Field Update
Enrollment Status Confirmed

Help for this Page

Rules Using This Field Update | Approval Processes Using This Field Update | Entitlement Processes Using This Field Update

Field Update Detail Edit Delete

Name	Enrollment Status Confirmed
Unique Name	Enrollment_Status_Confirmed
Description	
Object	Enrolled Student
Field to Update	Enrolled Student: Enrollment Status
Field Data Type	Picklist
Re-evaluate Workflow Rules after Field Change	☐
New Field Value	Completed

Edit Delete

Rules Using This Field Update Rules Using This Field Update Help

Action	Rule Name	Description	Object	Active
Edit | Del | Deactivate	Set Enrollment Confirmed		Enrolled Student	✓

Custom Notifications:

Weekly dashboard subscriptions (Email notification on every Sunday at 4:00 PM) were set up to show email key admission metrics dashboards (applications, decisions, enrollments) to Admissions team members.

Edit Subscription

Schedule dashboard refreshes and subscribe to receive results.

Settings

Frequency

Daily | Weekly | Monthly

Days

Sun | Mon | Tue | Wed | Thu | Fri | Sat

Time

4:00 PM

Recipients

⚠ Recipients see the same report data as the person running the report.

☑ Receive new results by email when dashboard is refreshed.

Send email to

Me, OrgFarm EPIC

Edit Recipients

Unsubscribe Cancel Save

Custom Email Dashboard Subscription Notification:

Dashboard refreshed (Admissions Dashboard) Inbox ×

Mallela Sri Vaishnavi via ytfd12y5b35k.gl-bgz9xuah.can98.bnc.salesforce.com 05:31 (16 hours ago)
to me, epic.orgfarm@salesforce.com

Mallela,
Here is the dashboard you're subscribed to.

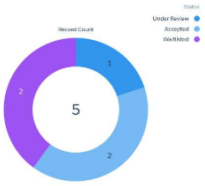
Admissions Dashboard

As of September 21, 2025 at 5:00 PM · Viewing As Mallela Sri Vaishnavi

View Dashboard

Applications by Program

Decision Breakdown



View Report

Total Applicaitons

STUDENT APPLICATION: APPLICATION NUMBER	APPLICANT FULL NAME	STATUS
APP-00002	Aarav Sharma	Accepted
APP-00003	Ananya Reddy	Waitlisted
APP-00004	Vikram Iyer	Waitlisted
APP-00005	Priya Nair	Under Review
APP-00006	Rohit Banerjee	Accepted

View Report

Enrollments per Course

GPA Distribution Report

GPA	ACADEMIC RECORD: ACADEMIC RECORD NAME	GPA RANGE
3.40	Vikram Iyer	Bucket 4
3.50	Rohit Banerjee	Bucket 4
3.60	Aarav Sharma	Bucket 3
3.80	Priya Nair	Bucket 3
3.90	Ananya Reddy	Bucket 3

View Report

Total Fees Collected

ENROLLED STUDENT: ENROLLMENT NUMBER	FEES PAID
ENR-00002	$25,000.00
ENR-00003	$60,000.00
ENR-00004	$20,000.00
ENR-00005	$150,000.00
ENR-00006	$10,000.00
GRAND TOTAL	$265,000.00

View Report

Phase 5 (Apex Programming)- Developer

Classes & Objects:

- **Classes** are blueprints for logic, containing methods and variables.

  In this project, we created:

  - **AdmissionDecisionPublisher** → Publishes admission decisions and updates student application records.

Apex Code:

AdmissionDecisionPublisherTest.apxc | AdmissionDecisionPublisher.apxc | ApplicationListController.apxc | ApplicationListControllerTest.apxc | EnrollmentController.apxc | EnrollmentCont

Code Coverage: None | API Version: 64

```
public with sharing class AdmissionDecisionPublisher {

    @AuraEnabled
    public static void publishDecision(Id studentAppId, String decision, String comments) {
        if (studentAppId == null) {
            throw new AuraHandledException('studentAppId is required');
        }

        // Fetch the application
        Student_Application__c app = [
            SELECT Id, Status__c, Program__c
            FROM Student_Application__c
            WHERE Id = :studentAppId
            LIMIT 1
        ];

        // Optional validation: ensure Program is selected if Accepted
        if (decision == 'Accepted' && app.Program__c == null) {
            throw new AuraHandledException('Program must be selected before marking application as Accepted.')
        }

        // Update the Student Application record
        app.Status__c = decision;
        app.Decision_Comments__c = comments;
        update app;

        // Publish the platform event
        Application_Decision__e evt = new Application_Decision__e(
            Application_Id__c = String.valueOf(studentAppId),
            Decision__c = decision,
            Officer_Id__c = UserInfo.getUserId(),
            Comments__c = comments,
            Decision_Date__c = System.now()
        );
        EventBus.publish(evt);
    }
}
```

Test class for AdmissionDecisionPublisher:

```apex
@IsTest
public class AdmissionDecisionPublisherTest {

    @IsTest
    static void testPublishDecision_updatesRecordAndPublishesEvent() {
        // Create Program
        Program__c prog = new Program__c(Name='Test Program');
        insert prog;

        // Create Student Application
        Student_Application__c app = new Student_Application__c(
            Applicant_First_Name__c = 'Test',
            Applicant_Last_Name__c = 'User',
            Email__c = 'test@example.com',
            Program__c = prog.Id,
            Submitted_Date__c = System.now()
        );
        insert app;

        Test.startTest();
        AdmissionDecisionPublisher.publishDecision(
            app.Id,
            'Accepted',
            'Good performance'
        );
        Test.stopTest();

        Student_Application__c updatedApp = [
            SELECT Id, Status__c, Decision_Comments__c
            FROM Student_Application__c
            WHERE Id = :app.Id
        ];

        System.assertEquals('Accepted', updatedApp.Status__c);
        System.assertEquals('Good performance', updatedApp.Decision_Comments__c);
    }
}
```

- **ApplicationListController** → Provides list view functionality for Student Applications in LWC.

Apex Code:

```apex
public with sharing class ApplicationListController {

    public class PaginatedResult {
        @AuraEnabled public List<Student_Application__c> records;
        @AuraEnabled public Integer totalSize;
    }

    @AuraEnabled(cacheable=true)
    public static PaginatedResult getApplications(Integer pageSize, Integer pageNumber, String statusFilter) {
        if (pageSize == null || pageSize <= 0) pageSize = 10;
        if (pageNumber == null || pageNumber <= 0) pageNumber = 1;
        Integer offsetVal = (pageNumber - 1) * pageSize;

        List<Student_Application__c> apps;
        Integer total = 0;

        if (String.isBlank(statusFilter)) {
            apps = [
                SELECT Id, Applicant_First_Name__c, Applicant_Last_Name__c,
                       Email__c, Status__c, Submitted_Date__c
                FROM Student_Application__c
                ORDER BY Submitted_Date__c DESC
                LIMIT :pageSize OFFSET :offsetVal
            ];
            total = [SELECT COUNT() FROM Student_Application__c];
        } else {
            apps = [
                SELECT Id, Applicant_First_Name__c, Applicant_Last_Name__c,
                       Email__c, Status__c, Submitted_Date__c
                FROM Student_Application__c
                WHERE Status__c = :statusFilter
                ORDER BY Submitted_Date__c DESC
                LIMIT :pageSize OFFSET :offsetVal
            ];
            total = [SELECT COUNT() FROM Student_Application__c WHERE Status__c = :statusFilter];
        }

        PaginatedResult res = new PaginatedResult();
        res.records = apps;
        res.totalSize = total;
        return res;
    }
```

Test class for ApplicationListController:

```
@IsTest
private class ApplicationListControllerTest {

    @IsTest static void testGetApplications_pagination() {
        List<Student_Application__c> apps = new List<Student_Application__c>();
        Program__c prog = new Program__c(Name='Test Program');
        insert prog;

        for (Integer i = 0; i < 12; i++) {
            apps.add(new Student_Application__c(
                Applicant_First_Name__c = 'Fn' + i,
                Applicant_Last_Name__c = 'Ln' + i,
                Email__c = 'test' + i + '@example.com',
                Program__c = prog.Id,
                Status__c = 'Submitted',
                Submitted_Date__c = System.now()
            ));
        }
        insert apps;

        Test.startTest();
        ApplicationListController.PaginatedResult res =
            ApplicationListController.getApplications(5, 1, null);
        Test.stopTest();

        System.assertNotEquals(null, res);
        System.assert(res.records.size() <= 5, 'Should return at most 5 apps');
        System.assert(res.totalSize >= 12, 'Total size should count all apps');
    }
}
```

- **EnrollmentController** → Manages enrolling accepted students into courses.

