Name: Grandhi Deekshita

Institution: Gayatri Vidya Parishad College of Engineering

Email Id: 21131A0482@gvcpe.ac.in

Project Title: A CRM Application to Manage the Services offered by an Institution

1. Project Overview

This project focuses on developing and implementing EduConsultPro, a comprehensive Customer Relationship Management (CRM) solution built on the Salesforce platform. It addresses the challenges faced by EduConsultPro Institute in managing its student lifecycle, from initial inquiry to enrollment, ongoing support, and even post-graduation follow-up. The goal is to create a centralized, efficient, and scalable system to enhance operational efficiency, improve communication, and provide data-driven insights. This will lead to an improved experience for students and staff, fostering student success and supporting the long-term growth of the Institute.

2. Objectives

Business Goals:

- Enhance Student Onboarding: Streamline the application process for a smooth experience.
- Optimize Appointment Scheduling: User-friendly system minimizing scheduling conflicts.
- Improve Communication & Collaboration: Seamless information sharing between stakeholders.
- Boost Operational Efficiency: Automate tasks, reduce manual entry, and minimize errors.
- Data-Driven Decision Making: Gather data on trends and performance for informed decisions.
- Increase Student Enrollment and Retention: Positive student experience fostering engagement.

- Scalability and Future Growth: Adaptable solution for evolving institutional needs.

Specific Outcomes:

- Functional Salesforce CRM application named "EduConsultPro."
- Custom objects (Course, Consultant, Student, Appointment, Registration, etc.) with defined fields.
- Multi-stage screen flows for student application, appointment scheduling, and case management.
- Apex triggers for welcome case creation, appointment confirmations, and data validation.
- User-friendly Lightning App interface with custom dashboards for easy navigation.
- Intuitive appointment approval process with notifications.
- Comprehensive testing (unit and user acceptance) and deployment.
- Ongoing maintenance and support.
- Detailed documentation (data models, UI, logic, testing, training).

3. Salesforce Key Features and Concepts Utilized

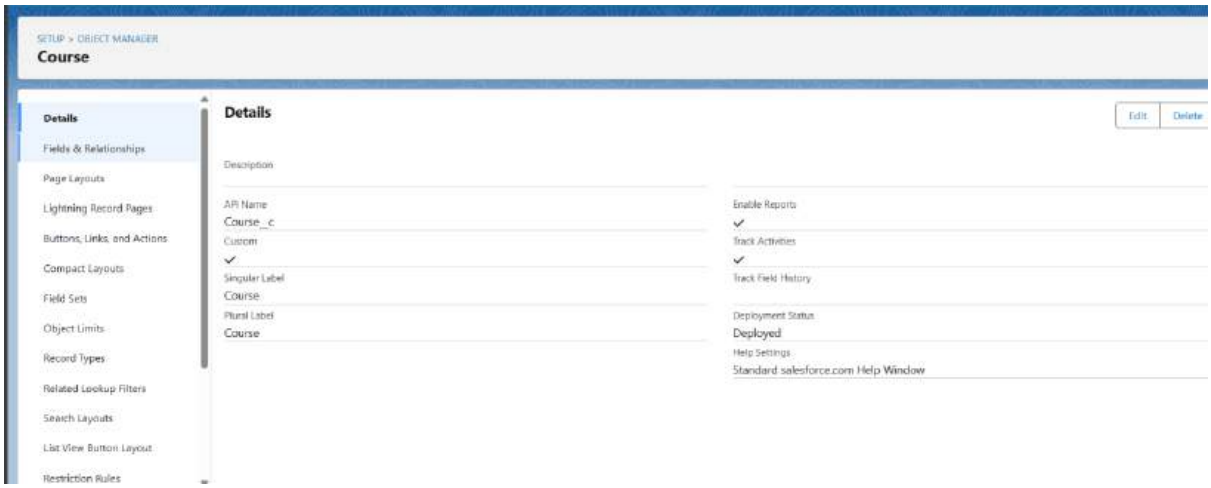
- **Custom Objects & Fields:** Representing the institution's data model.
- **Relationships (Lookup & Master-Detail):** Linking related data between objects.
- **Lightning App Builder:** Creating a tailored and intuitive user interface.
- **Screen Flows:** Automating complex processes with user input screens and conditional logic.
- **Apex Triggers & Classes:** Enforcing business logic, data validation, and bulkification.
- **Approval Processes:** Managing authorization workflows with notifications.
- **Standard Objects (e.g., Case):** Handling student inquiries and support.

- **Email Templates & Alerts:** Standardized communication for various events.

