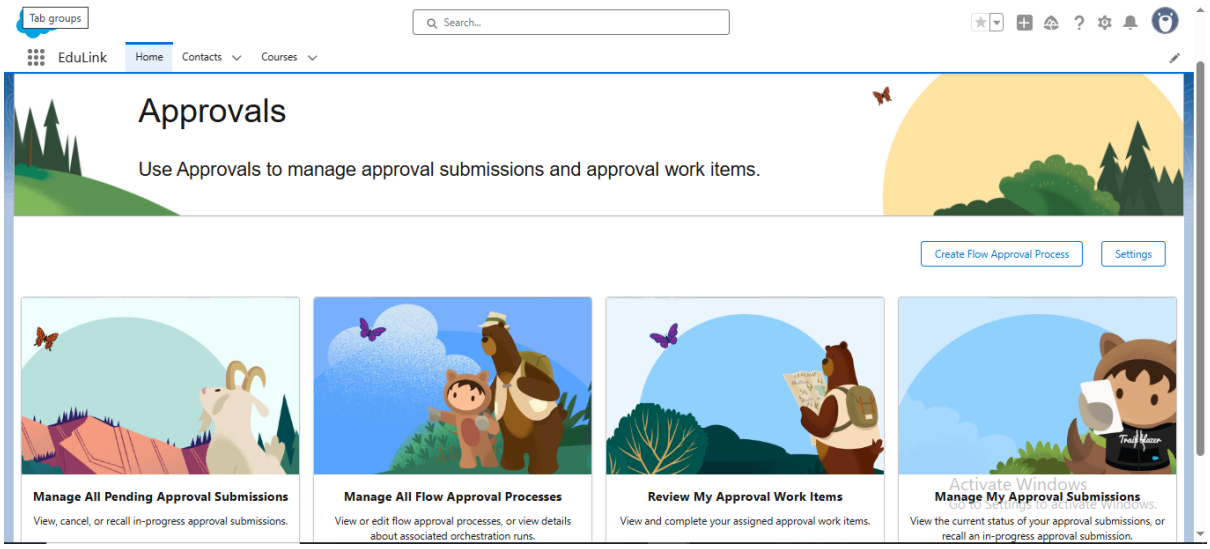
Phase 6: User Interface Development

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.