Apex Code:

```
public with sharing class EnrollmentController {
    @AuraEnabled
    public static List<Enrollment_Course__c> enrollCourses(Id studentId, List<Id> courseIds) {

        if (studentId == null || courseIds == null || courseIds.isEmpty()) {
            return new List<Enrollment_Course__c>();
        }

        // Get the Enrolled Student record to find its Application__c
        Enrolled_Student__c enrolled = [
            SELECT Id, Application__c
            FROM Enrolled_Student__c
            WHERE Id = :studentId
            LIMIT 1
        ];

        List<Enrollment_Course__c> inserts = new List<Enrollment_Course__c>();
        for (Id cId : courseIds) {
            inserts.add(new Enrollment_Course__c(
                Student__c = studentId,
                Course__c = cId,
                Application__c = enrolled.Application__c
            ));
        }
        insert inserts;
        return inserts;
    }

    @AuraEnabled(cacheable=true)
    public static List<Course__c> getCourses() {
        return [SELECT Id, Name FROM Course__c ORDER BY Name LIMIT 500];
    }
}
```

Test class for EnrollmentController:

```apex
@isTest
public class EnrollmentControllerTest {

    @isTest
    static void testEnrollCourses_createsEnrollments() {
        // Step 1: Create Enrolled Student (Name auto-number, no value needed)
        Enrolled_Student__c student = new Enrolled_Student__c();
        insert student;

        // Step 2: Create Courses
        Course__c c1 = new Course__c(Name='Course 1');
        Course__c c2 = new Course__c(Name='Course 2');
        insert new List<Course__c>{c1, c2};

        // Step 3: Call the method under test
        Test.startTest();
        List<Enrollment_Course__c> enrollments = EnrollmentController.enrollCourses(
            student.Id,
            new List<Id>{c1.Id, c2.Id}
        );
        Test.stopTest();

        // Step 4: Verify enrollments were created
        System.assertEquals(2, enrollments.size(), 'Two enrollments should be created');
        for (Enrollment_Course__c ec : enrollments) {
            System.assertEquals(student.Id, ec.Student__c, 'Student ID should match');
        }
    }
}
```

Overall Code Coverage

Class	Percent	Lines
Overall	**94%**	
AdmissionDecisionPublisher	100%	17/17
ApplicationListController	77%	14/18
EnrollmentController	100%	15/15

Apex Triggers (before/after insert/update/delete)

- **Triggers** allow automation when records are created/updated.
- Example (not in my project at the moment, but possible extension):
  - A trigger on **Student_Application__c** can set Status__c = 'Under Review' when a new record is created.
  - Another trigger can enforce validation (e.g., program must be selected before status = Accepted).

Trigger Design Pattern

- To avoid logic directly in triggers, we use **Trigger Handler Classes**.

- Example: Instead of putting logic in a before update trigger for Student_Application__c, call methods in a separate handler class like ApplicationTriggerHandler.
- Benefits: Cleaner code, easier testing, reusability.

SOQL & SOSL

- **SOQL (Salesforce Object Query Language):**
  - Used in controllers to fetch records.
  - Example in ApplicationListController and in other classes also.

```
List<Student_Application__c> apps;
Integer total = 0;

if (String.isBlank(statusFilter)) {
    apps = [
        SELECT Id, Applicant_First_Name__c, Applicant_Last_Name__c,
               Email__c, Status__c, Submitted_Date__c
        FROM Student_Application__c
        ORDER BY Submitted_Date__c DESC
        LIMIT :pageSize OFFSET :offsetVal
    ];
    total = [SELECT COUNT() FROM Student_Application__c];
} else {
    apps = [
        SELECT Id, Applicant_First_Name__c, Applicant_Last_Name__c,
               Email__c, Status__c, Submitted_Date__c
        FROM Student_Application__c
        WHERE Status__c = :statusFilter
        ORDER BY Submitted_Date__c DESC
        LIMIT :pageSize OFFSET :offsetVal
    ];
    total = [SELECT COUNT() FROM Student_Application__c WHERE Status__c = :statusFilter];
}
```

- **SOSL (Salesforce Object Search Language):**
  - Useful for searching across multiple objects at once.
  - Example: Searching applications by applicant name or email.

Collections: List, Set, Map

- **List:** Ordered collection (used to store multiple application records).
- **Set:** Stores unique values (used to avoid duplicate student IDs).

- **Map:** Key-value pairs (e.g., mapping Course Id → Course Name).

In my project:

- **EnrollmentController** might use a List<Enrollment_Course__c> for bulk insert.
- **ApplicationListController** can use Map<Id, Student_Application__c> to optimize queries.

Control Statements

- Apex supports if-else, for, while, and switch.
- Example from tests:

```
if (studentId == null || courseIds == null || courseIds.isEmpty()) {
    return new List<Enrollment_Course__c>();
}

for (Id cId : courseIds) {
    inserts.add(new Enrollment_Course__c(
        Student__c = studentId,
        Course__c = cId,
        Application__c = enrolled.Application__c
    ));
}
```

Batch Apex

- Used for processing large data in chunks.
- Example extension for your project:
    - Batch job to update status of all applications pending for more than 30 days.

Queueable Apex

- Supports async jobs with chaining.
- Example extension:
    - Queueable job to send bulk acceptance/rejection emails after decisions

Scheduled Apex

- Run jobs at specific times.
- Example extension:
  - A scheduled job every Sunday to generate weekly admission statistics.

Future Methods

- Used for async processing of callouts or heavy tasks.
- Example extension:
  - @future method to send an external notification (to an external system) when an application is accepted.

Exception Handling

- Apex supports try-catch-finally.
- In AdmissionDecisionPublisher, if update fails due to missing program, an exception is thrown.

```
try {
    update app;
} catch(DmlException e) {
    System.debug('Error while updating: ' + e.getMessage());
}
```

Test Classes

- Required to validate logic and ensure coverage.
- In your repo:
  - **AdmissionDecisionPublisherTest** → Tests decision publishing logic.
  - **ApplicationListControllerTest** → Tests pagination and retrieval of applications.
  - **EnrollmentControllerTest** → Tests course enrollment logic.

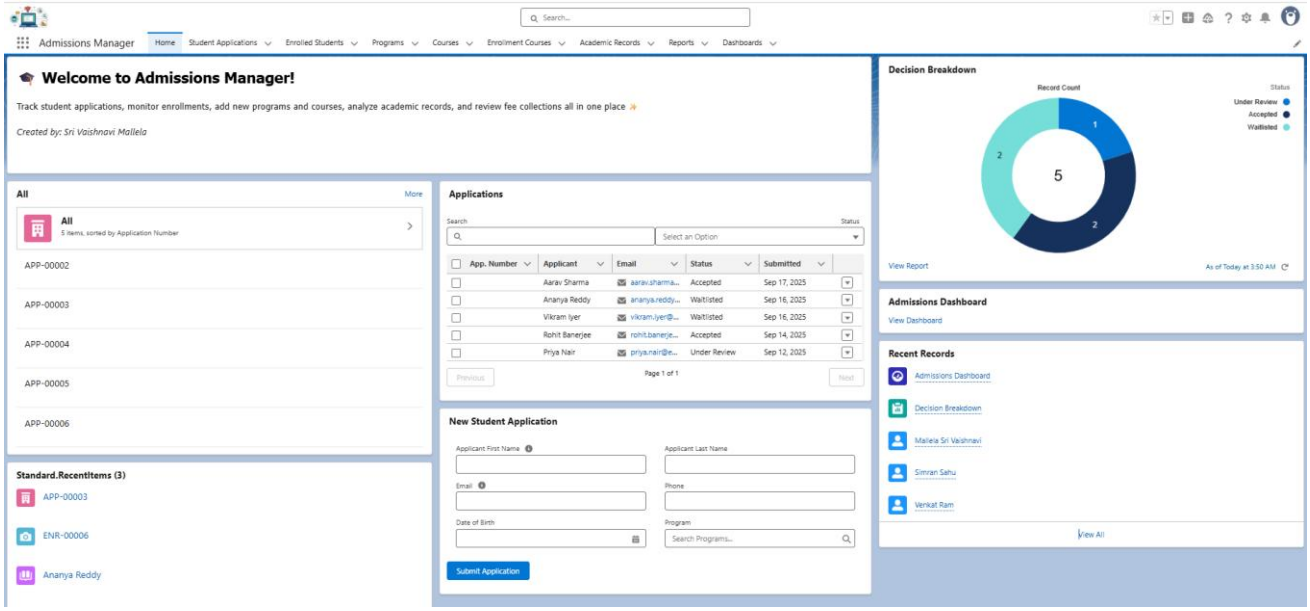
Salesforce requires **≥75% code coverage** for deployment to production.

Phase 6 (User Interface Development):

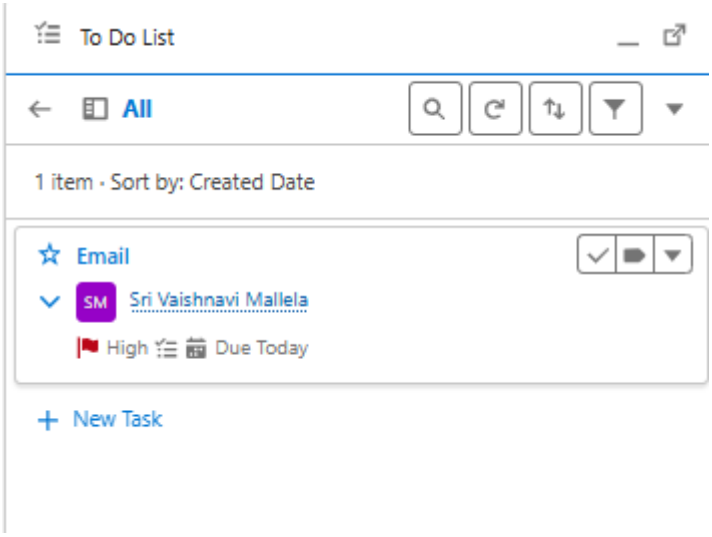
Lightning App Builder

- Built a **custom Admission Management App** with navigation for Applications, Students, Courses, and Enrollment, Academic Records etc.
- Added **Record Pages, Home Pages, Tabs, and Utility Bar** using Lightning App Builder.

