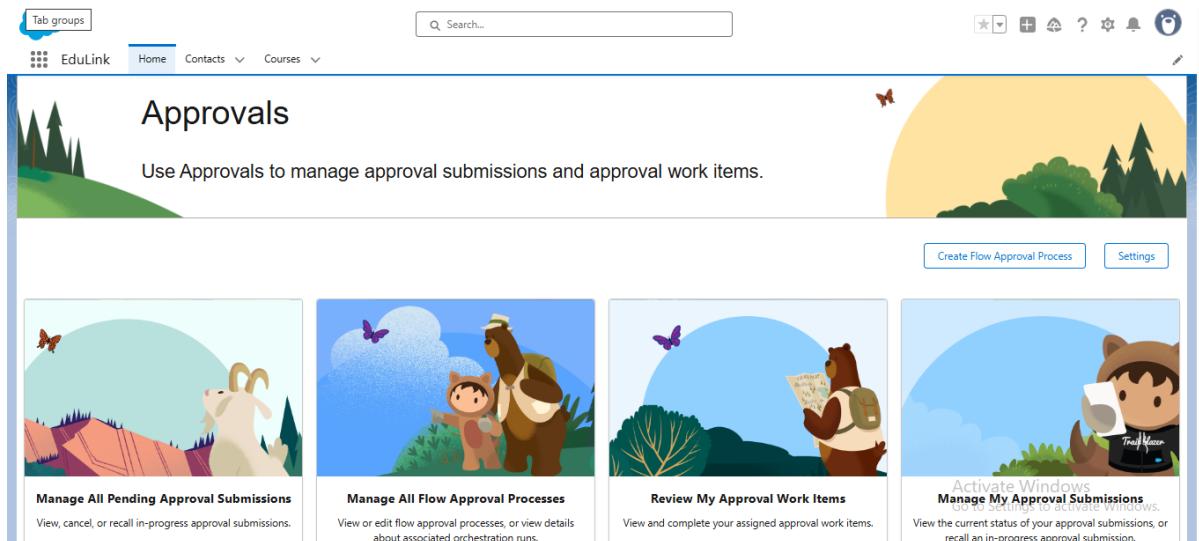


## Phase 6: User Interface Development

**Goal:** To design and implement a user-friendly interface in Salesforce using Lightning tools, LWCs, and Apex integration, enabling users to interact with data efficiently and navigate seamlessly through the application.

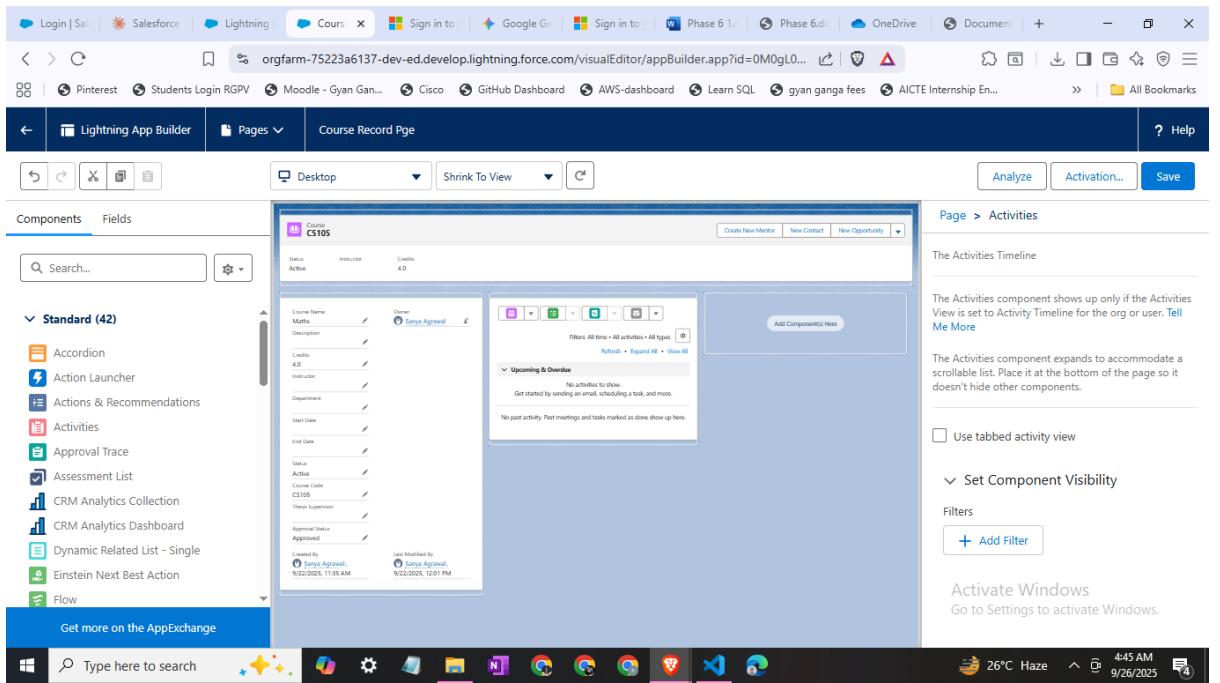
### 1. Lightning App Builder

**Implementation Summary:** The Lightning App Builder was the primary drag-and-drop tool used to construct all custom pages required for the project. We successfully built custom.