4. Detailed Steps to Solution Design

Create Objects From Spreadsheet

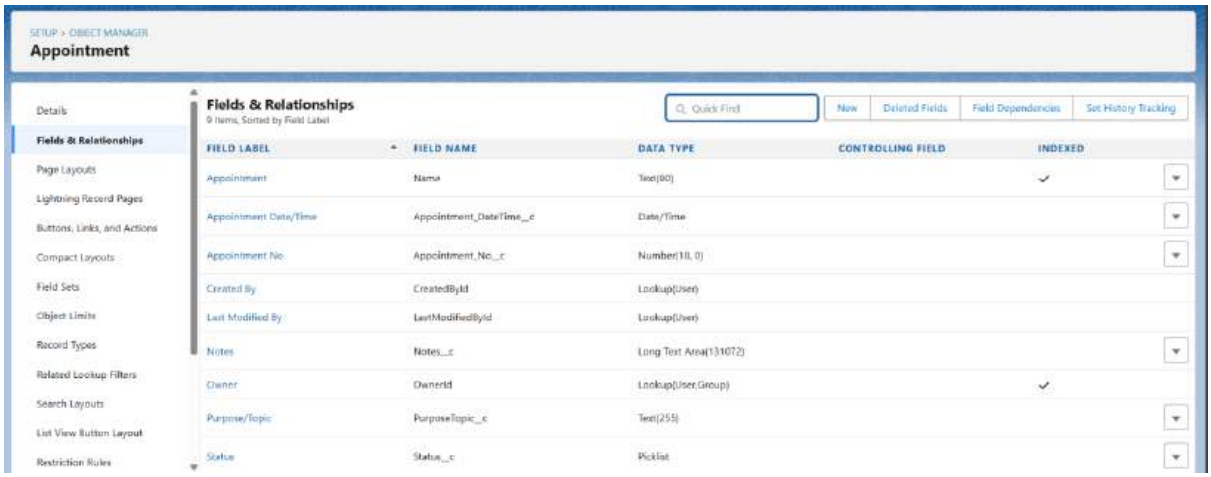
- **Create Course Object:** A custom object named "Course" was created using the data provided in the 'Course' spreadsheet. This involved mapping the spreadsheet columns to the Salesforce fields in the Course object.



- **Create Remaining Objects:** Similar to the Course object, custom objects were created for Consultant, Student, Appointment, and Registration using the provided spreadsheets. Field mapping ensured data integrity during the import process.

- **Create Relationship Among the Objects:** Lookup relationships were established between the objects to connect related data:
    - Appointment to Student (Student lookup on Appointment object)

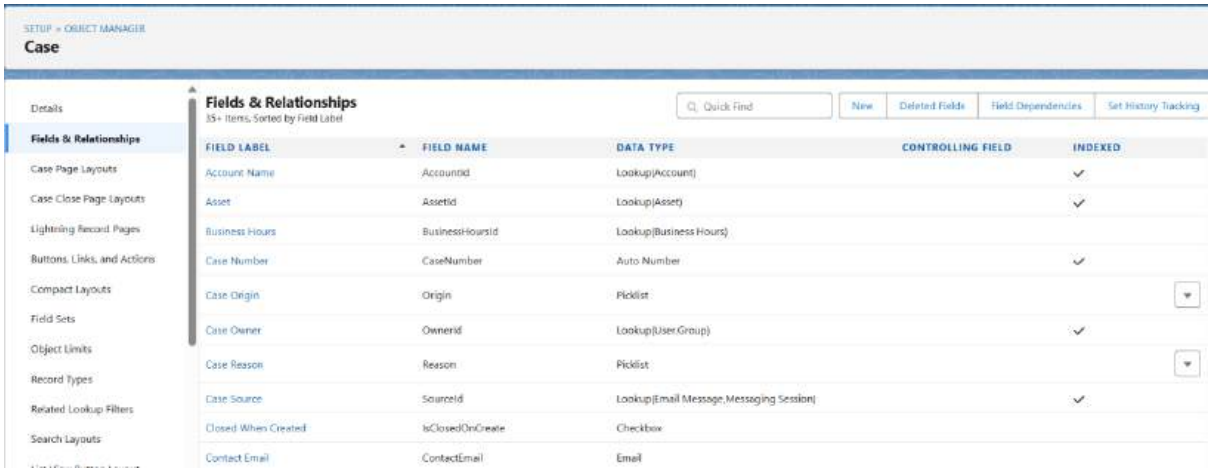
- Appointment to Consultant (Consultant lookup on Appointment object)
- A custom object, "Registration," was created to store student and course details. Lookup relationships were added:
  - Registration to Student
  - Registration to Course
- A lookup relationship was also established between Student and Case objects.