Home Page Layout:



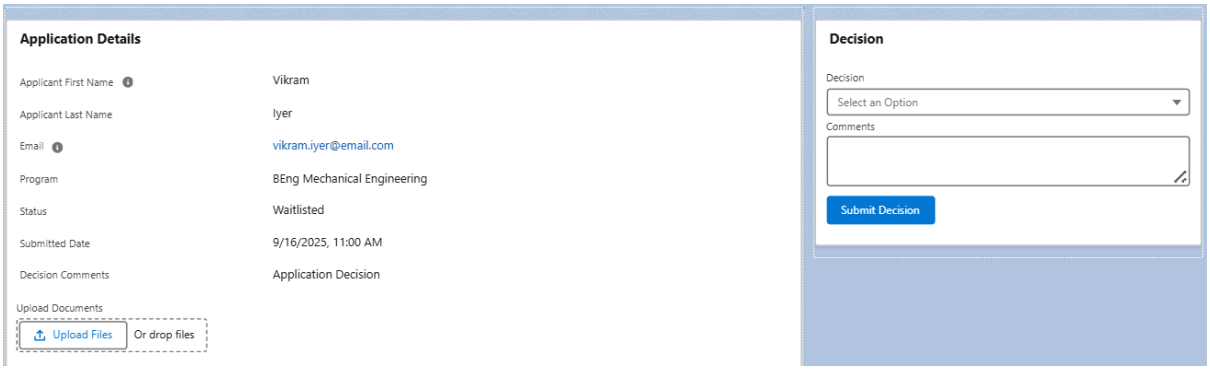
Utility Bar – To Do List:



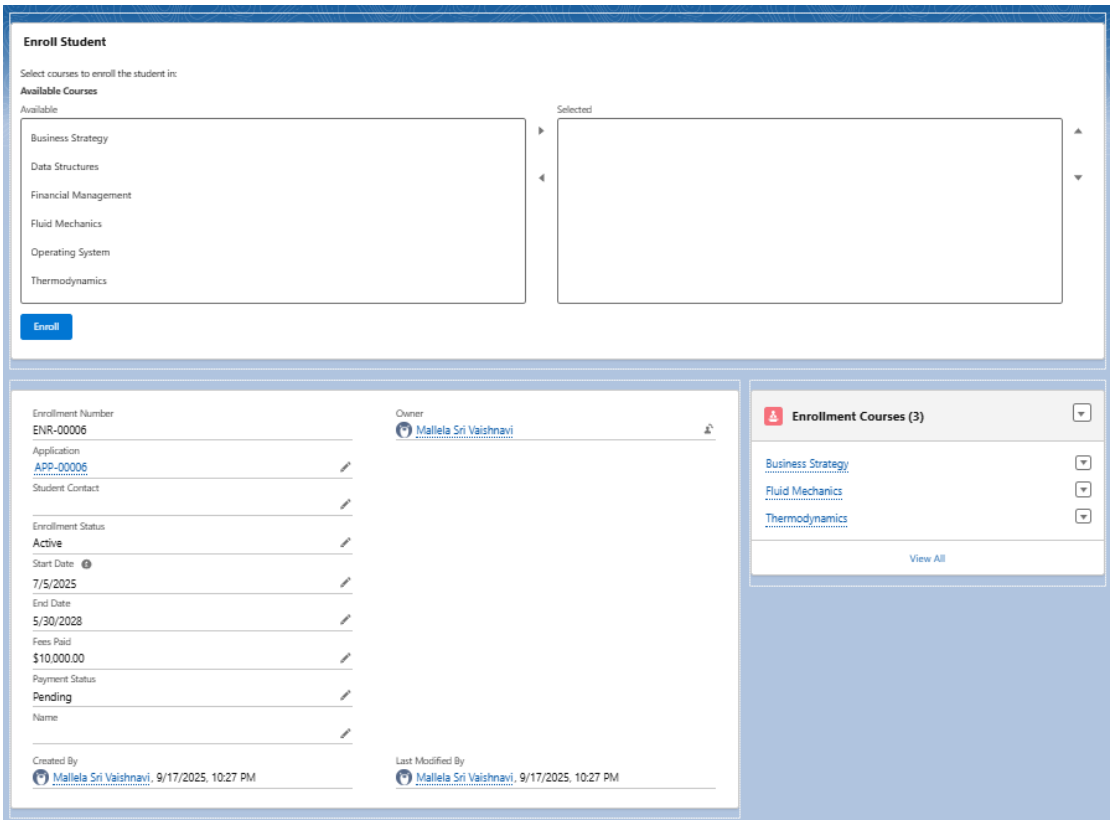
Record Pages

- Created a **custom record page** for **Student Application** object.
- Components: Application details, related list (enrollment, decisions), and highlights panel.

- **Lightning record page of Student_Application:**

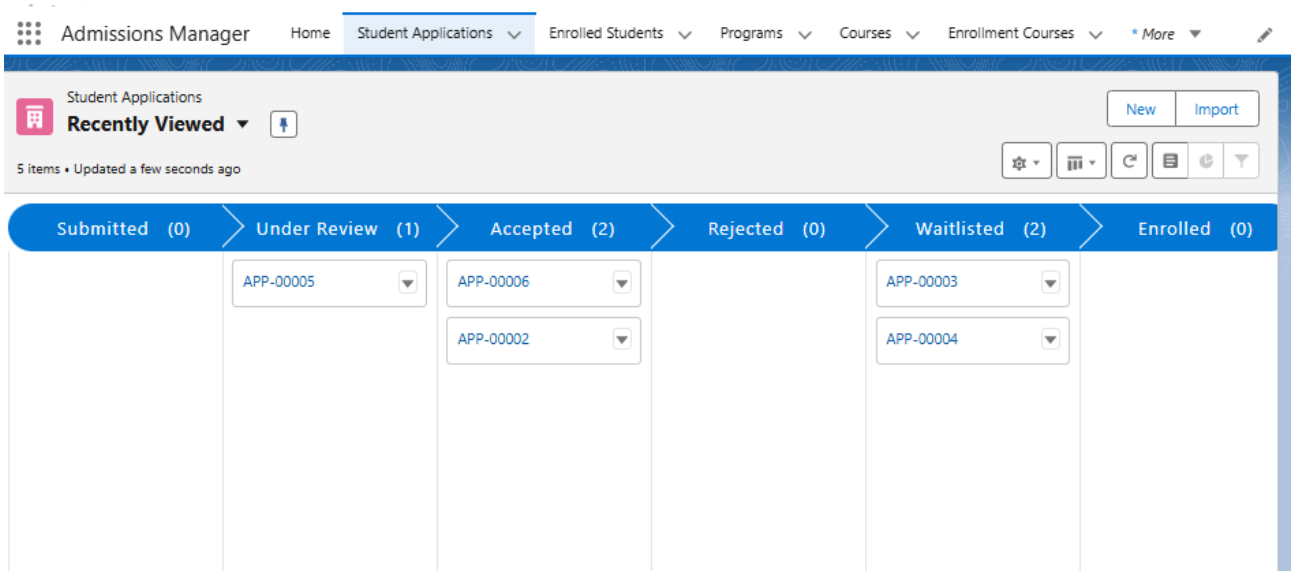


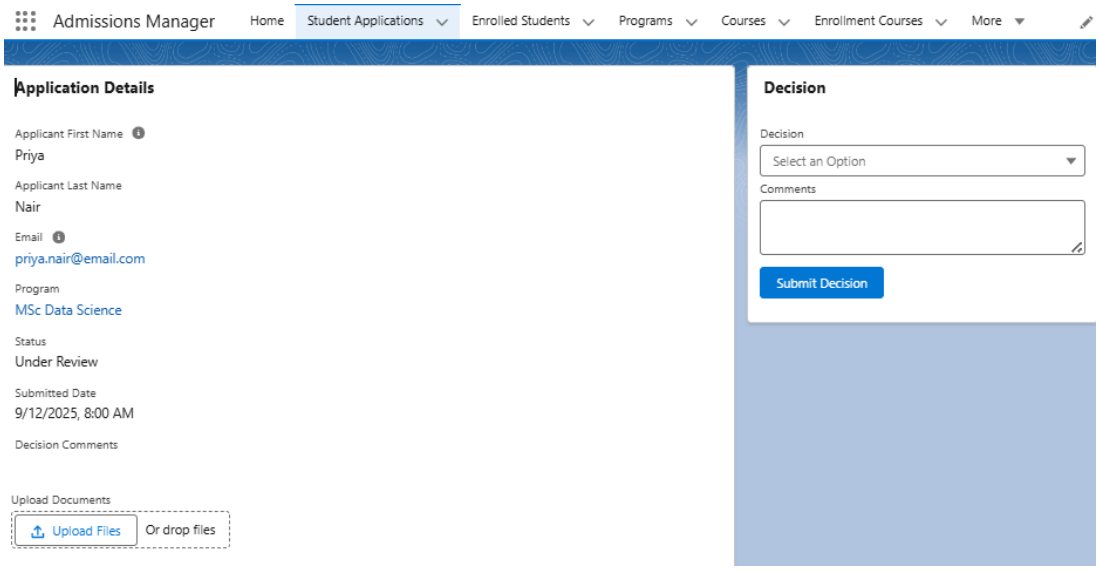
- **Lightning record page of Enrolled_Student:**



