Student Management System Using Salesforce CRM

Phase 1: Problem Understanding & Industry Analysis

1. Requirement Gathering

Educational institutions require a system to manage student details, course enrollments, attendance, grades, and fees in a centralized manner. The requirements were gathered from administrators, teachers, and students to identify challenges in manual management.

2. Stakeholder Analysis

- **Students** – Need easy enrollment, grade tracking, and fee updates.
- **Teachers** – Need to record attendance, manage grades, and monitor performance.
- **Administrators** – Require centralized control over student data, reporting, and fee collection.
- **Parents/Guardians (Future)** – Require visibility into student performance and attendance.

3. Business Process Mapping

- **Student Admission** → Creation of student record.
- **Course Enrollment** → Mapping students to multiple courses.
- **Attendance Tracking** → Daily attendance marking per course.
- **Grade Management** → Entering marks and calculating GPA.
- **Fee Management** → Recording payments, balances, and due alerts.
- **Reporting & Analytics** → Generating performance, fee, and attendance reports.

4. Industry-specific Use Case Analysis

- Current systems in schools and colleges rely on **spreadsheets/manual records**, causing redundancy and errors.
- Industry trend is shifting to **cloud-based CRMs** like Salesforce for data accuracy and accessibility.
- Similar use cases:
    - University student portals.

- Training centers managing course enrollments.
- Coaching institutes tracking performance.

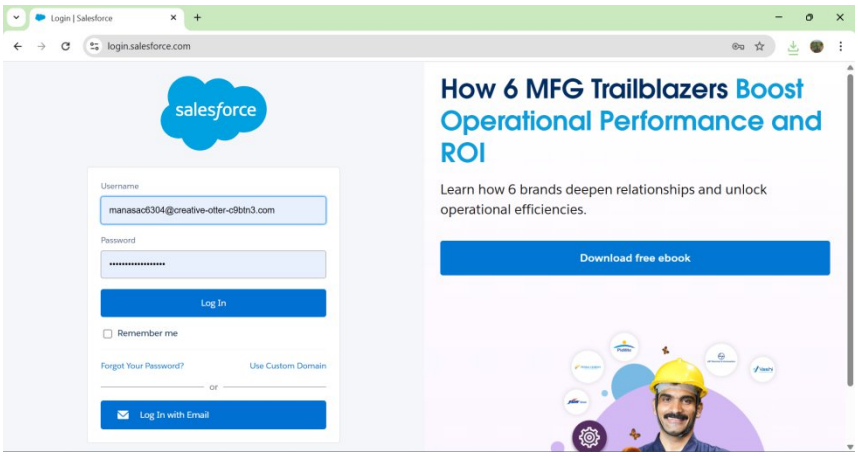
5. AppExchange Exploration

- Explored Salesforce **AppExchange** solutions for education management.
- Found apps like:
  - **Salesforce Education Cloud** – Comprehensive education CRM.
  - **Student Success Hub** – Focused on K-12 student outcomes.
  - **Attendance & Fee Management Apps** – Useful but often paid solutions.
- Decided to **build a custom Student Management System** tailored for specific institutional needs instead of relying fully on prebuilt paid apps.

Phase 2: Org Setup & Configuration – Student Management System

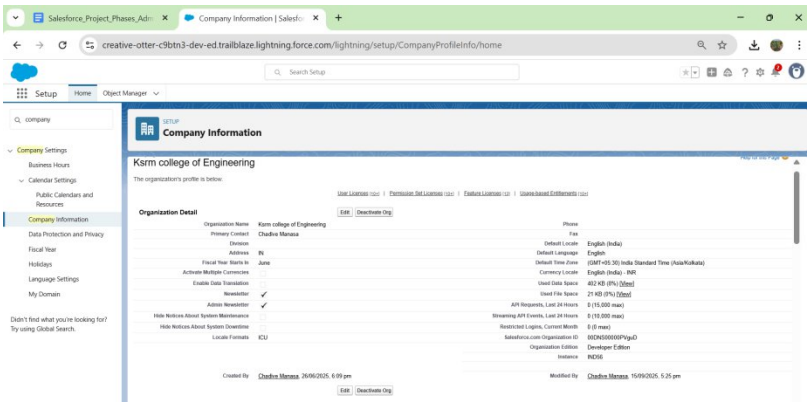
1. Salesforce Edition

- **Developer Edition** (Dev Org) used for project setup.



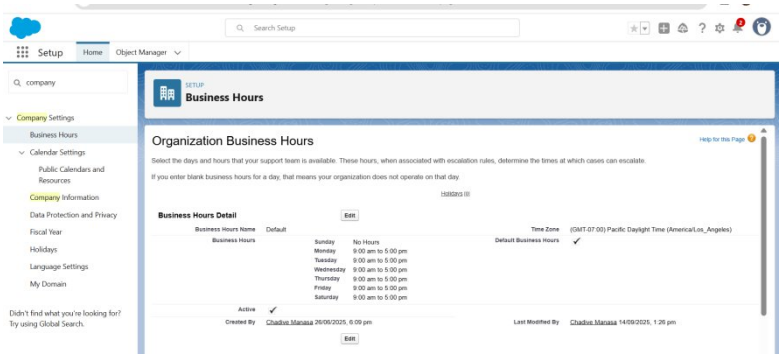
2. Company Profile Setup

- Configured **Institution Name, Address, Default Time Zone, Currency, and Locale**.

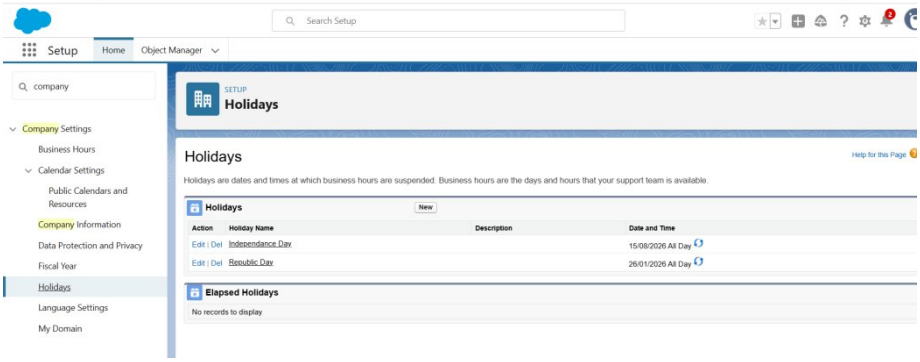


3. Business Hours & Holidays

- Defined **Standard Business Hours** (9 AM – 6 PM, Mon–Sat).

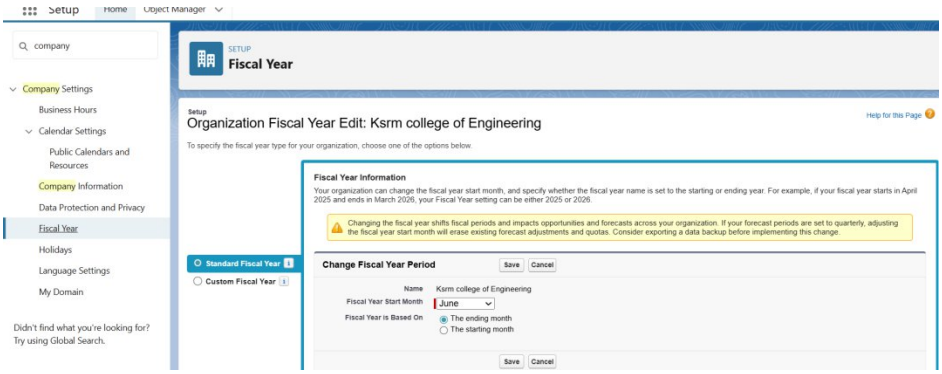


- Added **holidays** (e.g., Independence Day, New Year).