Configure The Case Object

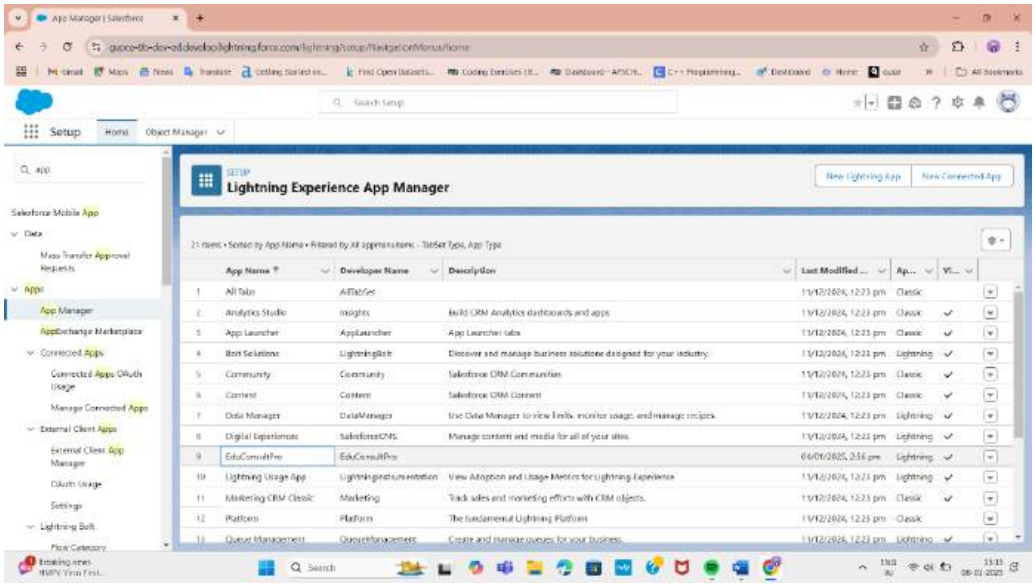
The standard Case object was configured to include custom picklist values:

- **Type Field:** 'Immigration' and 'Visa Application' values added.
- **Status Field:** 'Open' and 'In-Progress' values added. (Consider adding 'Closed' or similar for completeness).



Create A Lightning App

A Lightning App named “EduConsultPro” was created. This app included tabs for Home, Students, Courses, Consultants, Appointments, Registrations, and Cases, providing users with a centralized location to access all functionalities. The app was made available to the System Administrator profile.