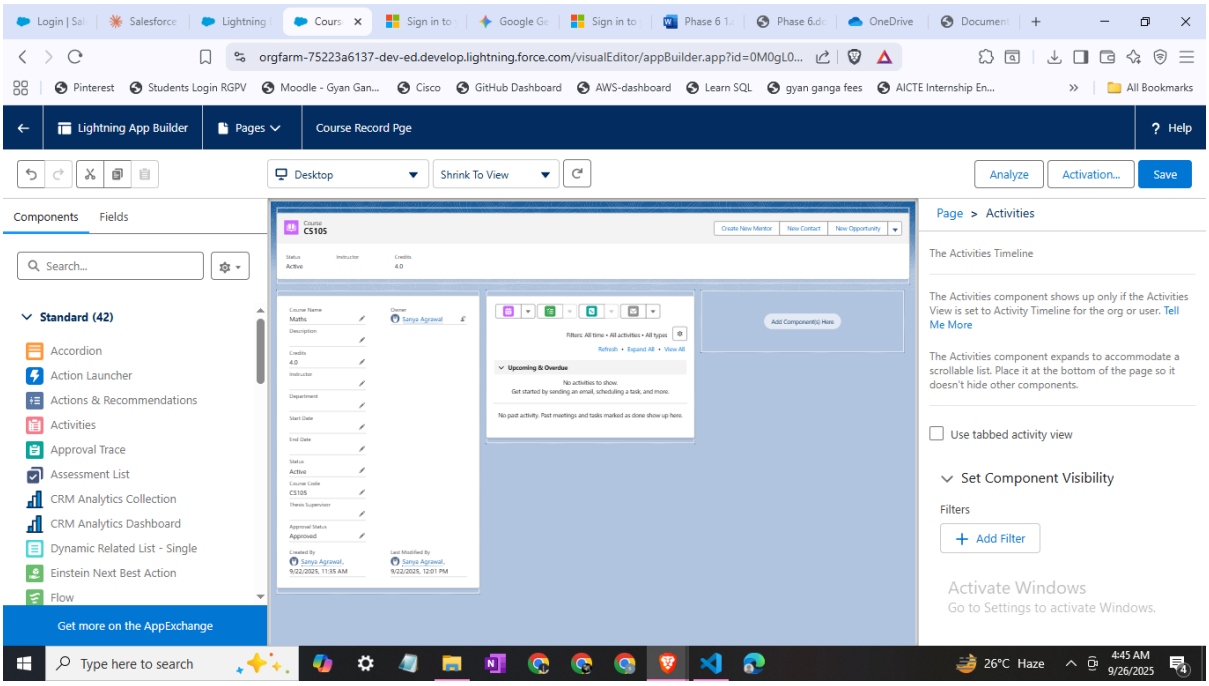
1. Lightning App Builder

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.