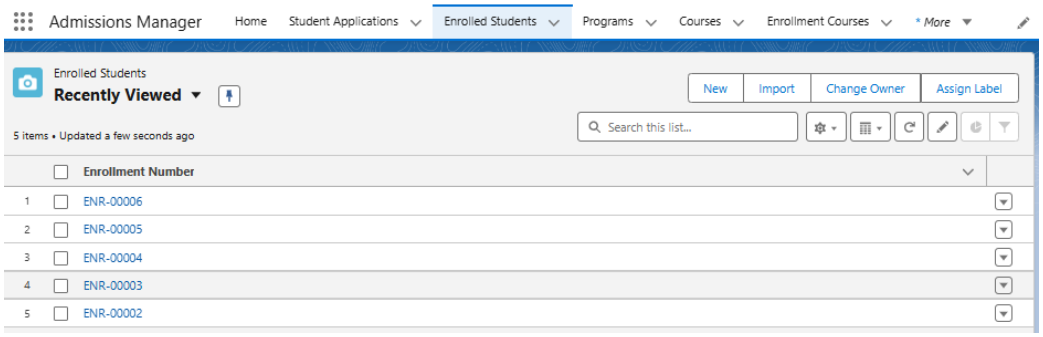
Tabs:

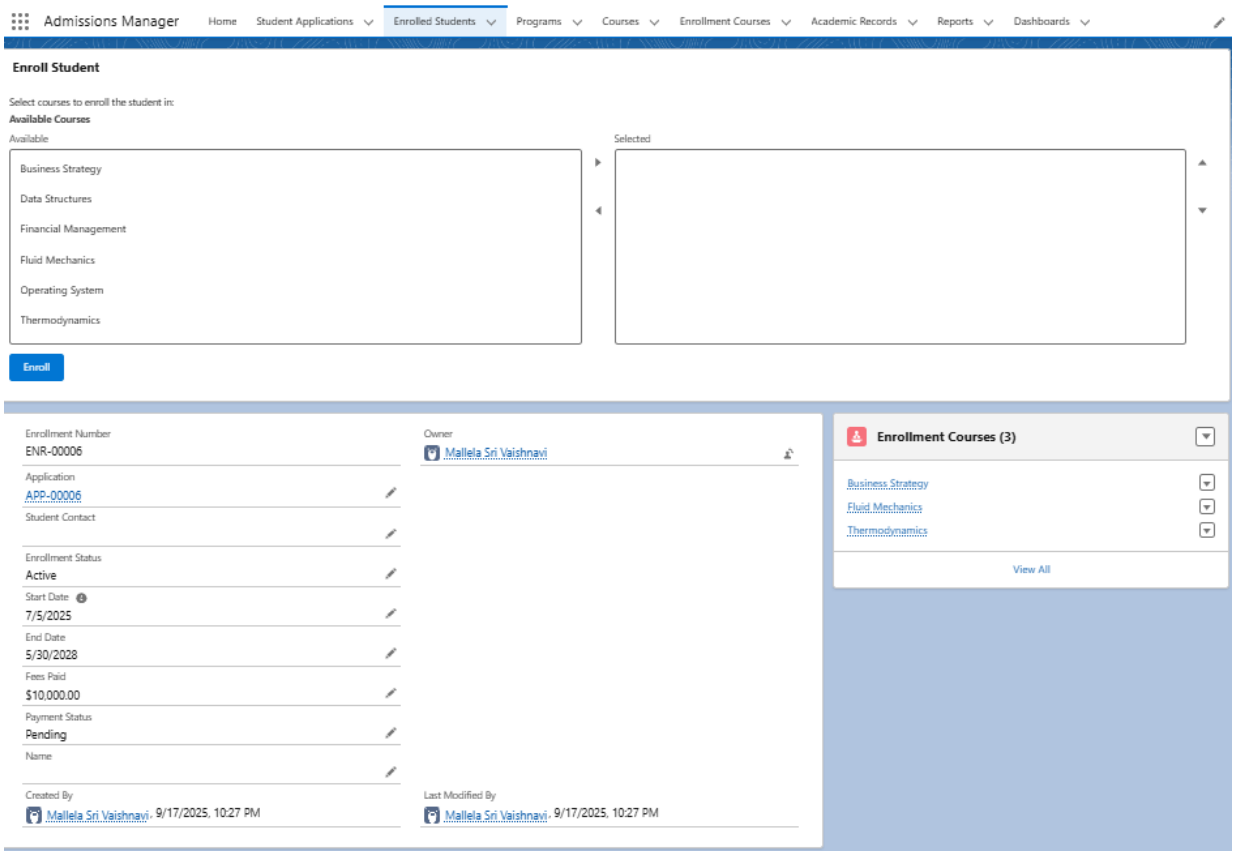
- **Student Applications tab**



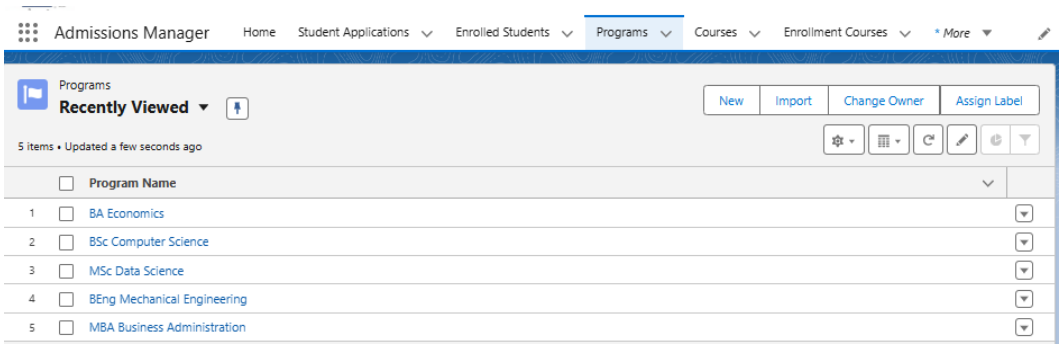


- **Enrolled Students tab**

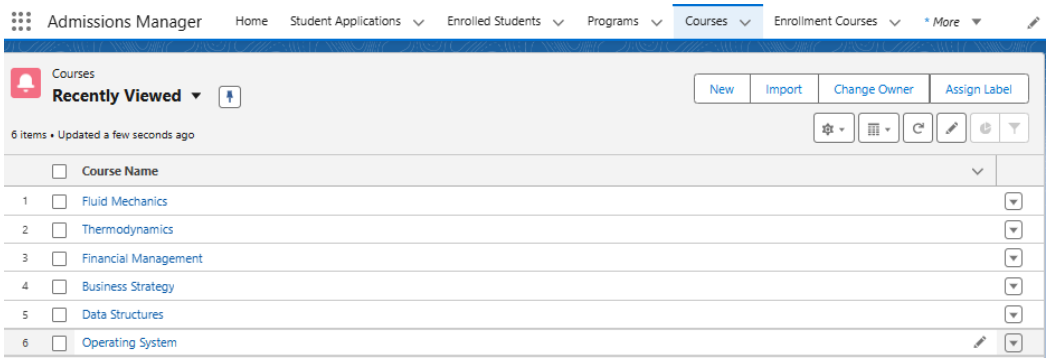




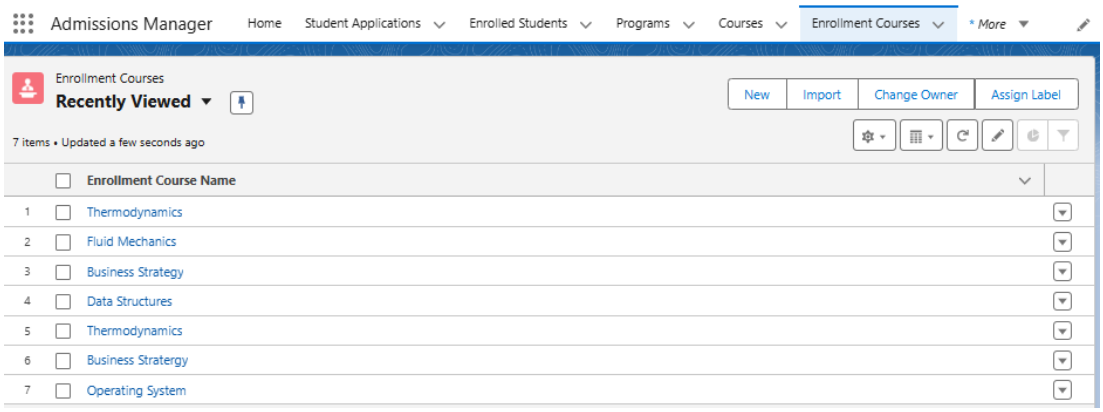
- **Programs tab**



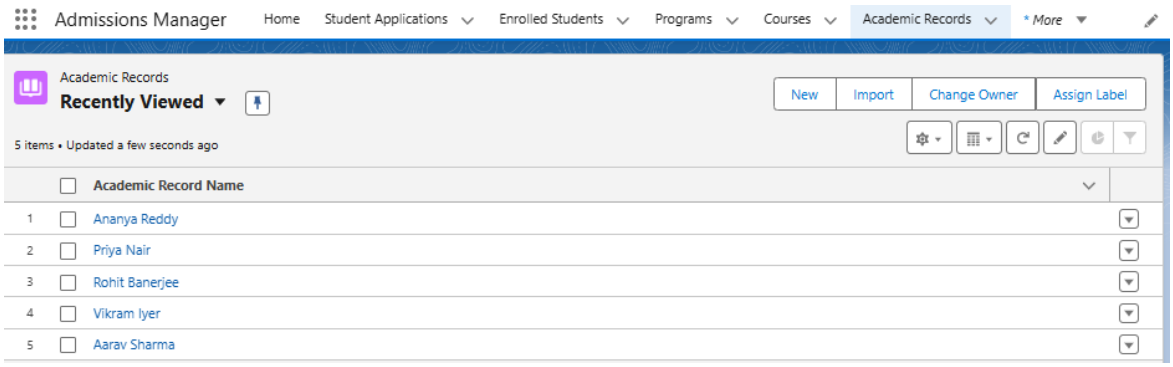
- **Courses tab**

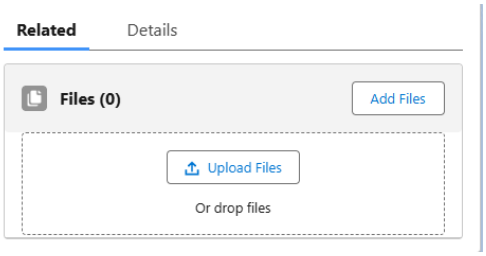


- **Enrollment Courses tab**



- **Academic Records tab**





Apex with LWC

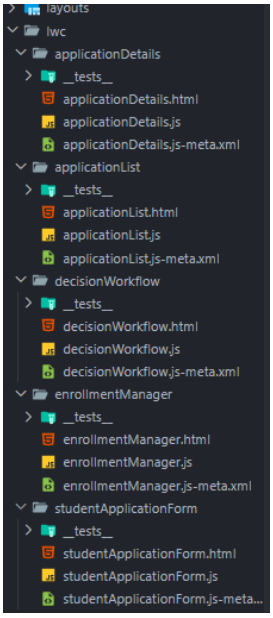
LWCs were integrated with **Apex controllers** to handle complex business logic and data operations that go beyond declarative tools.

Examples in Project:

- **studentApplicationForm** → Calls Apex to insert new applications into Salesforce.
- **decisionWorkflow** → Invokes Apex to publish platform events and send notifications after a decision is made.
- **enrollmentManager** → Uses SOQL in Apex to fetch related enrollments and update fee statuses.

Lightning Web Components (LWC)

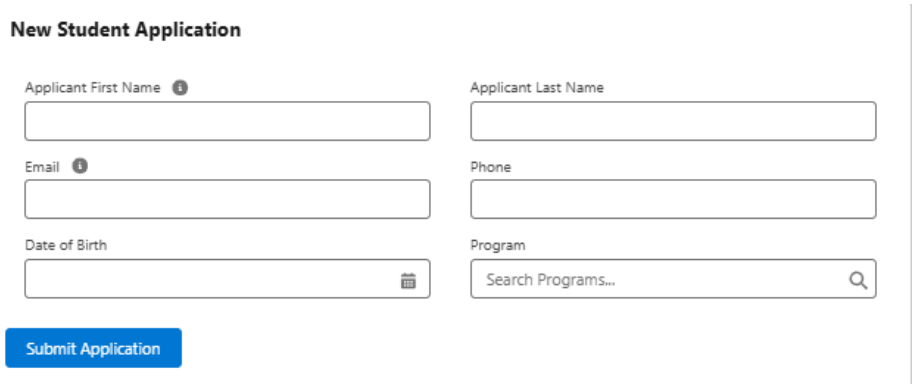