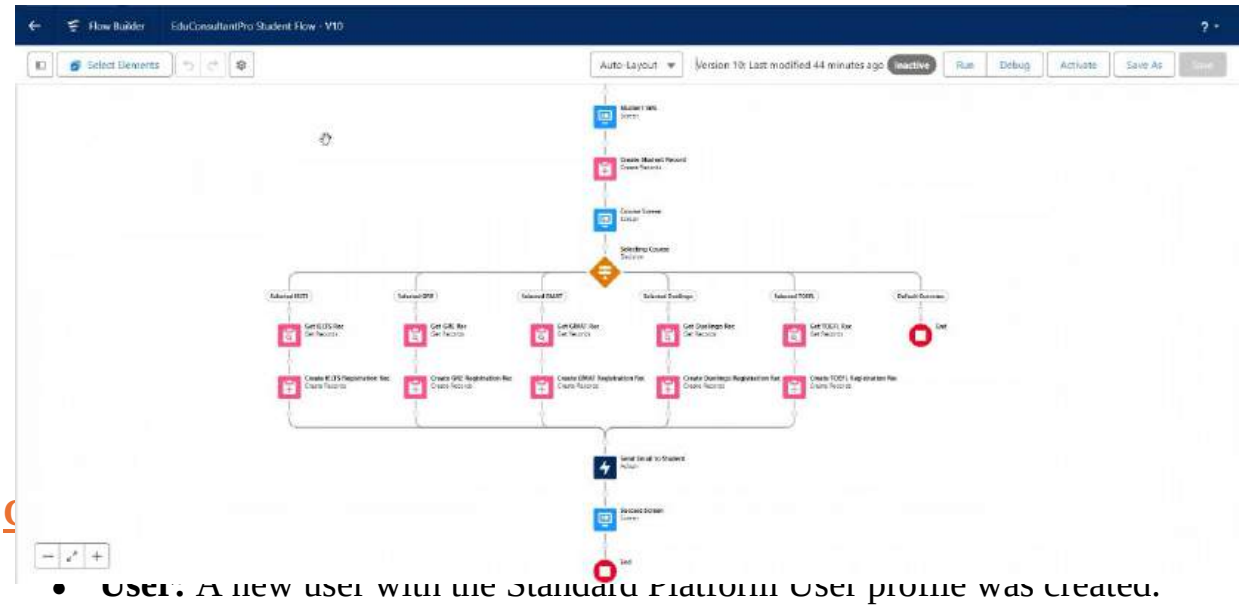


Create A ScreenFlow For Student Admission Application Process

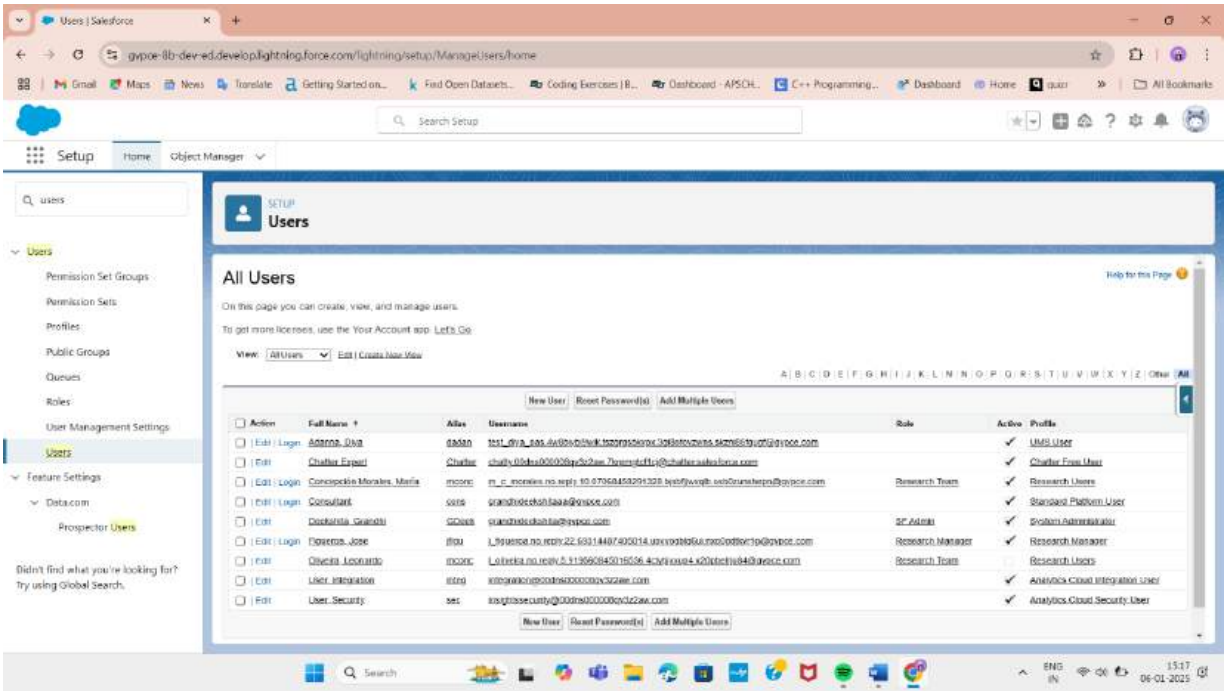
- **Add Screen Element (Student Info):** A screen element was added to collect student information. This element displayed fields from the Student object. A record variable resource StudentRecordRes was created.
- **Create Student Record Using Create Element:** A create element, “Create Student Record,” was added to create a new Student record using the data collected from the “Student Info” screen.
- **Add Screen Element (Course Screen):** This screen allows students to select a course (IELTS, GRE, GMAT, Duolingo, TOEFL) from a picklist. Choice variables were created for each course option.
- **Add Decision Element (Selecting Course):** A decision element checks the selected course from the "Course Screen" and routes the flow base on

the selection. Outcomes were defined for each course option (e.g., "Selected IELTS").

- **Add GET Record Element:** For each course outcome, a Get Record element retrieves the corresponding Course record based on the selected course name.
- **Create Registration Record Using Create Records Element:** A create element creates a Registration record, linking the newly created Student record and the retrieved Course record. This was done for each course outcome path.
- **Create Email Text Template Variables For Email Body And Subject:** Two text template resources were created:
  - StuRegistrationEmailTextTempBody for the email body content.
  - StuRegistrationEmailTextTempSub for the email subject.
- **Add An Action Element (Send Email to Student):** An email alert action was added to send a registration confirmation email to the student. It utilizes the email templates created in the previous step.
- **Add Screen Element (Success Screen):** A final screen displays a success message to the student.



- **User:** A new user with the Standard Platform User profile was created.
- **Configure The User Settings:** The new user was assigned a manager in their user settings, essential for the approval process.



Create An Approval Process For Property Object (Should be Appointment Object)

- **Create An Email Template:** Email templates were created for Submission, Approval, and Rejection stages of the Appointment approval process.