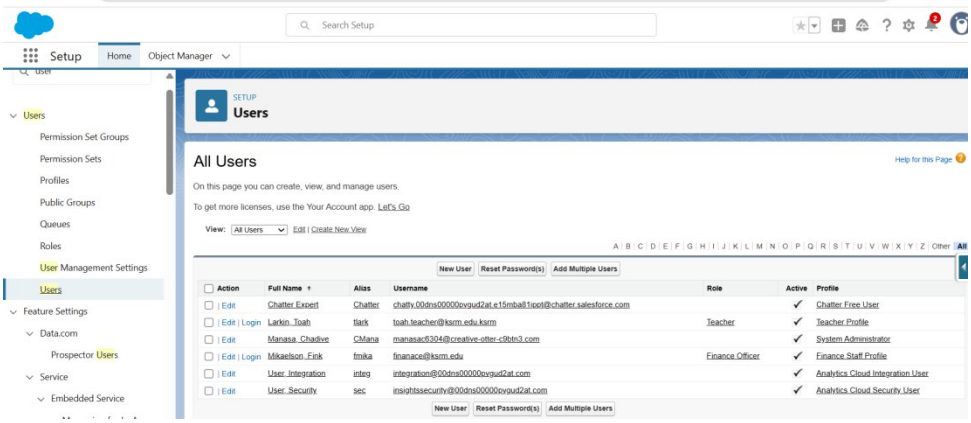
4. Fiscal Year Settings

- Configured **Standard Fiscal Year** (Jan–Dec).

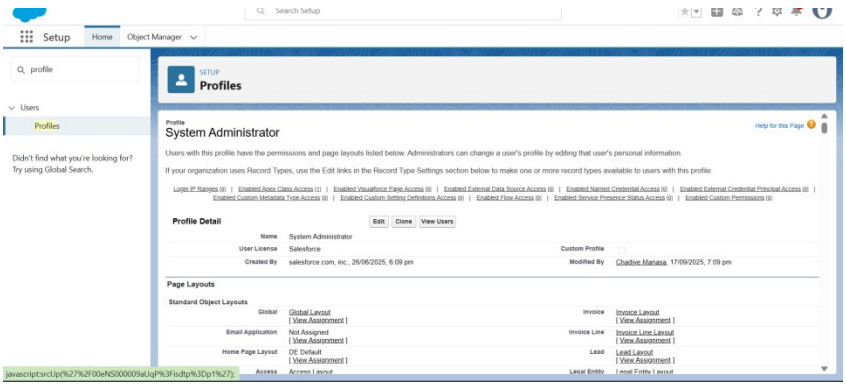


5. User Setup & Licenses

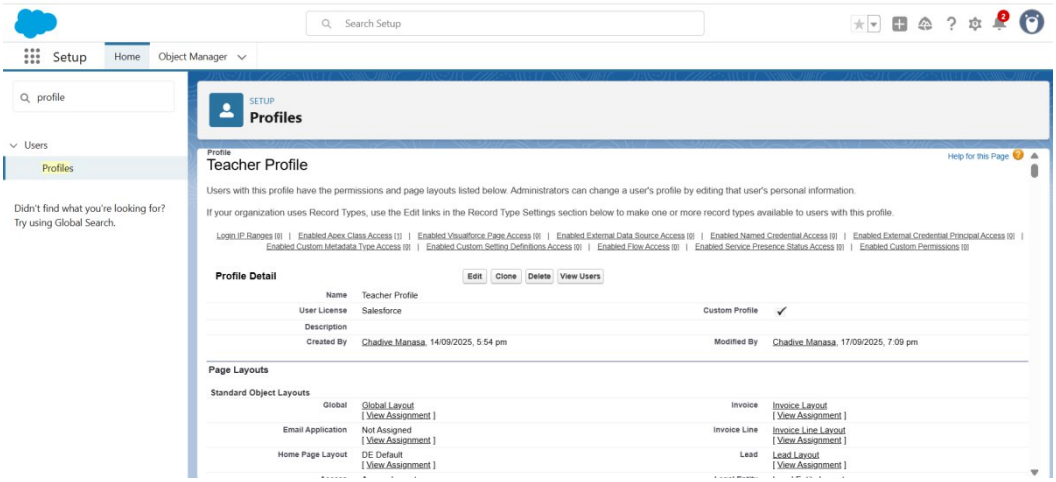
- Created sample users with **Salesforce licenses**:
    - **System Administrator** – Full access.
    - **Faculty User** – Limited teaching access.
    - **Student User** – Read-only/student data access.



6. Profiles

- Created/Modified Profiles:
    - **System Administrator** (default full access).
- 

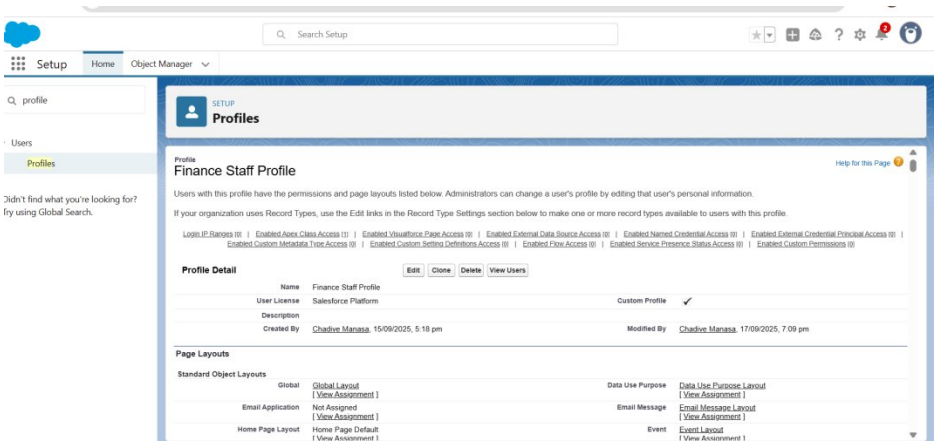
    - **Teacher Profile** – Access to Students, Courses, Attendance, Grades.



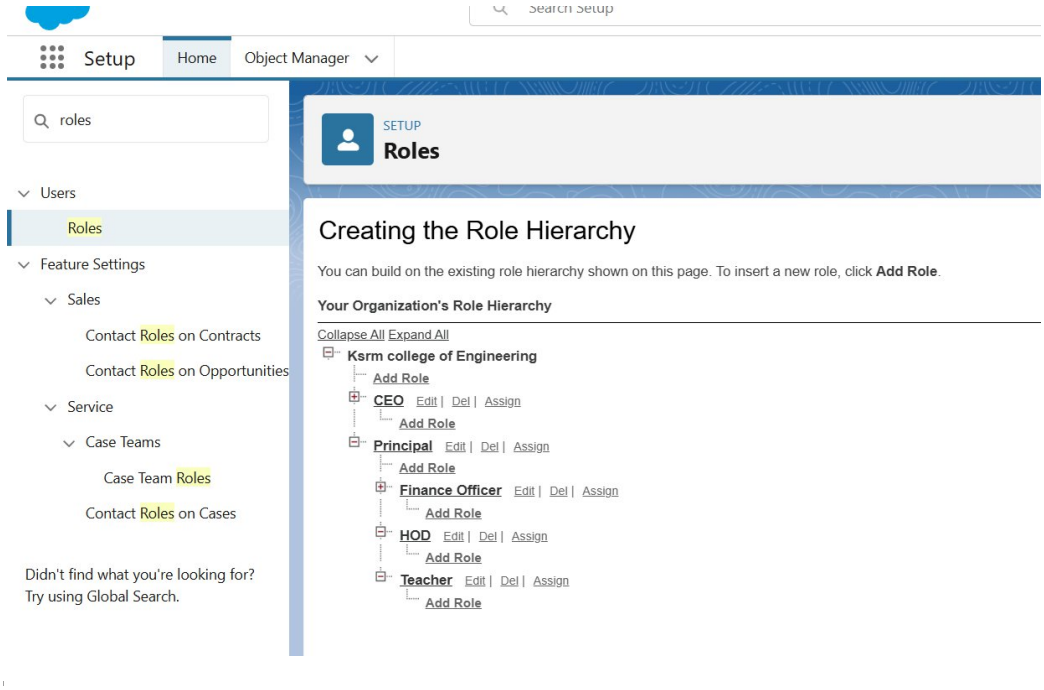
- **Finance Profile** – Access to Fees object.