In this project, **Lightning Web Components (LWCs)** were built to provide a modern, responsive, and user-friendly interface for managing the student admission lifecycle. LWCs leverage standard web technologies (HTML, JS, CSS) integrated with Salesforce data.



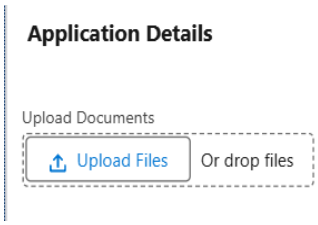
Action	Name ↑	Label	Type
Del	applicationDetails	applicationDetails	LWC
Del	applicationList	applicationList	LWC
Del	decisionWorkflow	decisionWorkflow	LWC
Del	enrollmentManager	enrollmentManager	LWC
Del	studentApplicationForm	studentApplicationForm	LWC

Key LWCs Developed:

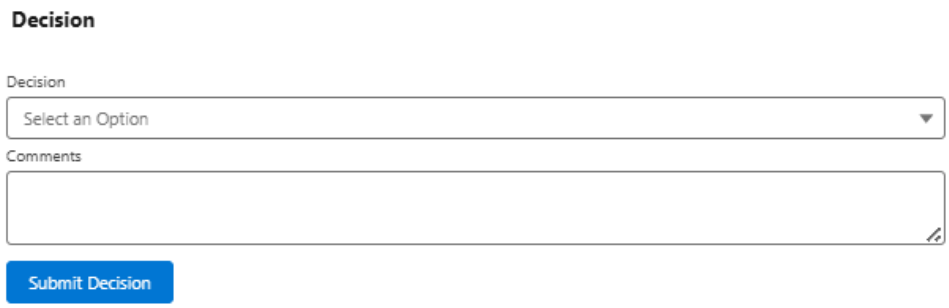
- **studentApplicationForm** → Allows applicants to submit personal, academic, and program details.



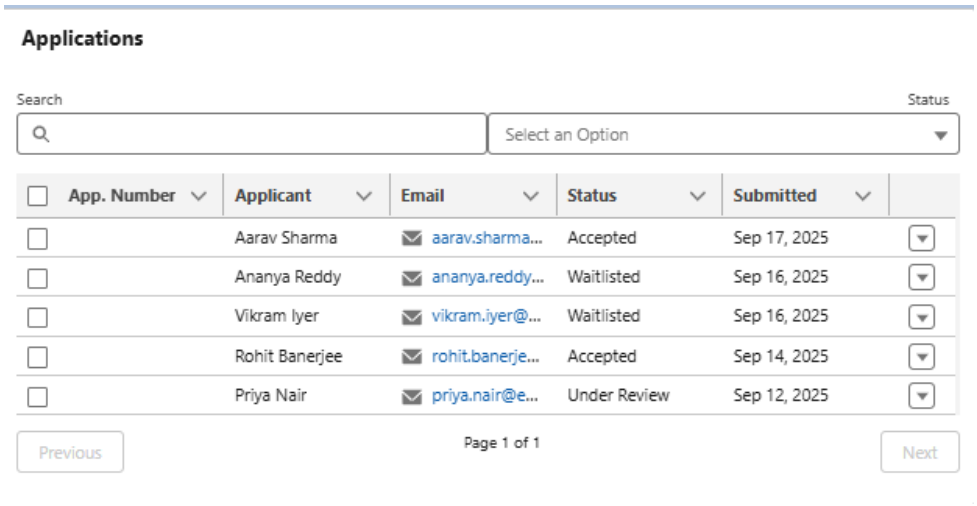
- **applicationDetails** → Shows detailed application records for review.



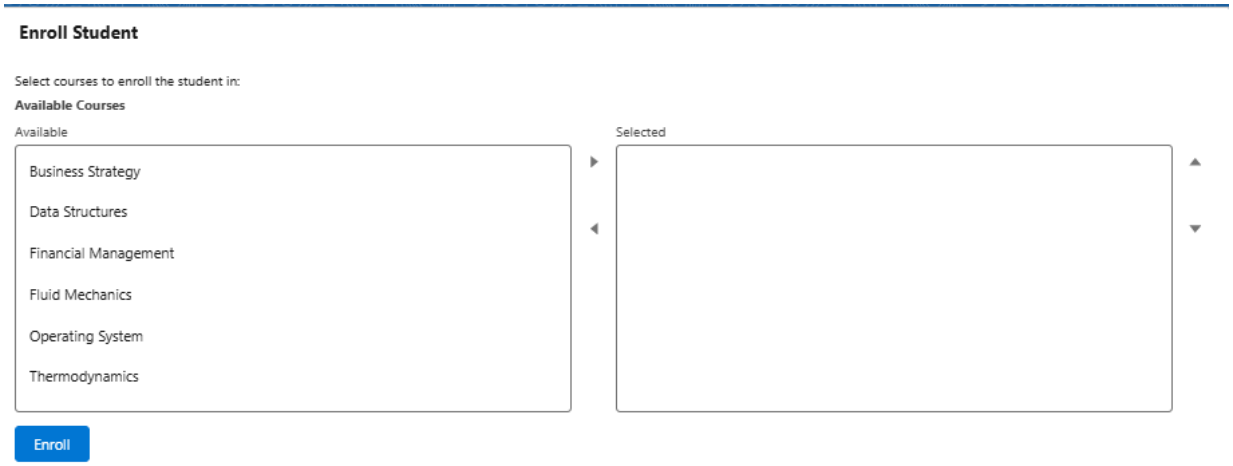
- **decisionWorkflow** → Enables admission officers to accept, reject, or waitlist applications.



- **applicationList** → Displays submitted applications with filtering and sorting for officers.