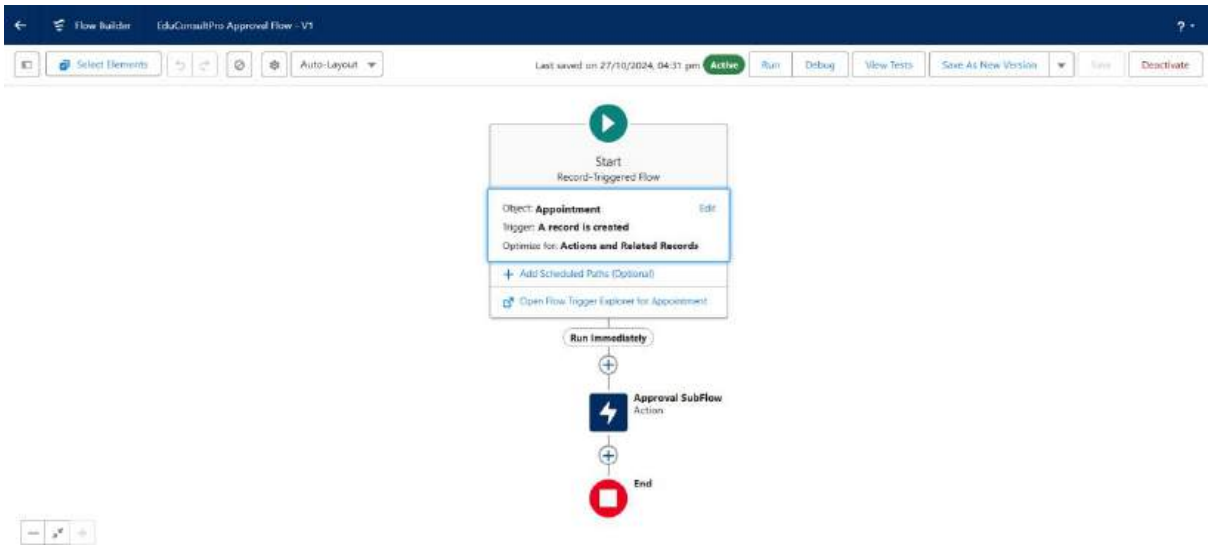
	Email Template Name	Description	Email Templates from Salesforce	Last Modified By ID	Last Modified Date
1	Approval Template		Public Email Templates	Grandhi Deekshita	05/01/2025, 10:27 pm
2	Enhancement Request Approved	Enhancement Request Approval email template	Public Email Templates	Grandhi Deekshita	11/12/2024, 12:23 pm
3	New Enhancement Request	New Enhancement Request email template	Public Email Templates	Grandhi Deekshita	11/12/2024, 12:23 pm
4	Rejection Template		Public Email Templates	Grandhi Deekshita	05/01/2025, 10:27 pm
5	Submission Template		Public Email Templates	Grandhi Deekshita	05/01/2025, 10:27 pm

- **Create An Approval Process:** An approval process for the Appointment object was configured. The manager of the submitting user was designated as the approver. Record editability properties were set to allow administrators or the assigned approver to edit records during the approval process. Initial Submission, Final Approval, and Final Rejection actions were configured to update the Appointment record's status and send email notifications using the respective templates.

Action	Approval Process Name	Des
Edit | Activate | Del	Appointment Approval	

Create A Record Triggered Flow

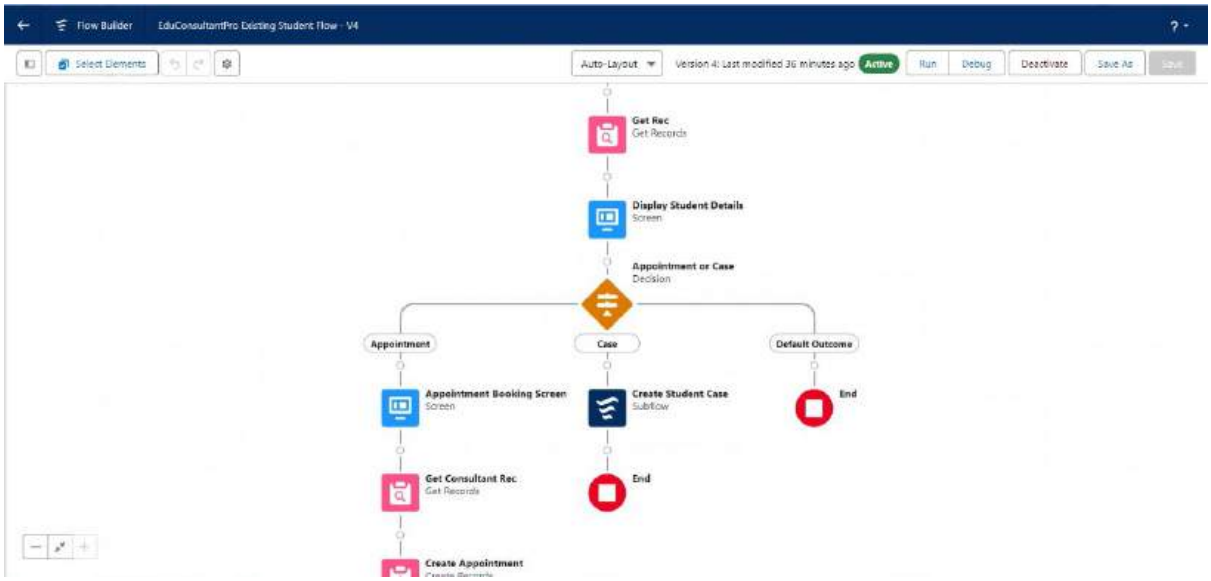
- **Configure The Start Element:** A record-triggered flow was initiated, triggering when an Appointment record is created.
- **Add An Action Element:** An action element was added to submit the newly created Appointment record for approval using the Submit for Approval action.