7. Roles

- Defined Role Hierarchy:
  - **Principal (Top Role)**
    - HOD
    - Finance Officer



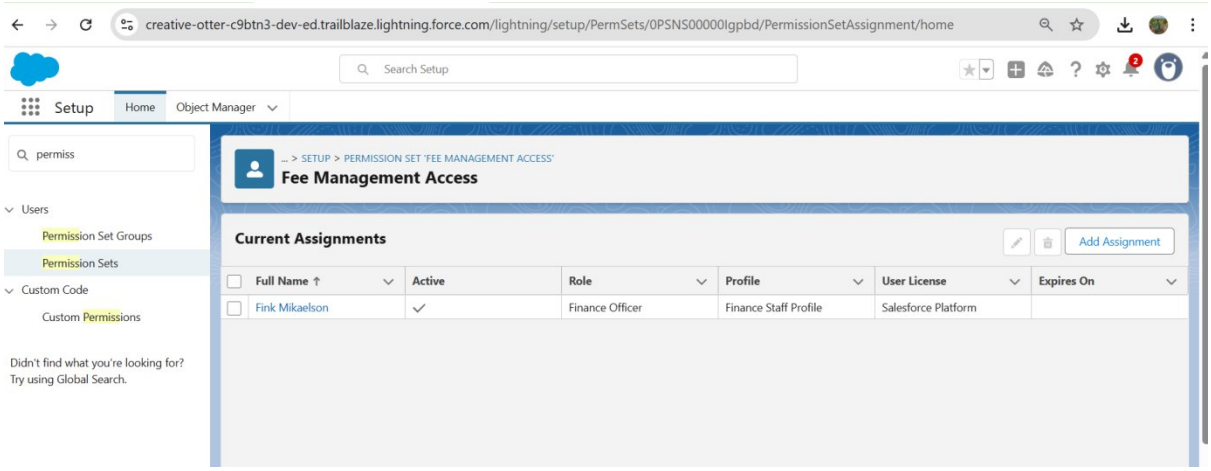
- Teacher



8

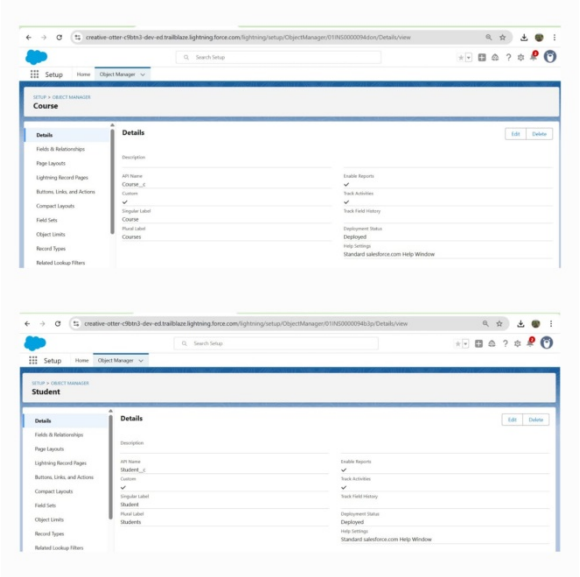
8.Permission Sets

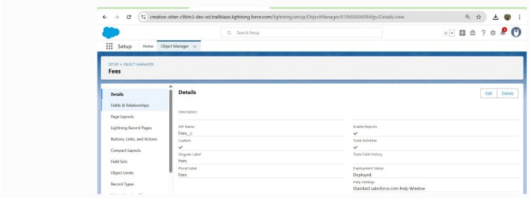
- Created Permission Sets for additional access:
    - **Report Access** – Extra reporting permissions.
    - **Fee Management Access** – Extended access to Fees object.



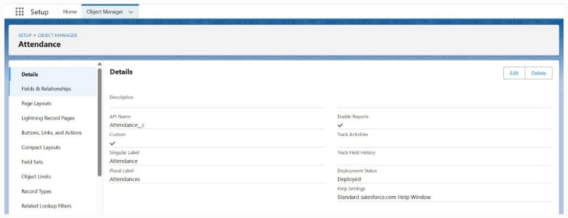
9. Objects Created

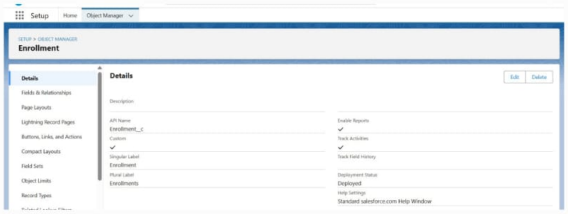
- **Student (Custom Object)** – Manage student details.
- **Course (Custom Object)** – Store course information.
- **Enrollment (Junction Object)** – Relationship between Student & Course.
- **Attendance (Custom Object)** – Track daily student attendance.
- **Grades (Custom Object)** – Store marks, grades, GPA.
- **Fees (Custom Object)** – Manage payments and pending balances.

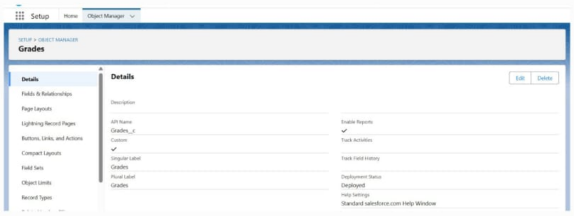




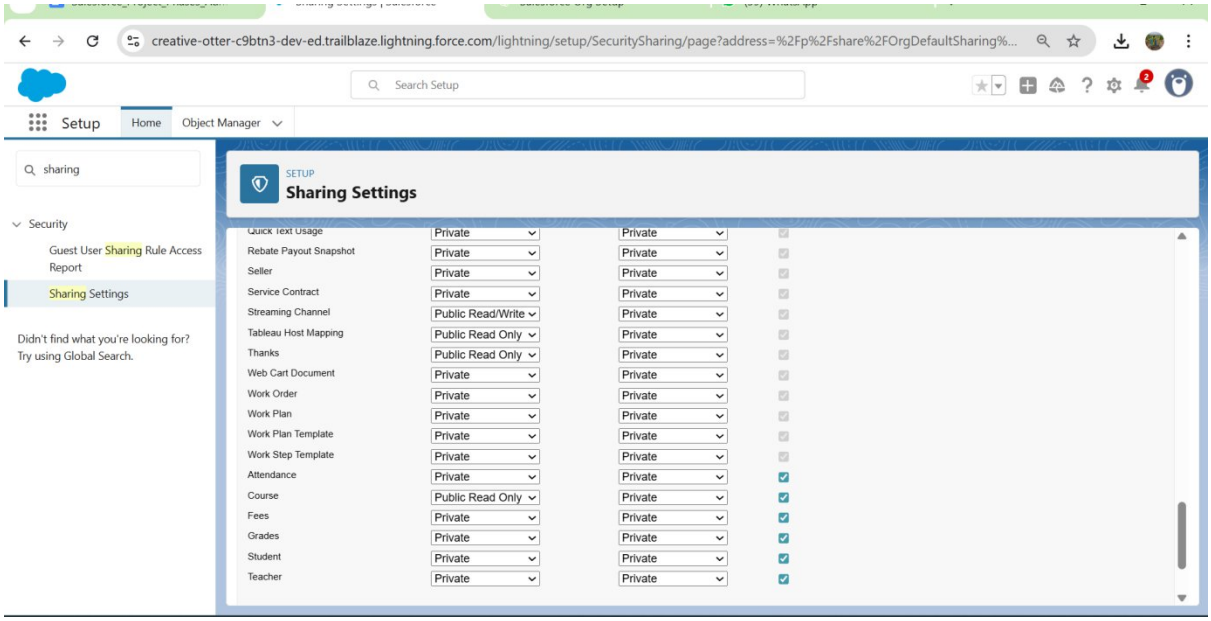
10. OWD (Org-Wide Defaults)