- **enrollmentManager** → Helps track enrollments, fee status, and course allocations.



Phase 7 (Integration & External Access) –

(Not integrated in Project)

Named Credentials

Named Credentials provide a secure and simplified way to call external APIs without storing authentication details in Apex code. They bundle the endpoint URL and authentication in one place, so developers only need to reference them in callouts.

- **Usage:** Used for integrating with third-party services such as payment gateway APIs to verify student fee payments.
- **Key Benefit:** Centralized management of authentication and endpoints, reducing hardcoding and increasing security.

External Services

External Services allow Salesforce to connect with APIs described by an OpenAPI/Swagger specification and automatically generate Apex actions or Flow actions. This enables both developers and admins to leverage APIs

- **Usage:** Used to integrate with external services such as document verification APIs (for verifying student identity, certificates, or test scores).
- **Key Benefit:** Exposes external system functionality inside Salesforce, which can then be directly used in Flows to automate admission processes.

Web Services (REST/SOAP)

Salesforce supports consuming and exposing web services using REST and SOAP protocols. REST is commonly used for lightweight and modern integrations, while SOAP is still supported for legacy systems.

- **Usage:** REST services can be used for sending student admission data to external analytics platforms. SOAP services can be used for exchanging admission decisions with traditional ERP systems.

- **Key Benefit:** Enables Salesforce to work seamlessly with both modern and legacy applications.

Callouts

Apex callouts allow Salesforce to send HTTP requests to external services and process responses. They can be synchronous (real-time) or asynchronous.

- **Usage:** Used for confirming online payment transactions, checking scholarship eligibility, or fetching course details from external academic systems.
- **Key Benefit:** Enables Salesforce to stay connected with external data sources, ensuring accurate information during admission workflows.

Platform Events

Platform Events enable an event-driven architecture by allowing apps to publish and subscribe to custom events.

- **Usage:** When an admission decision (Accept, Reject, Waitlist) is made, a Platform Event can notify the student portal and trigger external notification services like SMS or email systems.
- **Key Benefit:** Provides real-time, asynchronous communication across Salesforce and external systems.

Change Data Capture (CDC)

Change Data Capture publishes real-time events whenever records in Salesforce are created, updated, deleted, or undeleted. External systems can subscribe to these events.

- **Usage:** When a student updates their contact details or when an application status changes.
- **Key Benefit:** Ensures synchronization of data across Salesforce and external systems without manual intervention.

Salesforce Connect

Salesforce Connect allows Salesforce to access external data sources in real-time without importing or storing the data in Salesforce.

- **Usage:** Used to access a student's past academic history from the university's external database without duplicating records in Salesforce.
- **Key Benefit:** Reduces storage costs, avoids redundancy, and provides up-to-date external data on demand.

API Limits

Salesforce enforces API request limits based on the edition and user licenses to maintain performance and system stability.

- **Usage:** The Student Admission CRM must carefully monitor API usage when performing frequent callouts for payment verification, document checks, and integration with third-party apps.
- **Key Benefit:** Helps manage integrations efficiently while staying within Salesforce's governor limits.

OAuth & Authentication

OAuth 2.0 is a secure, token-based authentication mechanism used for connecting Salesforce with external apps. It allows apps to access Salesforce data without exposing user credentials.

- **Usage:** Used when connecting Salesforce with student self-service portals, learning management systems (LMS)
- **Key Benefit:** Provides secure, scalable, and industry-standard authentication for integrated apps.

Remote Site Settings

Remote Site Settings whitelist external endpoints so Salesforce can send callouts to them. This was the traditional approach before Named Credentials.

- **Usage:** Used for integrations with legacy services such as SMS gateways or APIs that do not require advanced authentication.
- **Key Benefit:** Protects Salesforce from sending data to unauthorized or untrusted endpoints.

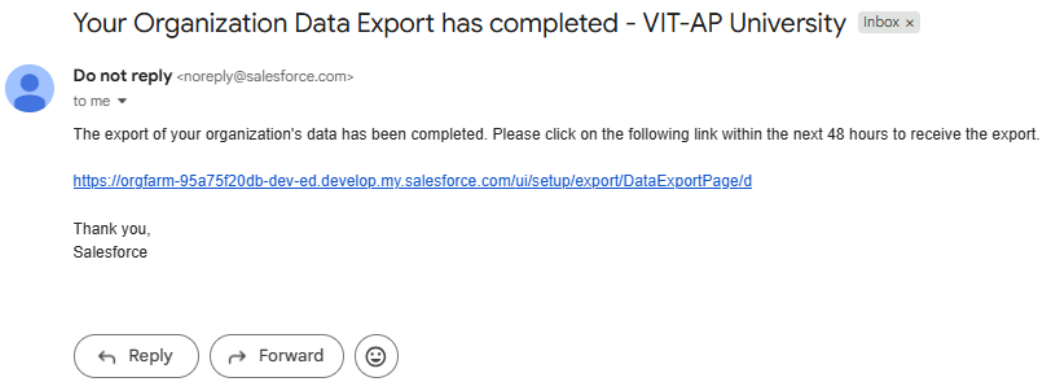
Phase 8 (Data Management & Deployment)

Data Import Wizard

- Browser-based tool to import up to 50,000 records.
- Supports standard/custom objects with field mapping and duplicate prevention.
- Used by Admissions Manager to upload new applications, academic records, or program/course details received offline.

Data Loader

- Desktop client to handle large volumes of records (millions).
- Supports insert, export, update, upsert, delete.
- Used for migrating historical applications, bulk updating fee statuses, or exporting enrollment data for audits.



Duplicate Rules

- Works with Matching Rules to prevent duplicate records.
- Can block duplicates or allow with alerts.
- Ensures no duplicate student account,contact,
- records are created when the same applicant reapplies with the same email or ID.

All Duplicate Rules

What Are Duplicate Rules?

View: All Duplicate Rules

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R

New Rule

Rule Name	Description	Object	Matching Rule	Active	Last Modified By
Standard Account Duplicate Rule	Identify accounts that duplicate other accounts.	Account	Standard Account Matching Rule	✓	OEPIC
Standard Contact Duplicate Rule	Identify contacts that duplicate other contacts and leads.	Contact	Standard Contact Matching Rule Standard Lead Matching Rule	✓	OEPIC
Standard Lead Duplicate Rule	Identify leads that duplicate other leads and contacts.	Lead	Standard Lead Matching Rule Standard Contact Matching Rule	✓	OEPIC

Data Export & Backup

- Provides manual and scheduled export of Salesforce data.
- Exports records into CSV files bundled in .zip.
- Weekly/monthly backups ensure secure storage of student applications, enrollments, and course allocations for compliance.

Monthly Export Service

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

Next scheduled export:
None

Export Now | Schedule Export

Scheduled By	Mallela Sri Vaishnavi
Schedule Date	9/21/2025
Export File Encoding	ISO-8859-1 (General US & Western European, ISO-LATIN-1)

Action	File Name	File Size
download	WE_00DgL00000BGZ9xUAH_1.ZIP	3.1M

VS Code & SFDX

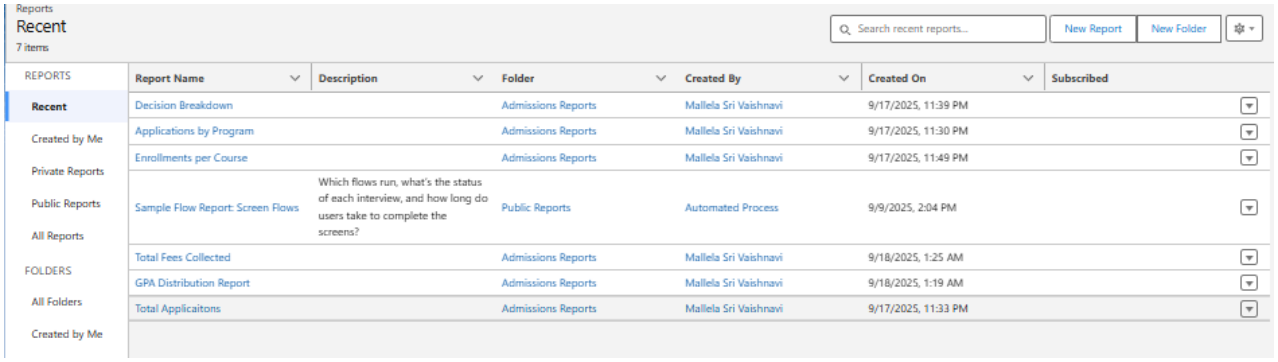
- Modern development environment for Salesforce.
- SFDX enables scratch orgs, push/pull metadata, testing, and deployments.
- Used to build and deploy LWCs like studentApplicationForm, applicationList, and enrollmentManager with version control integration.

```
──────────── Deploying Metadata ────────────

Deploying v64.0 metadata to srivaishnavimallela24213@agentforce.com using the v64.0 SOAP API.

✓ Preparing 397ms
◯ Waiting for the org to respond - Skipped
✓ Deploying Metadata 640ms
  ▸ Components: 1/1 (100%)
◯ Running Tests - Skipped
◯ Updating Source Tracking - Skipped
✓ Done 0ms

Status: Succeeded
Deploy ID: 0AfgL00000AXWBTSA5
Target Org: srivaishnavimallela24213@agentforce.com
Elapsed Time: 1.06s

Deployed Source
```