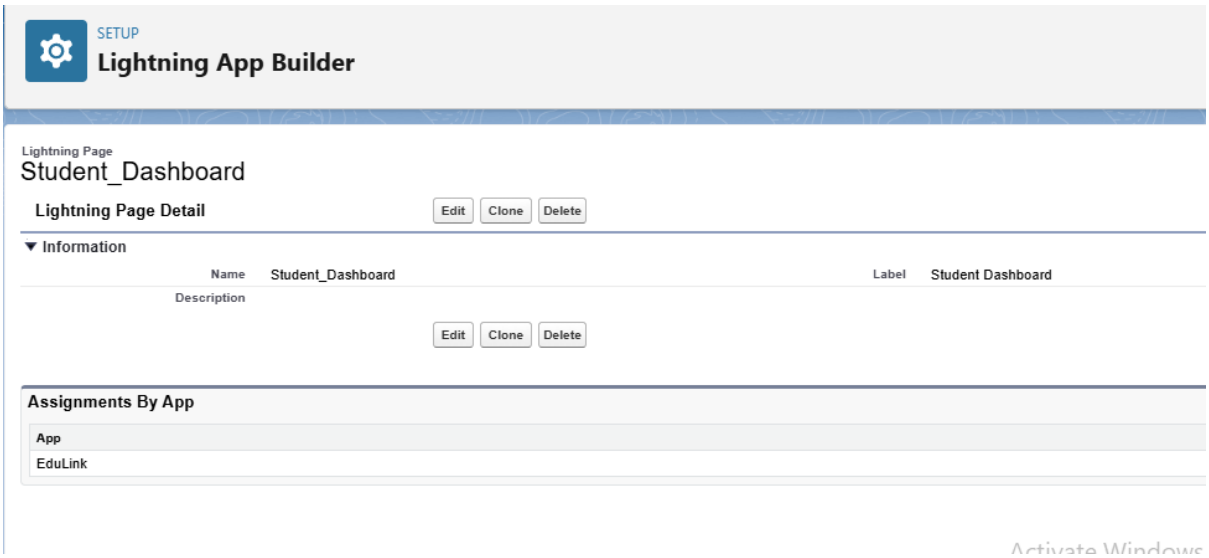
2. Record Pages

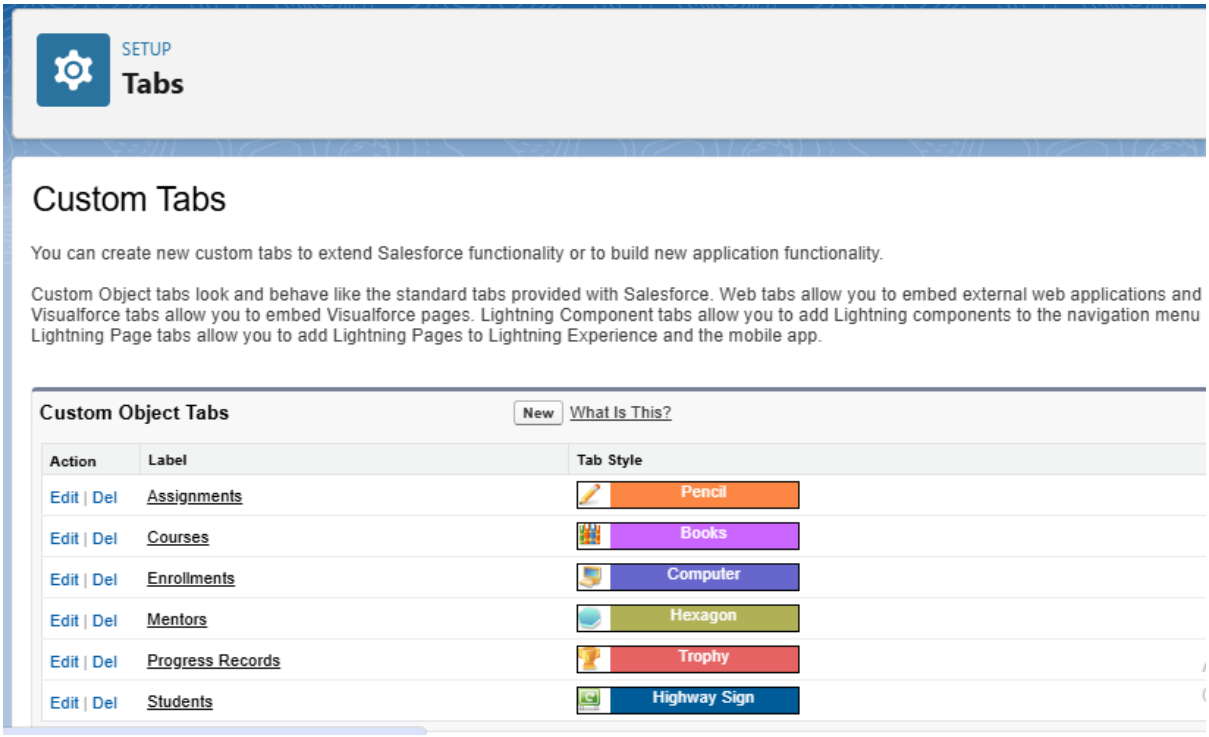
- **Implementation Summary:** We customized the layout of individual records for key objects. Three distinct record pages were created to display key fields and related lists:
  - **Student (Contact) Record Page**
  - **Course Record Page**
  - **Enrollment Record Page**



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.





4. Home Page Layouts

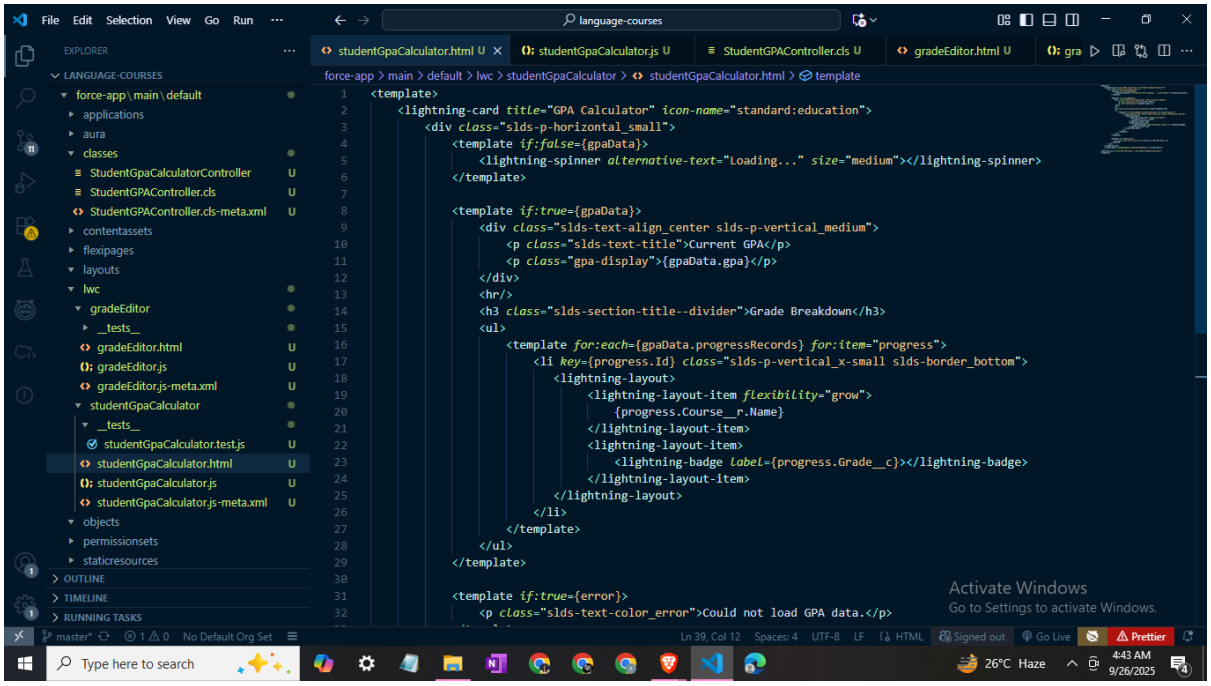
- **Implementation Summary:** A custom **"Student Services Home Page"** was built to enhance the user experience for admissions staff. The layout includes a dashboard component, a task list, and recent items. This page was assigned to specific user profiles within the newly created "EduLink" Lightning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