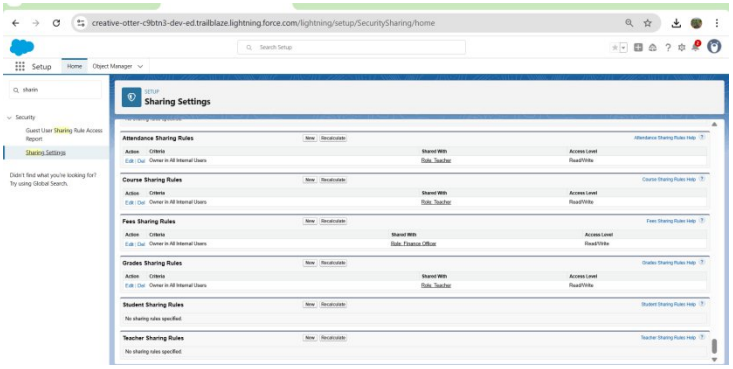


- Student → Private
- Course → Only Public Read
- Enrollment → Private
- Attendance → Private
- Grades → Private
- Fees → Private



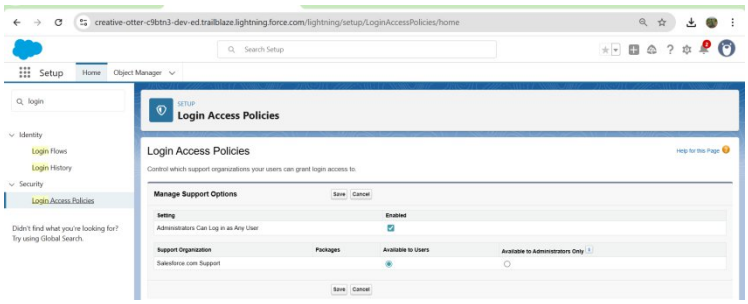
11. Sharing Rules

- Shared **Courses** with all Faculty.
- Shared **Student Records** read-only with Faculty (except Finance objects).
- Finance users granted access to **Fees object**.



12. Login Access Policies

- Enabled **Administrators Can Log in as Any User**.
- Configured **Trusted IP Ranges** for secure login.

13. 

Dev Org Setup

- Enabled **Dev Hub** in Developer Org.

- Connected CLI:
- sf org login web --set-default-dev-hub --alias DevHub

14. Sandbox Usage

- **Not available** in Developer Edition.
- Documented sandbox usage for real-world projects (Dev, Test, UAT).

15. Deployment Basics

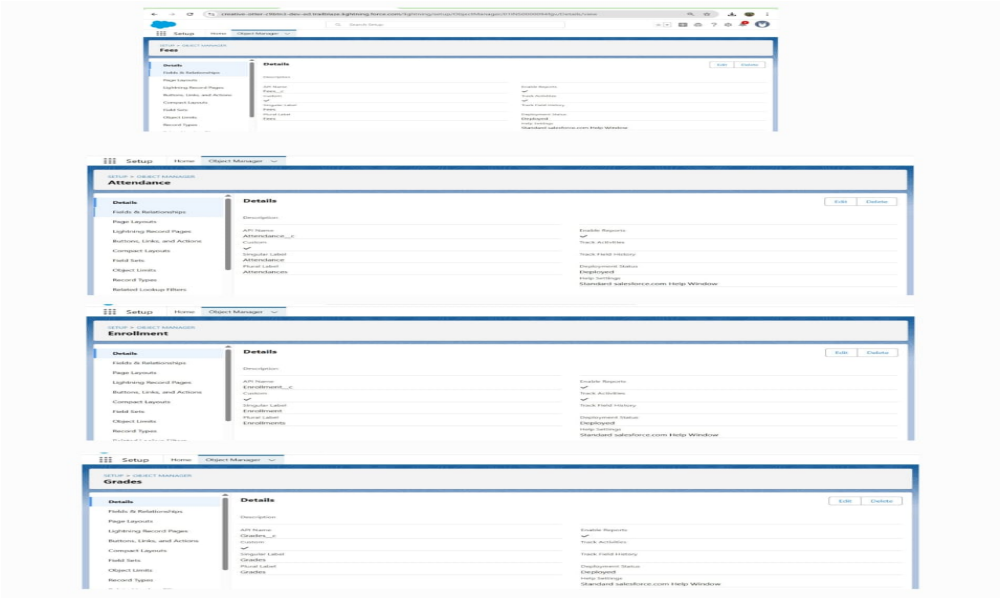
- Practiced deployment commands:
- sf project retrieve start --manifest manifest/package.xml -o DevHub
- sf project deploy start --target-org ScratchOrg
- Source code and configuration stored in **GitHub repo** for version control.

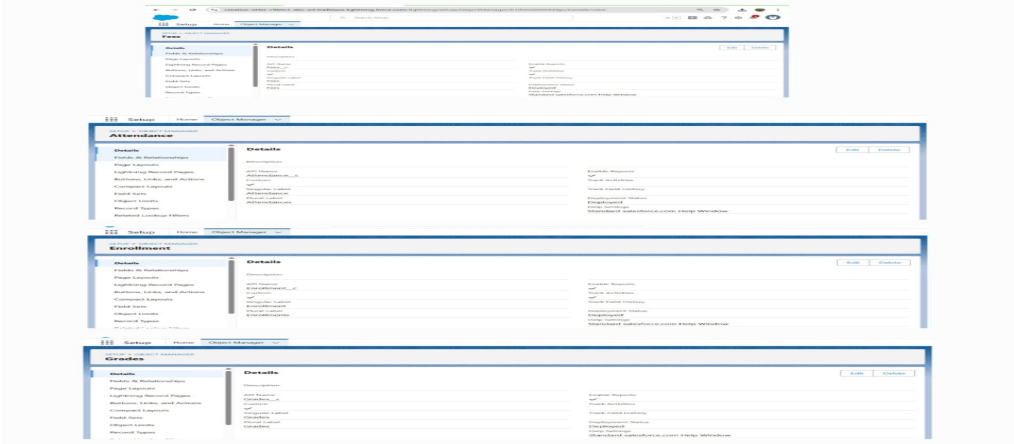
Phase 3: Data Modeling & Relationships

1. Standard & Custom Objects

In this project, we used standard objects like Account and Contact, and created custom objects such as Student, Course, and Enrollment.

Custom objects were designed to represent our business data model, where Students can enroll in multiple Courses through the Enrollment object.