State	Name	Type	Open file in editor (ctrl + click)
Changed	applicationDetails	LightningComponentBundle	force-app\main\default\lwc\applicationDetails\applicationDetails.html
Changed	applicationDetails	LightningComponentBundle	force-app\main\default\lwc\applicationDetails\applicationDetails.js
Changed	applicationDetails	LightningComponentBundle	force-app\main\default\lwc\applicationDetails\applicationDetails.js-meta.xml

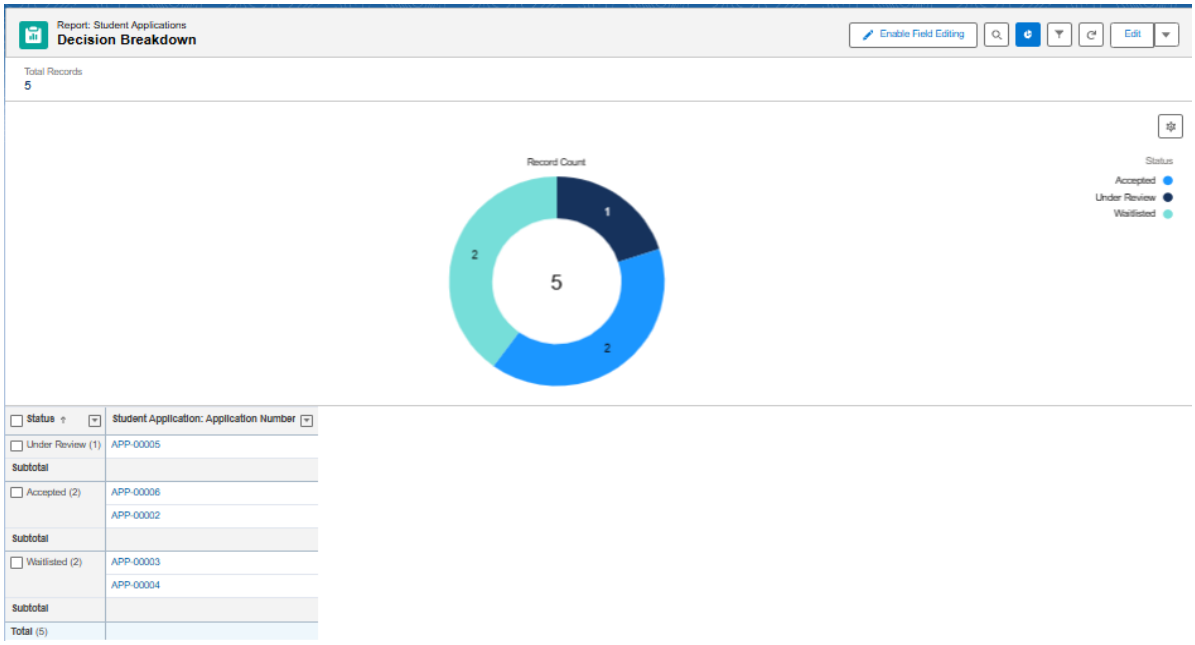
Phase 9 (Reporting, Dashboards & Security Review)

Reports Created:



Decision Breakdown report:

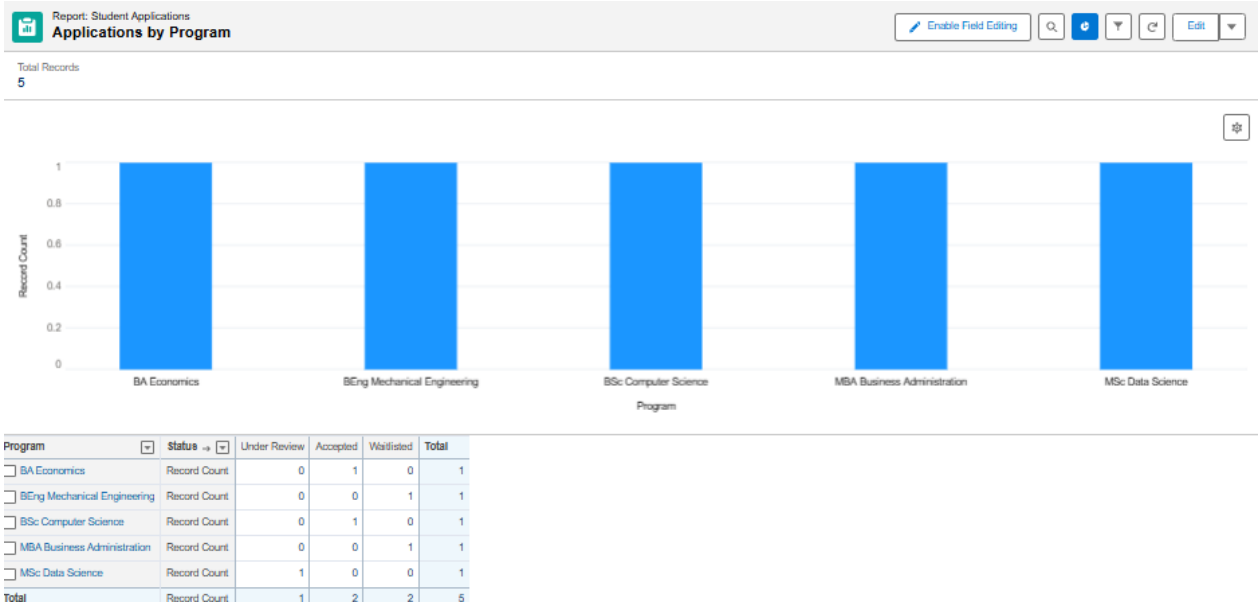
- Displays the count of applications by decision type (Accepted, Rejected, Waitlisted) as a pie chart and table.
- Helps Admission Officers monitor the distribution of decisions.
- Useful for tracking yearly admission patterns.



Applications by Program report:

- Shows number of applications submitted for each program/course as a bar graph and table for better visualization.

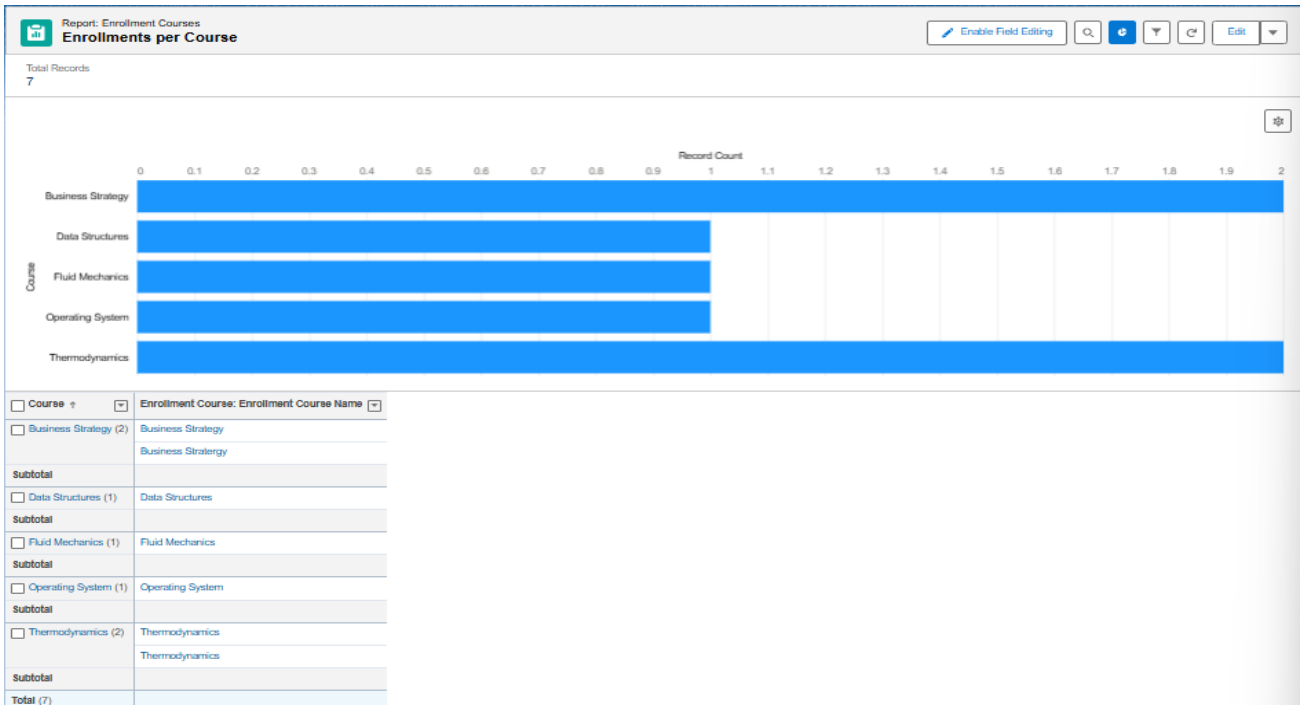
- Helps management identify the most popular programs.
- Supports resource planning (e.g., more faculty for high-demand programs).



Program	Status	Under Review	Accepted	Waitlisted	Total
BA Economics	Record Count	0	1	0	1
BEng Mechanical Engineering	Record Count	0	0	1	1
BSc Computer Science	Record Count	0	1	0	1
MBA Business Administration	Record Count	0	0	1	1
MSc Data Science	Record Count	1	0	0	1
Total	Record Count	1	2	2	5

Enrollments per course report:

- Displays number of students enrolled in each course as bar graph, table.
- Useful for academic planning and resource allocation (faculty, classrooms).
- Helps detect under-enrolled courses.