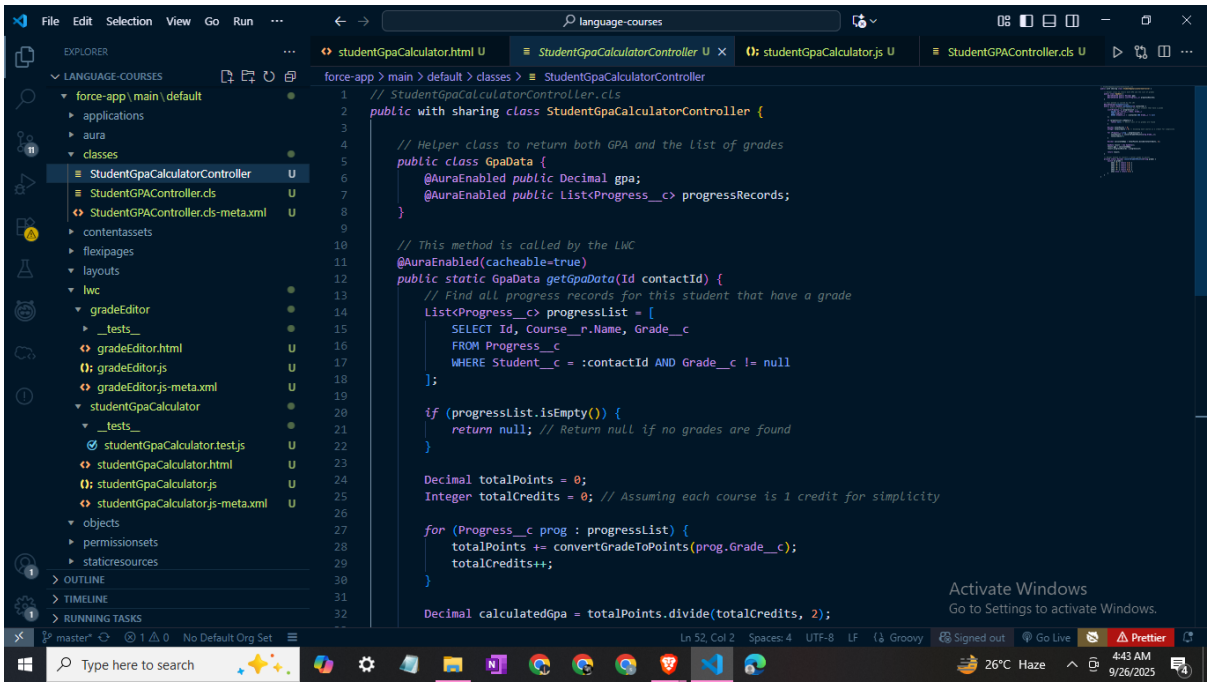


7. Apex with LWC

- **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



- **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.