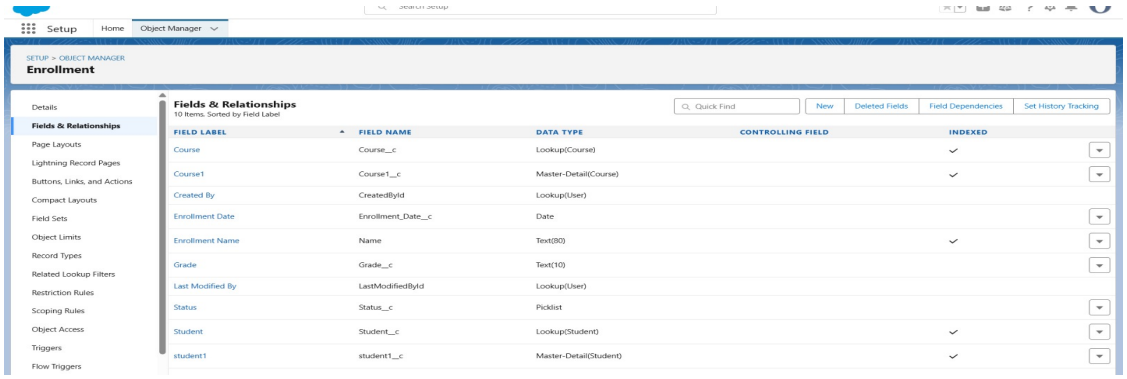


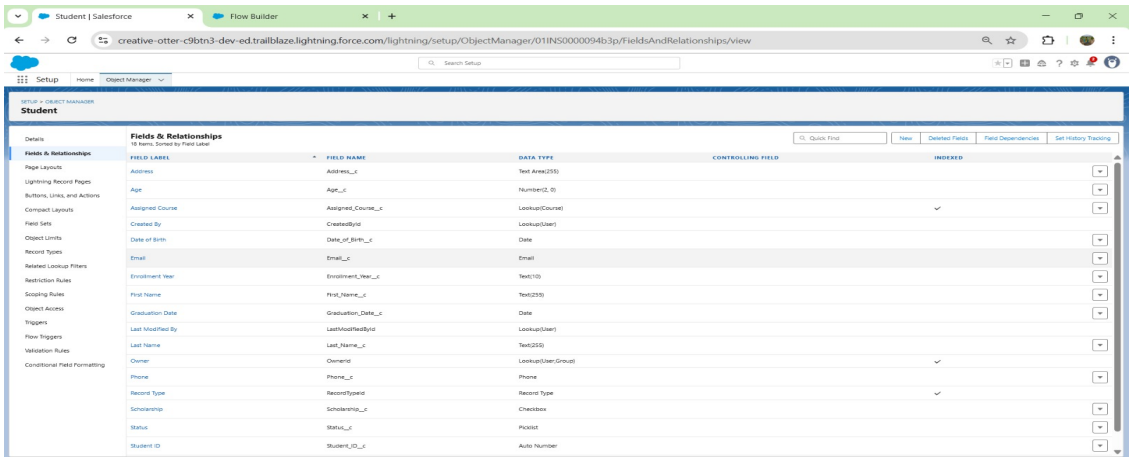


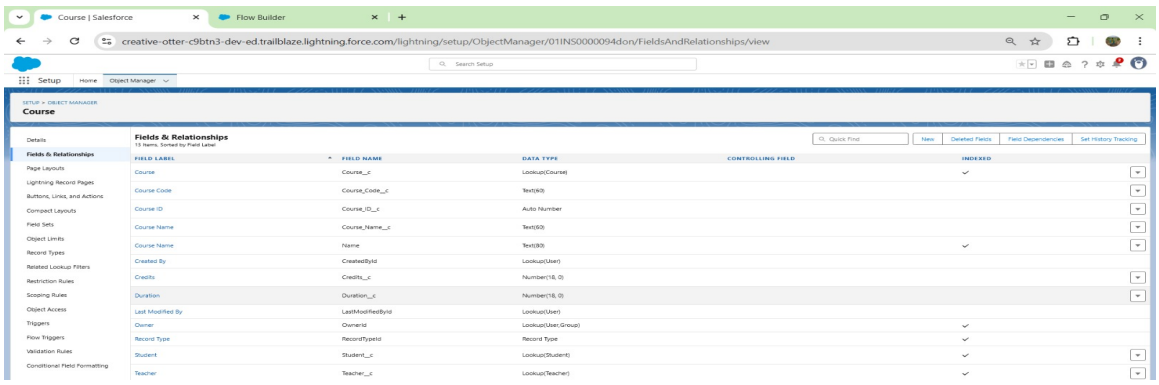
2. Fields

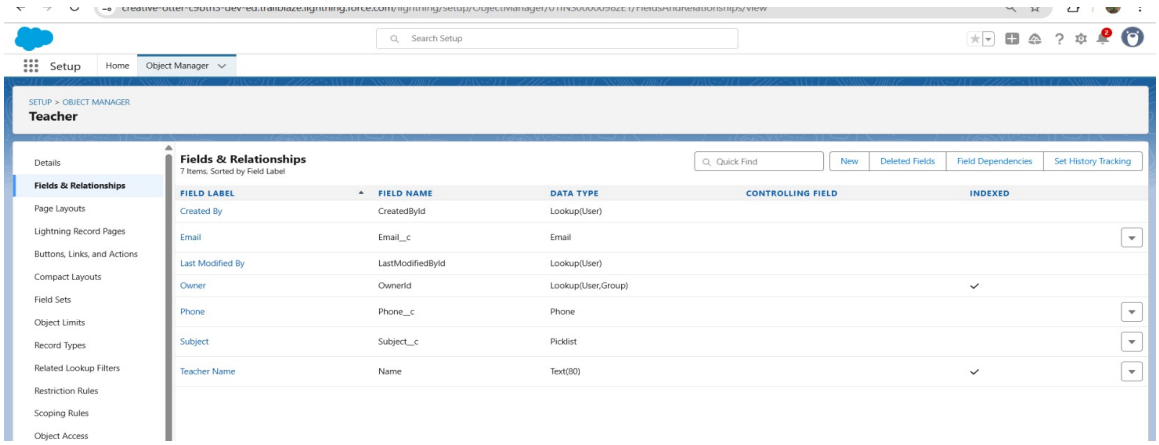
Custom fields were added to capture additional data for each object. Examples include:

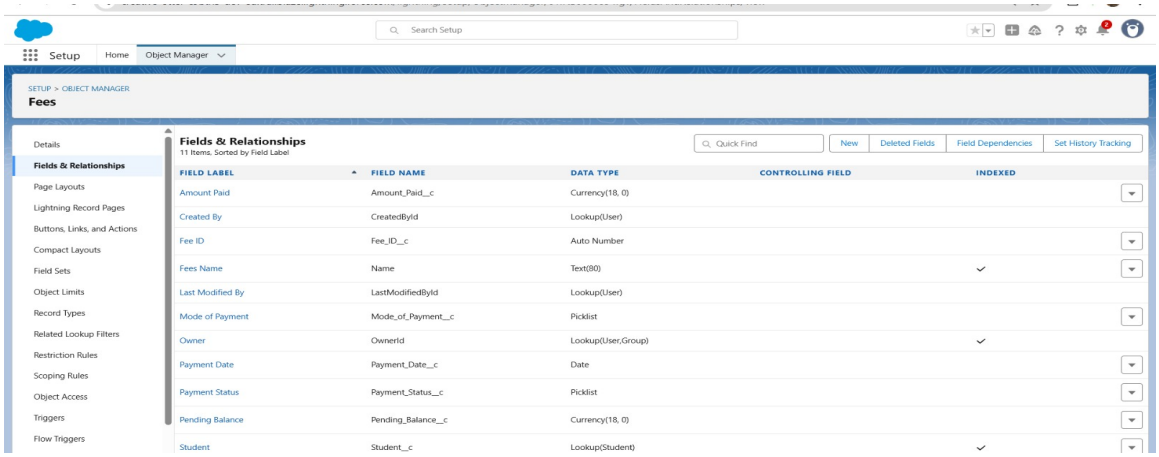
- Student: First Name, Last Name, Email, Phone, Department (Picklist), Roll Number.
- Course: Course Name, Course Code, Duration.
- Enrollment: Enrollment Date, Status.