Create A ScreenFlow For Existing Student To Book An Appointment

- **Add Screen Element (Get Student Info):** Collects existing student's name and email.
- **Add GET Record Element (Get Rec):** Retrieves the Student record matching the entered name and email.
- **Add Decision Element (Appointment or Case):** Determines whether the student wants to book an appointment or create a case (this branch of the flow wasn't fully described in the original instructions, so requires further definition). Presumably, a screen would be needed before this decision element to offer those choices.

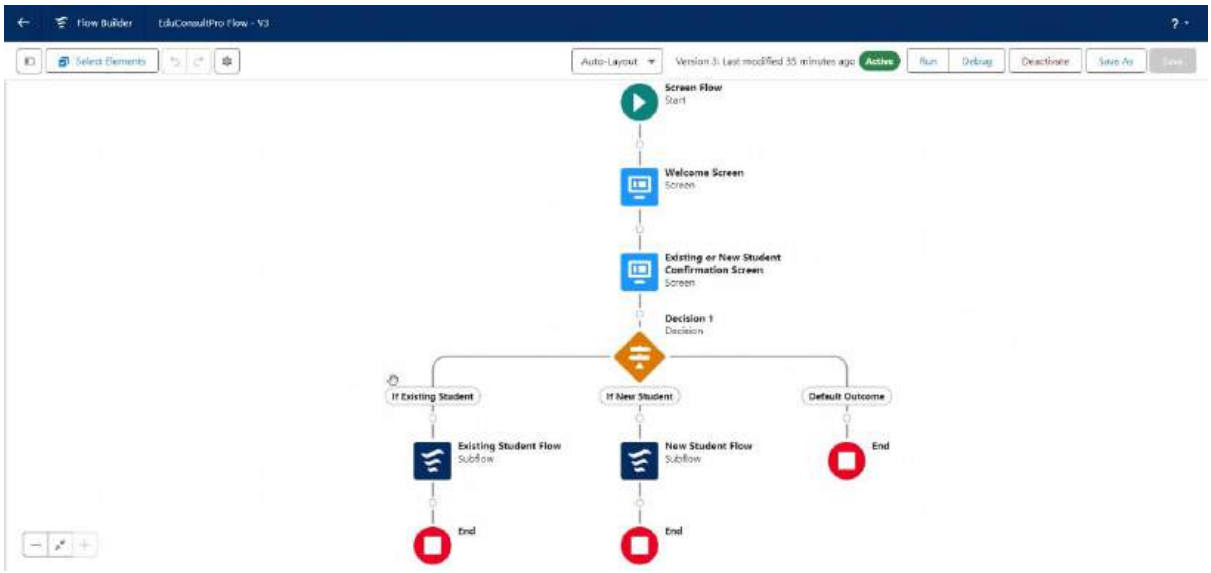
- **Add Screen Element (Appointment Booking Screen):** Displays fields from the Appointment object for the student to fill in. AppointmentRecordRes resource is used.
- **Add GET Record Element (Get Consultant Rec):** Retrieves the Consultant record based on the selected consultant name.
- **Create Appointment Record Using Create Records Element (Create Appointment):** Creates an Appointment record with details provided.
- **Add Screen Element (Confirmation Screen):** Displays confirmation message with appointment details.
- **Add An SubFlow Element (Create Student Case):** (This was under the "Case" path of the Decision element and requires further definition, assuming it involves creating a Case record if the student chose that option).



Create A ScreenFlow To Combine All The Flows At One Place

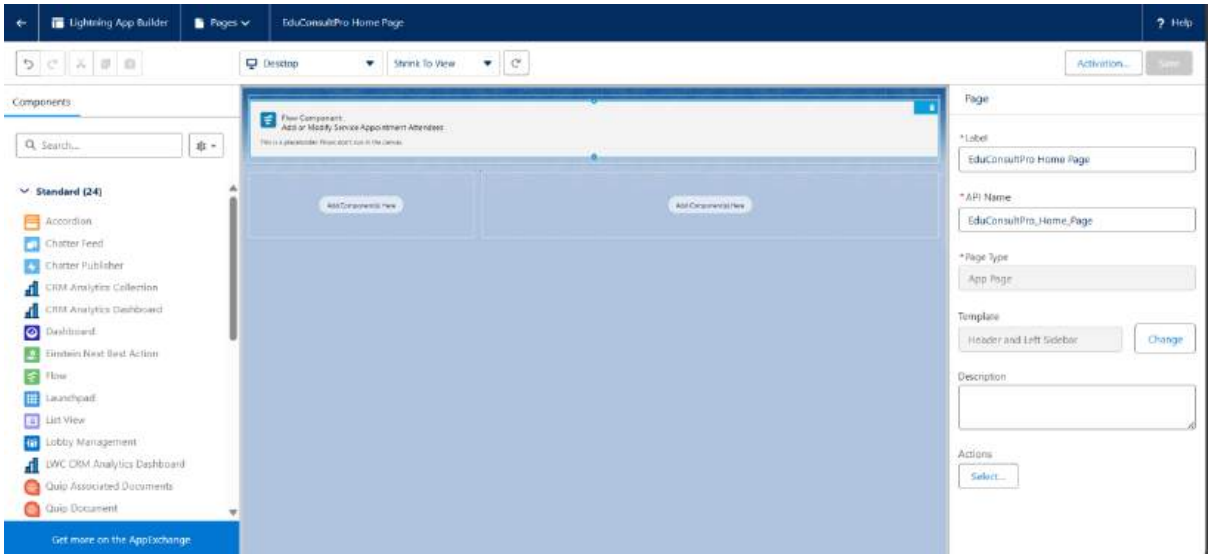
- **Add Screen Element (Welcome Screen):** Displays a welcome message.
- **Add Screen Element (Existing or New Student Confirmation Screen):** Asks the user if they are an existing student.
- **Add Decision Element (Decision 1):** Routes the flow based on user input (Existing/New Student).