Course	Enrollment Course: Enrollment Course Name
Business Strategy (2)	Business Strategy
	Business Stratergy
Subtotal	
Data Structures (1)	Data Structures
Subtotal	
Fluid Mechanics (1)	Fluid Mechanics
Subtotal	
Operating System (1)	Operating System
Subtotal	
Thermodynamics (2)	Thermodynamics
	Thermodynamics
Subtotal	
Total (7)	

Total fees collected report:

- Tracks total fees received from enrolled students and individual payments
- Allows Finance/Admin team to ensure payment completion.
- Can be filtered by Program or Enrollment Term for deeper analysis.

Report: Enrolled Students

Total Fees Collected

Total Records	Total Fees Paid
5	$265,000.00

	Enrolled Student: Enrollment Number	Fees Paid
1	ENR-00002	$25,000.00
2	ENR-00003	$60,000.00
3	ENR-00004	$20,000.00
4	ENR-00005	$150,000.00
5	ENR-00006	$10,000.00
6		$265,000.00

GPA Distribution report:

- Analyzes applicants' academic performance by grouping based on GPA ranges into buckets and also calculates total GPA.
- Useful for evaluating applicant quality across programs.
- Helps in comparing GPA trends between accepted and rejected students.

Report: Academic Records

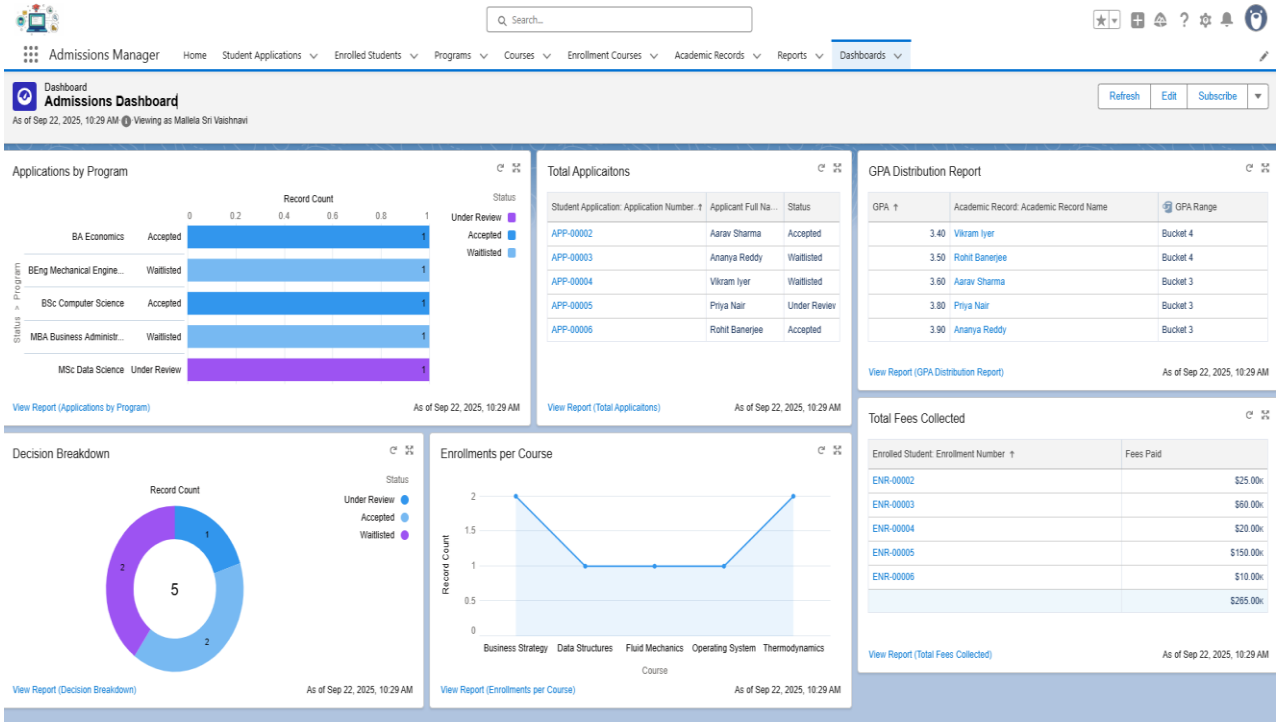
GPA Distribution Report

Total Records	Total GPA
5	18.20

	GPA	Academic Record: Academic Record Name	GPA Range
1	3.40	Vikram Iyer	Bucket 4
2	3.50	Rohit Banerjee	Bucket 4
3	3.90	Ananya Reddy	Bucket 3
4	3.60	Aarav Sharma	Bucket 3
5	3.80	Priya Nair	Bucket 3
6	18.20		

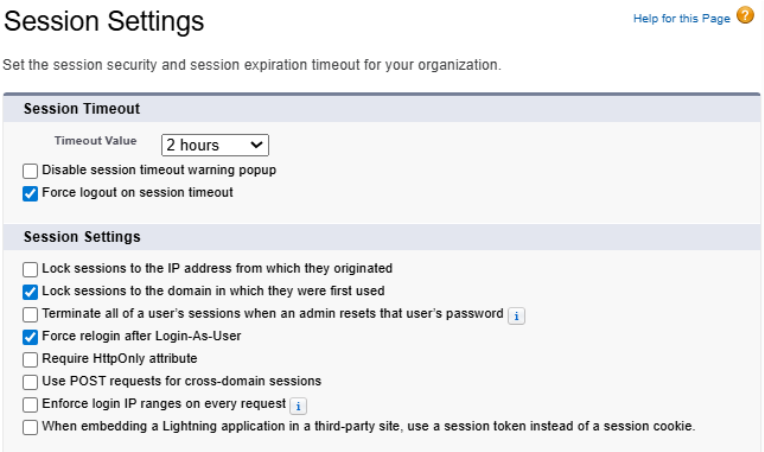
Dashboards

- Dashboards provide a visual representation that can be refreshed from time to time of admission reports such as Decision Breakdown, Applications by Program, GPA Distribution, and Enrollments per Course.
- In the Student Admission project, dashboards allow Admission Officers and Management to quickly monitor admission trends, financial status and course enrolment.



Session Settings

- In the Student Admission project, session settings can enforce logout after inactivity to prevent unauthorized access to application data.



Field Level Security (FLS)

- Ensures sensitive fields (such as GPA, Fee Payment Details, and Admission Decision Comments) are only visible to authorized users.
- This maintains data confidentiality and compliance with institutional policies.

Example: Admission Officers can view GPA but cannot edit it, while Finance staff can see payment details but not academic records.

Silver Partner User	☐	☐
Solution Manager	☐	☐
Standard Platform User	☐	☐
Standard User	☐	☐
System Administrator	☑	☐
Work.com Only User	☐	☐

Audit Trail

- Tracks all administrative changes (like modifications to fields, validation rules, or user permissions) it ensures transparency by recording changes.

View Setup Audit Trail

Help for this Page

The last 20 entries for your organization are listed below. You can download your organization's setup audit trail for the last six months (Excel .csv file).

View Setup Audit Trail

Date	User	Source Namespace Prefix	Action	Section	Delegate User
9/22/2025, 10:04:21 AM PDT	srivaishnavimallela24213@agentforce.com		Changed applicationDetails Lightning Web Component	Lightning Components	
9/21/2025, 11:07:05 PM PDT	srivaishnavimallela24213@agentforce.com		Requested an export	Data Export	
9/21/2025, 10:51:09 AM PDT	srivaishnavimallela24213@agentforce.com		Changed Lightning Page: Enrolled Student	Lightning Pages	
9/21/2025, 10:41:16 AM PDT	srivaishnavimallela24213@agentforce.com		Changed applicationDetails Lightning Web Component	Lightning Components	
9/21/2025, 10:40:57 AM PDT	srivaishnavimallela24213@agentforce.com		Changed enrollmentManager Lightning Web Component	Lightning Components	
9/21/2025, 10:39:12 AM PDT	srivaishnavimallela24213@agentforce.com		Changed decisionWorkflow Lightning Web Component	Lightning Components	
9/21/2025, 10:15:17 AM PDT	srivaishnavimallela24213@agentforce.com		Organization setup action: localDevelopmentPreviewPrefOnOff has changed.		
9/21/2025, 10:15:16 AM PDT	srivaishnavimallela24213@agentforce.com		Organization setup action: localDevelopmentPreviewPrefOffOn has changed.		
9/21/2025, 4:16:40 AM PDT	srivaishnavimallela24213@agentforce.com		Changed Permission Set Dashboard Subscriptions: Session Activation Required has changed from disabled to enabled	Manage Users	
9/21/2025, 4:14:21 AM PDT	srivaishnavimallela24213@agentforce.com		Permission set Dashboard Subscriptions with expiration: assigned to user Mallela Sri Vaishnavi (UserID: [005gL000007dR17]), expiring on Sun Sep 28 18:29:59 GMT 2025		
9/21/2025, 4:12:09 AM PDT	srivaishnavimallela24213@agentforce.com		Permission set Dashboard Subscriptions: unassigned from user Mallela Sri Vaishnavi (UserID: [005gL000007dR17])	Manage Users	

Phase 10 (Final Presentation & Demo Day)