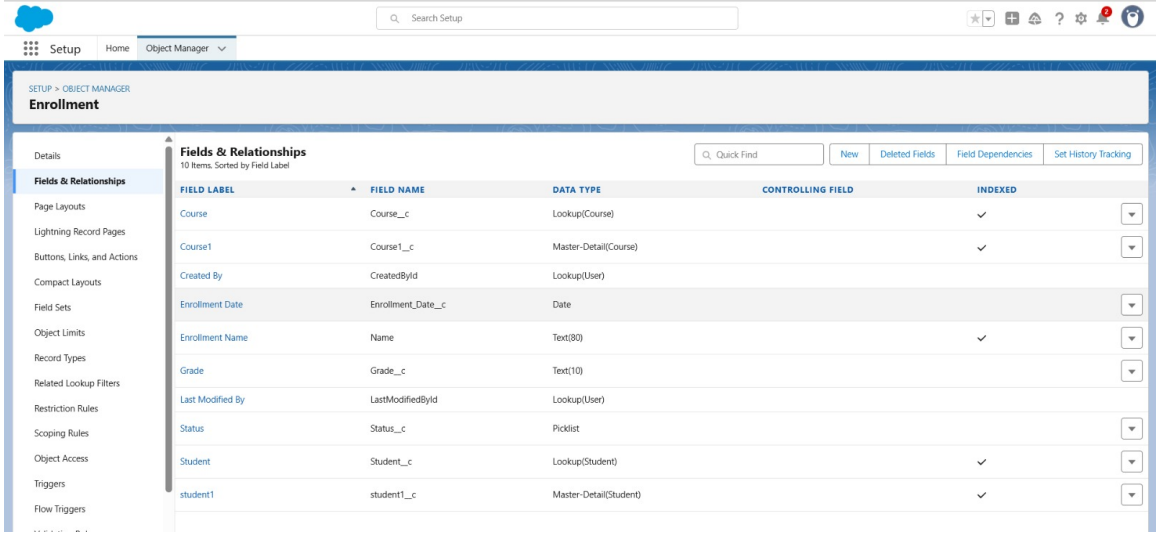






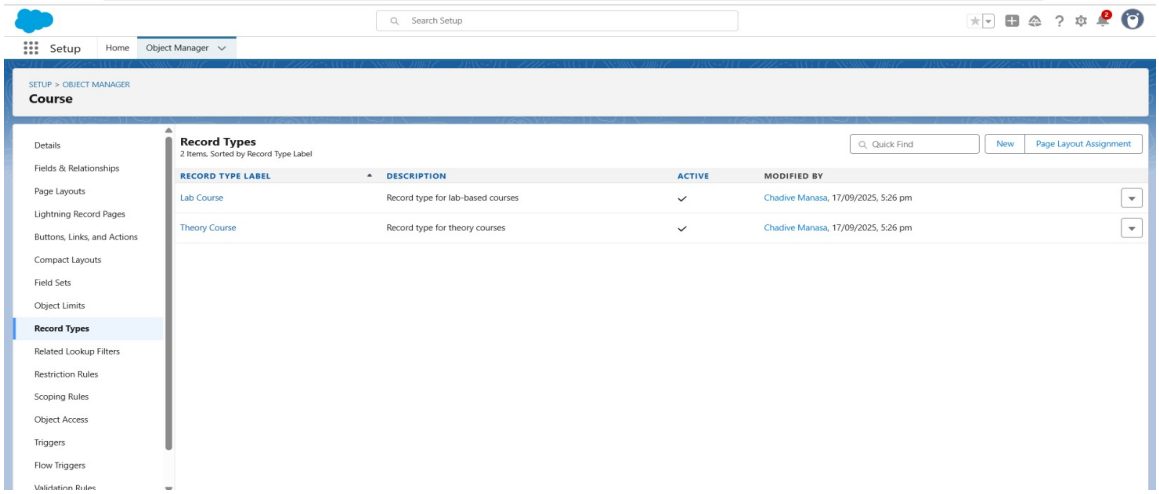

Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Teacher
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Restriction Rules
Scoping Rules
Object Access
Fields & Relationships
7 Items, Sorted by Field Label
Quick Find
New
Deleted Fields
Field Dependencies
Set History Tracking
FIELD LABEL
FIELD NAME
DATA TYPE
CONTROLLING FIELD
INDEXED
Created By
CreatedById
Lookup(User)
Email
Email__c
Email
Last Modified By
LastModifiedById
Lookup(User)
Owner
OwnerId
Lookup(User,Group)
Phone
Phone__c
Phone
Subject
Subject__c
Picklist
Teacher Name
Name
Text(80)


Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Fees
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Restriction Rules
Scoping Rules
Object Access
Triggers
Flow Triggers
Fields & Relationships
11 Items, Sorted by Field Label
Quick Find
New
Deleted Fields
Field Dependencies
Set History Tracking
FIELD LABEL
FIELD NAME
DATA TYPE
CONTROLLING FIELD
INDEXED
Amount Paid
Amount_Paid__c
Currency(18, 0)
Created By
CreatedById
Lookup(User)
Fee ID
Fee_ID__c
Auto Number
Fees Name
Name
Text(80)
Last Modified By
LastModifiedById
Lookup(User)
Mode of Payment
Mode_of_Payment__c
Picklist
Owner
OwnerId
Lookup(User,Group)
Payment Date
Payment_Date__c
Date
Payment Status
Payment_Status__c
Picklist
Pending Balance
Pending_Balance__c
Currency(18, 0)
Student
Student__c
Lookup(Student)


Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Enrollment
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Restriction Rules
Scoping Rules
Object Access
Triggers
Flow Triggers
Fields & Relationships
10 Items, Sorted by Field Label
Quick Find
New
Deleted Fields
Field Dependencies
Set History Tracking
FIELD LABEL
FIELD NAME
DATA TYPE
CONTROLLING FIELD
INDEXED
Course
Course__c
Lookup(Course)
Course1
Course1__c
Master-Detail(Course)
Created By
CreatedById
Lookup(User)
Enrollment Date
Enrollment_Date__c
Date
Enrollment Name
Name
Text(80)
Grade
Grade__c
Text(10)
Last Modified By
LastModifiedById
Lookup(User)
Status
Status__c
Picklist
Student
Student__c
Lookup(Student)
student1
student1__c
Master-Detail(Student)

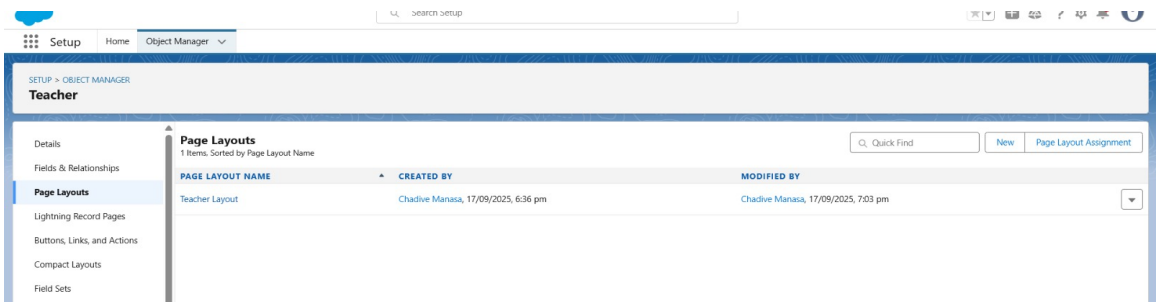
3. Record Types

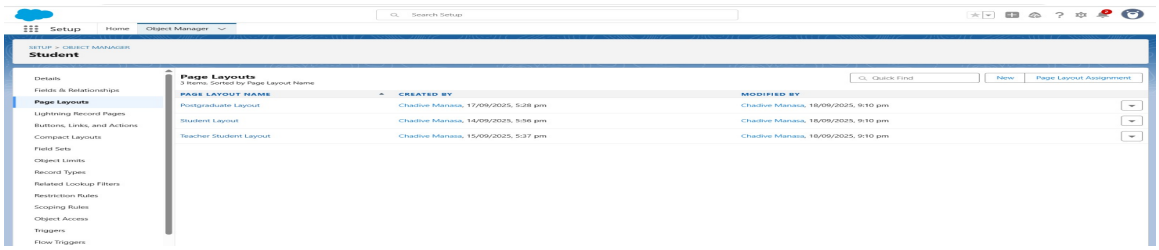
Record types were created on the Course object to differentiate between Online and Offline courses. Each record type uses a specific page layout to show relevant fields.

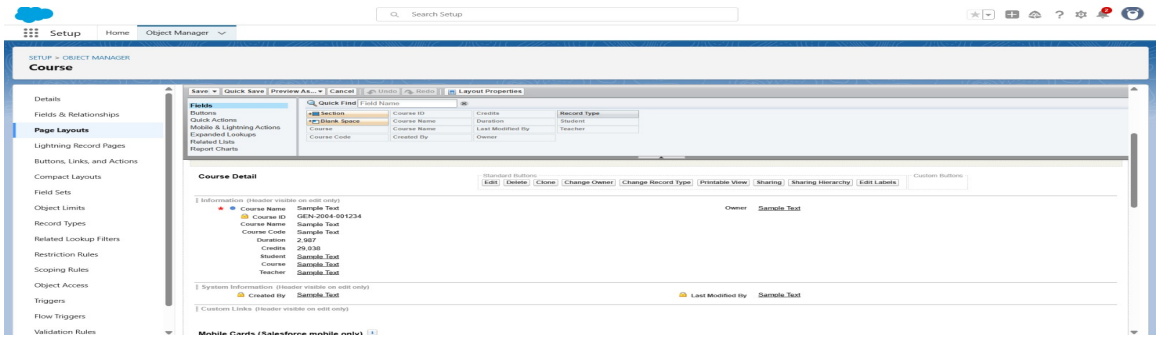


4. Page Layouts

Page layouts were customized to organize fields and related lists in a user-friendly way. Different page layouts were assigned based on record type.