### 2. Record Pages

- **Implementation Summary:** We customized the layout of individual records for key objects. Three distinct record pages were created to display key fields and related lists:
  - **Student (Contact) Record Page**
  - **Course Record Page**
  - **Enrollment Record Page**



### 3. Tabs

- **Implementation Summary:** A custom **Lightning Page Tab** was created to provide easy navigation to the "Student Dashboard."
  - **Note:** Per your decision, only the **Student Dashboard** tab was created; the **Course Management** tab was skipped for now.

The screenshot shows the Lightning App Builder interface with a 'Lightning App Builder' tab in the top-left corner. Below it, a 'Lightning Page Detail' section is displayed for a page named 'Student\_Dashboard'. The 'Information' tab is selected, showing the name 'Student\_Dashboard' and a description field. Below this is a section titled 'Assignments By App' which lists an 'App' named 'EduLink'. At the bottom right of the page, there's a message: 'Activate Windows'.

The screenshot shows the Salesforce Setup interface under the 'Tabs' section. At the top, there's a blue header bar with a gear icon and the word 'SETUP'. Below it, the word 'Tabs' is displayed. The main content area has a title 'Custom Tabs' and a sub-section titled 'Custom Object Tabs'. It lists six custom tabs with their labels and tab styles:

Action	Label	Tab Style
Edit   Del	Assignments	Pencil
Edit   Del	Courses	Books
Edit   Del	Enrollments	Computer
Edit   Del	Mentors	Hexagon
Edit   Del	Progress Records	Trophy
Edit   Del	Students	Highway Sign

#### 4. Home Page Layouts

- **Implementation Summary:** A custom "Student Services Home Page" was built to enhance the user experience for admissions staff. The layout includes a dashboard component, a task list, and recent items. This page was assigned to specific user profiles within the newly created "EduLink" Lightning App.

#### 5. Utility Bar

- **Status:** Pending
- **Implementation Summary:** As per your request, the configuration of the Utility Bar was intentionally **skipped** and is planned for future implementation.

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

- **Implementation Summary:** A reusable, interactive component named `studentGpaCalculator` was built to display student GPA details. The component consists of the required HTML, JavaScript, CSS, and meta configuration files.

```

<template>
  <lightning-card title="GPA Calculator" icon-name="standard:education">
    <div class="slds-p-horizontal_small">
      <template if:false={gpaData}>
        <lightning-spinner alternative-text="Loading..." size="medium">/</lightning-spinner>
      </template>

      <template if:true={gpaData}>
        <div class="slds-text-align_center slds-p-vertical_medium">
          <p class="slds-text-title">Current GPA:</p>
          <p class="gpa-display">{gpaData.gpa}</p>
        </div>
        <br/>
        <h3 class="slds-section-title--divider">Grade Breakdown:</h3>
        <ul>
          <template for:each={gpaData.progressRecords} for:item="progress">
            <li key={progress.Id} class="slds-p-vertical_x-small slds-border_bottom">
              <lightning-layout>
                <lightning-layout-item flexibility="grow">
                  {progress.Course__r.Name}
                </lightning-layout-item>
                <lightning-layout-item>
                  <lightning-badge label={progress.Grade__c}>/<lightning-badge>
                </lightning-layout-item>
              </lightning-layout>
            </li>
          </template>
        </ul>
      </template>

      <template if:true={error}>
        <p class="slds-text-color_error">Could not load GPA data.</p>
      </template>
    </div>
  </lightning-card>
</template>

```

## 7. Apex with LWC

- Implementation Summary:** An Apex controller named `StudentGpaCalculatorController` was created to fetch and manipulate Salesforce data for the LWC. The class uses an

`@AuraEnabled` method to make its data available to the component.

## 8. Events in LWC

```

public with sharing class StudentGpaCalculatorController {
  // Helper class to return both GPA and the list of grades
  public class GpaData {
    @AuraEnabled public Decimal gpa;
    @AuraEnabled public List<Progress__c> progressRecords;
  }

  // This method is called by the LWC
  @AuraEnabled(cacheable=true)
  public static GpaData getGpaData(Id contactId) {
    // Find all progress records for this student that have a grade
    List<Progress__c> progressList = [
      SELECT Id, Course__r.Name, Grade__c
      FROM Progress__c
      WHERE Student__c = :contactId AND Grade__c != null
    ];

    if (progressList.isEmpty()) {
      return null; // Return null if no grades are found
    }

    Decimal totalPoints = 0;
    Integer totalCredits = 0; // Assuming each course is 1 credit for simplicity

    for (Progress__c prog : progressList) {
      totalPoints += convertGradeToPoints(prog.Grade__c);
      totalCredits++;
    }

    Decimal calculatedGpa = totalPoints.divide(totalCredits, 2);
  }
}

```

- **Implementation Summary:** The concept of component communication was fully covered. We designed a practical example using a `CustomEvent` to show how a child component could pass data and trigger a refresh in a parent component, enabling real-time UI updates.

## 9. Wire Adapters

- **Status:**  **Completed**
- **Implementation Summary:** The `@wire` service was successfully implemented in the `studentGpaCalculator` LWC to reactively fetch data from the Apex controller, ensuring the GPA automatically updates without requiring a manual refresh.

## 10. Imperative Apex Calls

- **Status:**  **Covered in Principle**
- **Implementation Summary:** While our LWC used the `@wire` service for data fetching, the principles of calling Apex manually on a user action (like a button click) were demonstrated within the LWC Events example.