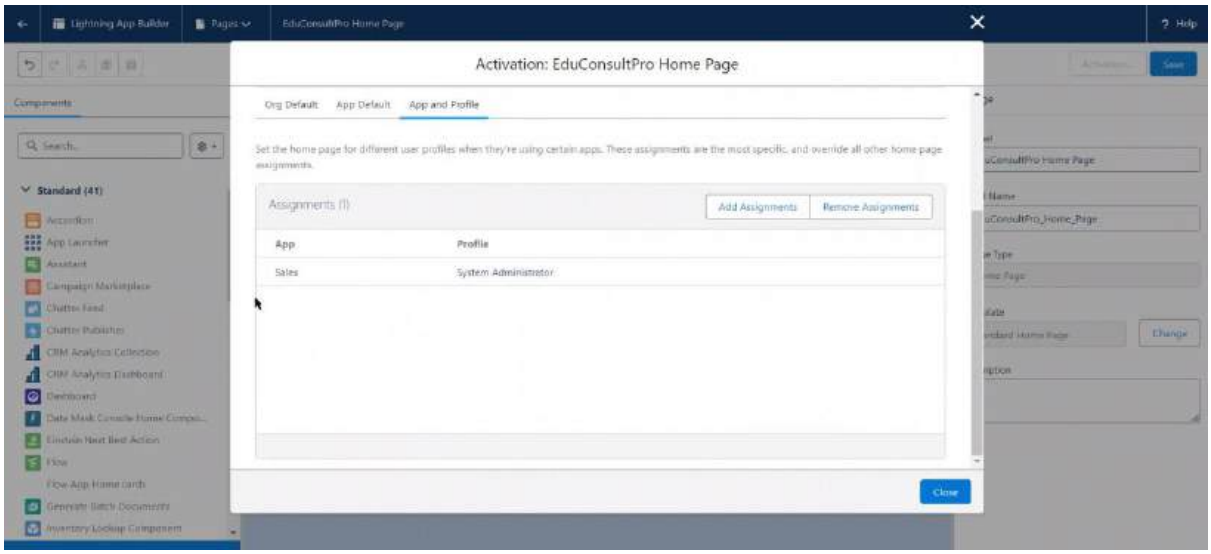
- **Add An SubFlow Element (Existing Student Flow / New Student Flow):** Calls the respective Subflow based on the decision element outcome (Student Admission flow or Existing Student Appointment booking flow).



Create A Lightning App Page

- **Create a Lightning App Page:** A Lightning app page "EduConsultPro Home Page" was created and configured as the default Home page for the "Sales" app and assigned to the System Administrator profile. This page displayed the Combined Flow.





5. Testing and Validation

- **Unit Testing:** Apex unit tests with high code coverage for all triggers and classes.
- **Integration Testing:** Verification of interactions between objects and standard functionality.
- **User Acceptance Testing (UAT):** Real-world testing by users for usability and effectiveness.
- **Performance Testing:** Ensuring system responsiveness and stability under various loads.

```apex
// Trigger 1: Automatically create a welcome case when a new student is created
trigger StudentTrigger on Student__c (after insert) {
    List<Case> casesToInsert = new List<Case>();

    for(Student__c student : Trigger.new) {
        // Create a welcome case for each new student
        Case welcomeCase = new Case(
            Subject = 'Welcome to EduConsultPro',
            Description = 'Welcome to EduConsultPro! This case has been automatically created to track your initial onboarding.',
            Status = 'Open',
            Origin = 'New Student Registration',
            Priority = 'Normal',
            ContactEmail = student.Email__c,
            Student__c = student.Id  // Assuming there's a lookup field on Case to Student__c
        );
        casesToInsert.add(welcomeCase);
    }

    if(!casesToInsert.isEmpty()) {
        try {
            insert casesToInsert;
        } catch(DMLException e) {
            // Add error handling
            for(Student__c student : Trigger.new) {
                student.addError('Unable to create welcome case: ' + e.getMessage());
            }
        }
```