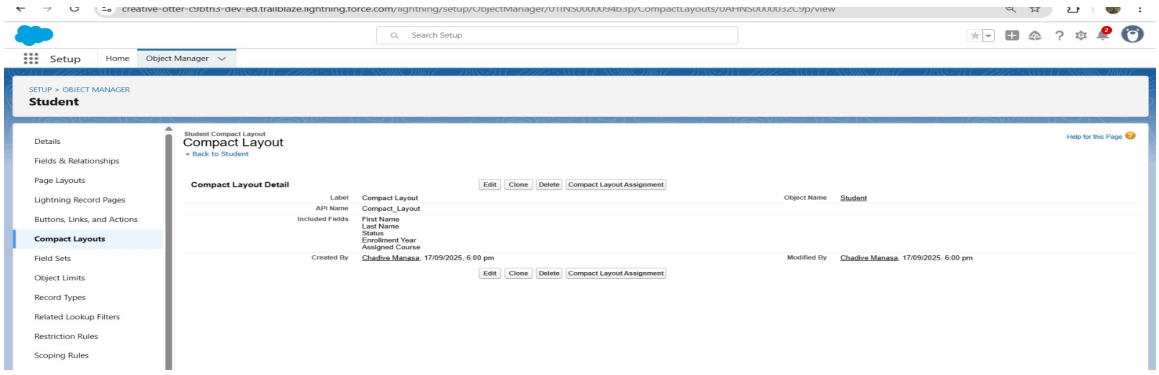






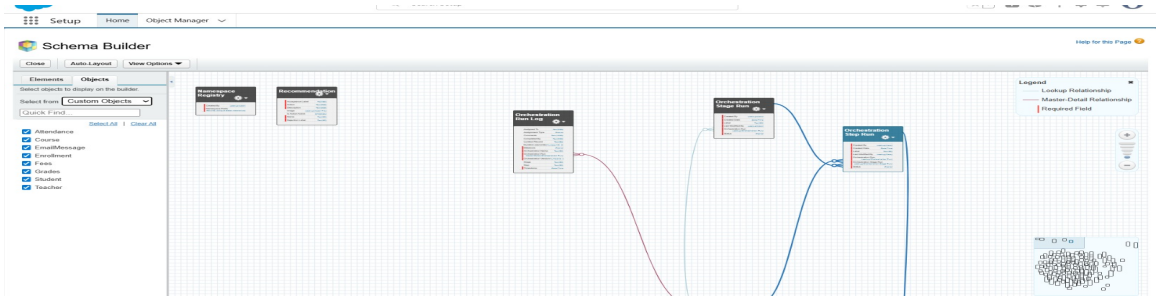
5. Compact Layouts

Compact layouts were configured to display key information (Name, Email, Department for Student) in the highlights panel.



6. Schema Builder

Schema Builder was used to visualize relationships between Student, Course, and Enrollment objects, making it easier to validate the data model.



7. Lookup vs Master-Detail vs Hierarchical Relationships

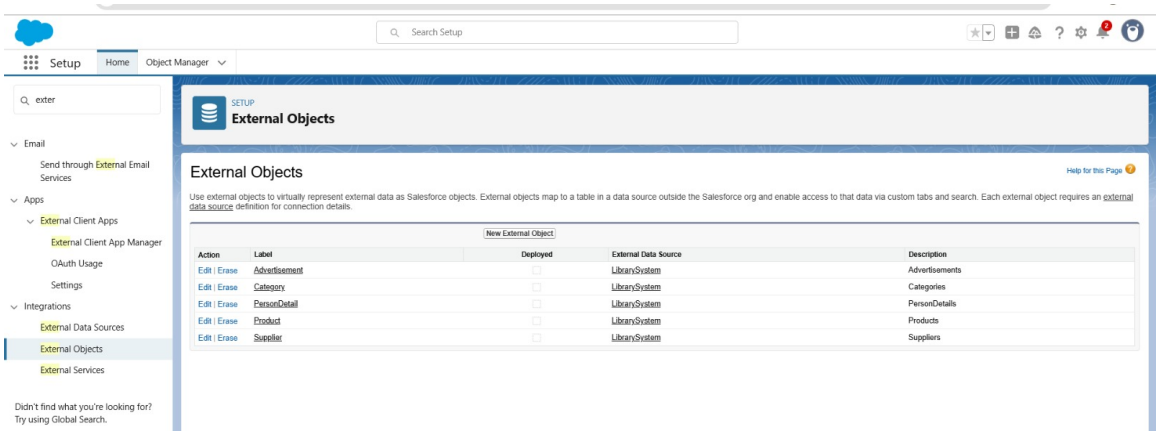
Relationships were created as follows:
- Student ↔ Course: Many-to-Many relationship implemented using the Enrollment junction object.
- Lookup relationships were used where child records can exist independently.
- Master-Detail relationships were used for Enrollment to ensure cascading delete behavior and roll-up summary fields.

8. Junction Objects

Enrollment object was created as a junction object with two master-detail fields:
- Student (Master-Detail)
- Course (Master-Detail)

9. External Objects

An external object was configured to fetch reference data about available training materials from an external system, allowing real-time access without data duplication.



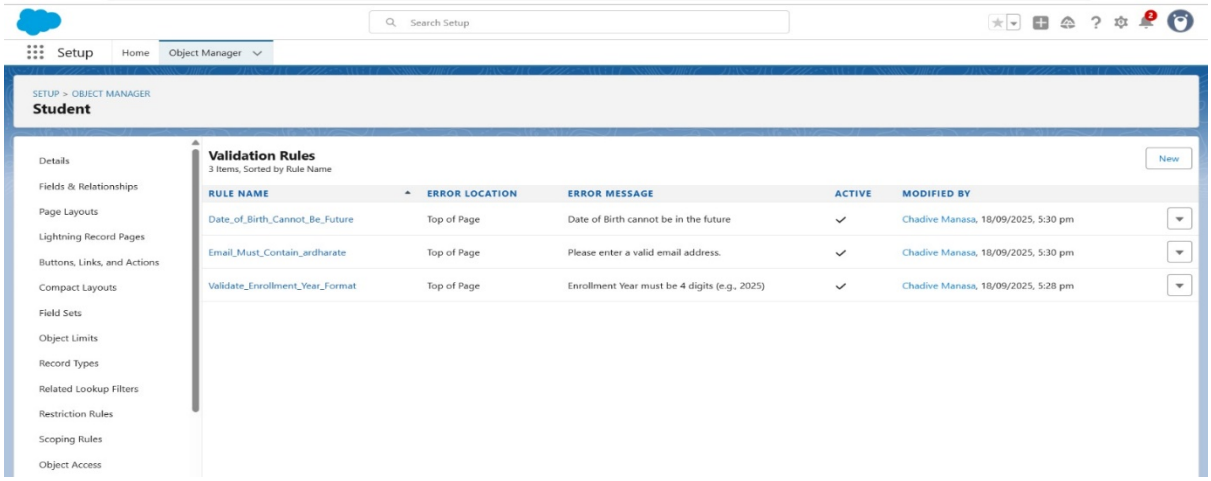
Phase 4: Process Automation (Admin)

Goal: The goal of this phase is to automate repetitive administrative tasks in the **Student Management System**. Using Salesforce automation tools, we ensure student records are accurate, status updates are automatic, reminders are sent on time, and approval requests are streamlined without manual effort.