Trigger1.apxt * Trigger2.apxt *

Code Coverage: None API Version: 62

```
// Trigger 2: Validate appointment scheduling and send notification
trigger AppointmentTrigger on Appointment__c (before insert, before update, after insert) {
    if(Trigger.isBefore) {
        // Validate appointment scheduling
        for(Appointment__c appt : Trigger.new) {
            // Check if appointment is during business hours (9 AM to 5 PM)
            DateTime apptDateTime = DateTime.newInstance(
                appt.Appointment_Date__c,
                appt.Appointment_Time__c
            );

            Integer hourOfDay = apptDateTime.hour();

            if(hourOfDay < 9 || hourOfDay >= 17) {
                appt.addError('Appointments can only be scheduled between 9 AM and 5 PM');
            }

            // Check if consultant is available at the requested time
            if(Trigger.isInsert || (Trigger.isUpdate &&
                appt.Consultant__c != Trigger.oldMap.get(appt.Id).Consultant__c ||
                appt.Appointment_Date__c != Trigger.oldMap.get(appt.Id).Appointment_Date__c ||
                appt.Appointment_Time__c != Trigger.oldMap.get(appt.Id).Appointment_Time__c)) {

                // Query for existing appointments
                List<Appointment__c> existingAppointments = [
                    SELECT Id
                    FROM Appointment__c
                    WHERE Consultant__c = :appt.Consultant__c
                    AND Appointment_Date__c = :appt.Appointment_Date__c
                    AND Appointment_Time__c = :appt.Appointment_Time__c
                    AND Id != :appt.Id
                ];

                if(!existingAppointments.isEmpty()) {
                    appt.addError('Consultant is already booked for this time slot');
                }
            }
        }
    }

    if(Trigger.isAfter && Trigger.isInsert) {
        // Send email notifications
        List<Messaging.SingleEmailMessage> emailsToSend = new List<Messaging.SingleEmailMessage>();

        for(Appointment__c appt : Trigger.new) {
            // Query for related records to get email addresses
            Appointment__c apptWithRelations = [
                SELECT Id, Student__r.Email__c, Consultant__r.Email__c,
                       Appointment_Date__c, Appointment_Time__c
                FROM Appointment__c
                WHERE Id = :appt.Id
            ];

            // Create email for student
            if(apptWithRelations.Student__r.Email__c != null) {
                Messaging.SingleEmailMessage studentEmail = new Messaging.SingleEmailMessage();
                studentEmail.setToAddresses(new List<String>{apptWithRelations.Student__r.Email__c});
                studentEmail.setSubject('Appointment Confirmation');
                studentEmail.setPlainTextBody(
                    'Your appointment has been scheduled for ' +
                    apptWithRelations.Appointment_Date__c.format() + ' at ' +
                    apptWithRelations.Appointment_Time__c.format() + '.'
                );
                emailsToSend.add(studentEmail);
            }

            // Create email for consultant
            if(apptWithRelations.Consultant__r.Email__c != null) {
                Messaging.SingleEmailMessage consultantEmail = new Messaging.SingleEmailMessage();
                consultantEmail.setToAddresses(new List<String>{apptWithRelations.Consultant__r.Email__c});
                consultantEmail.setSubject('New Appointment Scheduled');
                consultantEmail.setPlainTextBody(
                    'A new appointment has been scheduled for ' +
                    apptWithRelations.Appointment_Date__c.format() + ' at ' +
                    apptWithRelations.Appointment_Time__c.format() + '.'
                );
                emailsToSend.add(consultantEmail);
            }
        }
```

```
        if(!emailsToSend.isEmpty()) {
            try {
                Messaging.sendEmail(emailsToSend);
            } catch(Exception e) {
                // Handle email sending errors
                System.debug('Error sending appointment notification emails: ' + e.getMessage());
            }
        }
    }
}
```

6. Key Scenarios Addressed by Salesforce in the Implementation Project

- **Multi-Channel Communication:** Personalized communication through email and the platform.
- **Role-Based Access Control:** Maintaining data security and confidentiality through access levels.
- **Reporting and Analytics:** Real-time monitoring of key metrics and KPIs.
- **Integration with Existing Systems (Future Consideration):** Flexibility for future expansion.

7. Conclusion

The "CRM Application to Manage the Services Offered by an Institution" project successfully developed a tailored Salesforce CRM for EduConsultPro Institute, streamlining operations, enhancing communication, and centralizing key information.

With automated processes for admissions and appointment scheduling, the CRM reduces manual work, minimizes errors, and boosts staff efficiency. Automated email notifications keep students informed throughout their engagement, enriching the overall student experience.

By consolidating all relevant data, the CRM empowers EduConsultPro with comprehensive reporting and analytics, enabling data-driven decisions and continuous improvement. The flexible Salesforce platform supports future scalability, positioning EduConsultPro Institute to grow and adapt to evolving needs while delivering efficient, student-centered service management.