1. Validation Rules

Object: Student__c

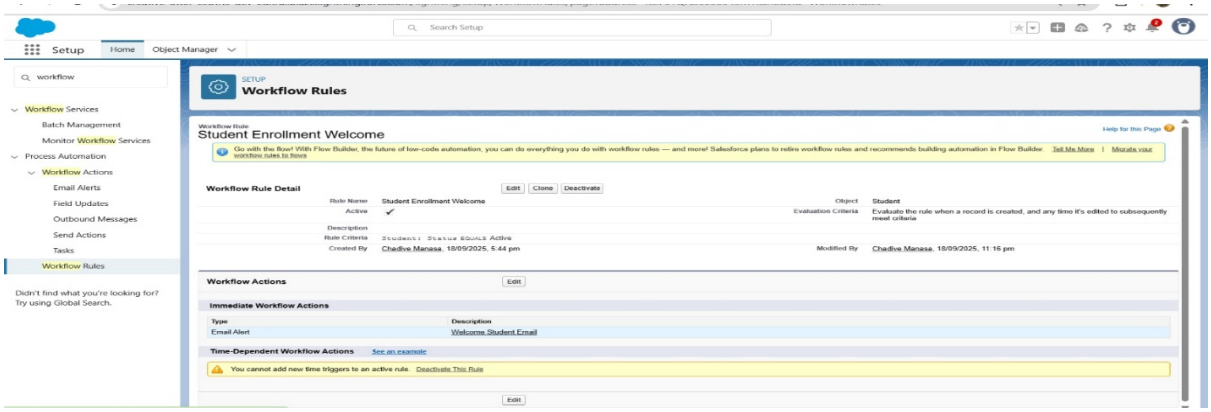
- **Rule:** Ensure Enrollment Year is not in the future.
- **Condition:** Enrollment_Year__c > YEAR(TODAY())
- **Error Message:** "Enrollment Year cannot be greater than the current year."

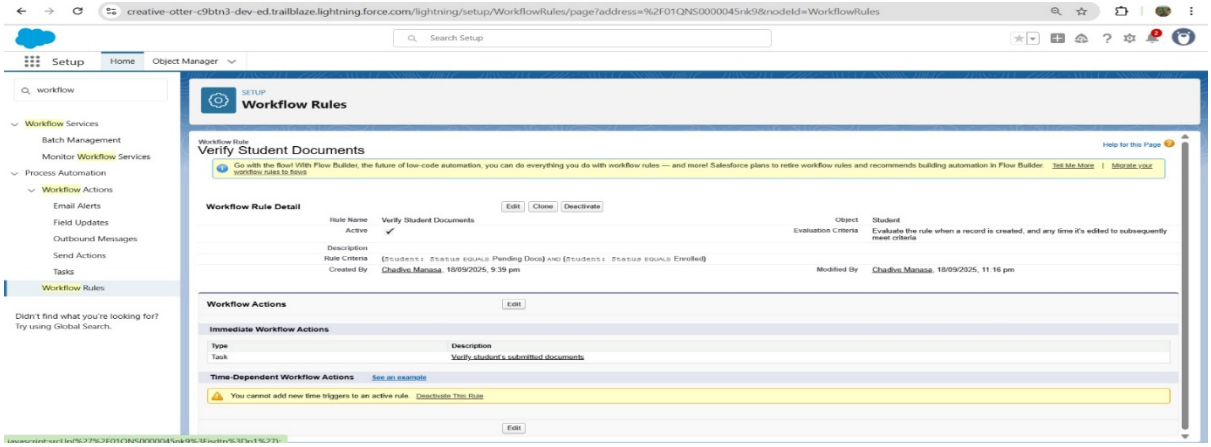


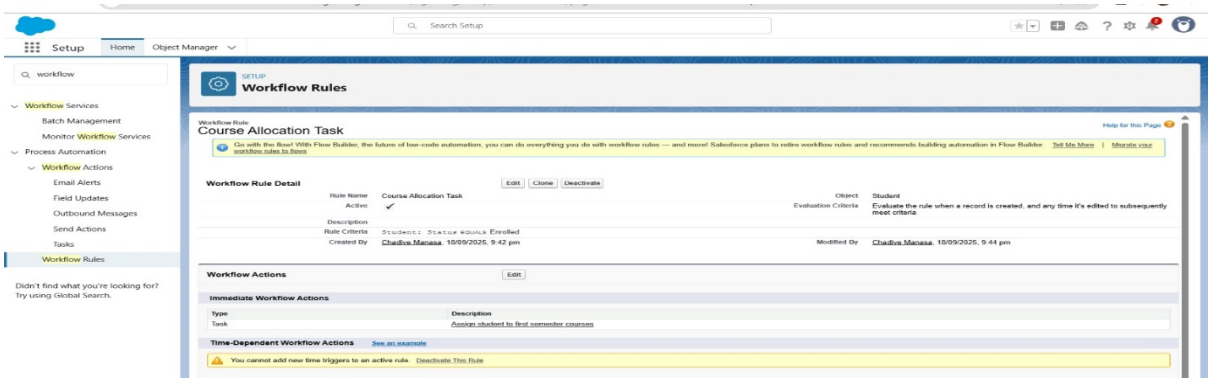
2. Workflow Rules

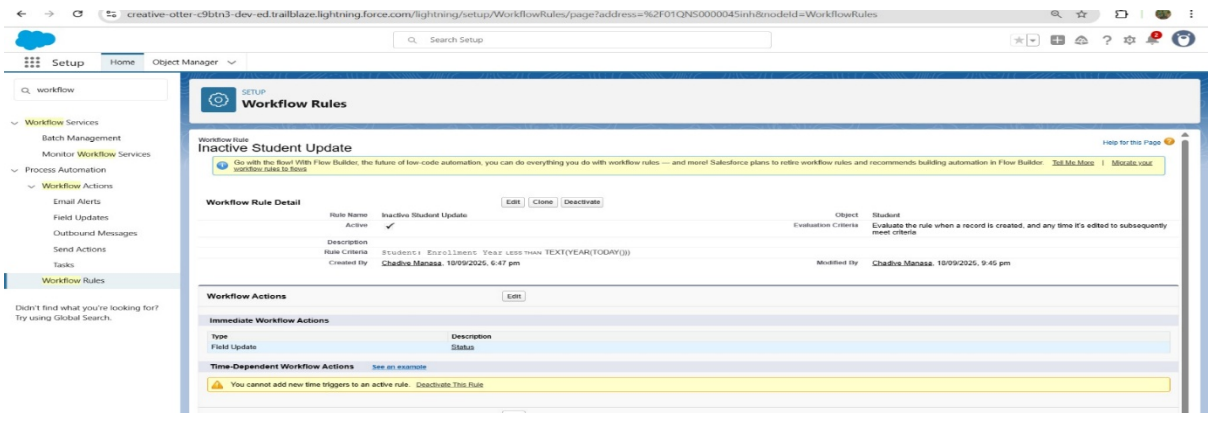
Object: Student__c

- **Rule Name:** Inactive Student Update
- **Criteria:** Enrollment_Year__c < TEXT(YEAR(TODAY()))
- **Action:** Field Update → Status__c = "Inactive"


Setup
Home
Object Manager
Search Setup
workflow
Workflow Services
Batch Management
Monitor Workflow Services
Process Automation
Workflow Actions
Email Alerts
Field Updates
Outbound Messages
Send Actions
Tasks
Workflow Rules
Didn't find what you're looking for?
Try using Global Search.
SETUP
Workflow Rules
Workflow Rule
Student Enrollment Welcome
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
Student Enrollment Welcome
Active
Description
Rule Criteria
Student: Status EQUALS Active
Created By
Chadive Manasa, 18/09/2025, 5:44 pm
Object
Student
Evaluation Criteria
Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
Modified By
Chadive Manasa, 18/09/2025, 11:16 pm
Workflow Actions
Edit
Immediate Workflow Actions
Type
Email Alert
Description
Welcome Student Email
Time-Dependent Workflow Actions
See an example
You cannot add new time triggers to an active rule. Deactivate This Rule
Edit


creative-otter-c9btn3-dev-ed.trailblaze.lightning.force.com/lightning/setup/WorkflowRules/page?address=%2F01QNS0000045nk9&nodeId=WorkflowRules
Setup
Home
Object Manager
Search Setup
workflow
Workflow Services
Batch Management
Monitor Workflow Services
Process Automation
Workflow Actions
Email Alerts
Field Updates
Outbound Messages
Send Actions
Tasks
Workflow Rules
Didn't find what you're looking for?
Try using Global Search.
SETUP
Workflow Rules
Workflow Rule
Verify Student Documents
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
Verify Student Documents
Active
Description
Rule Criteria
(Student: Status EQUALS Pending Docs) AND (Student: Status EQUALS Enrolled)
Created By
Chadive Manasa, 18/09/2025, 9:39 pm
Object
Student
Evaluation Criteria
Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
Modified By
Chadive Manasa, 18/09/2025, 11:16 pm
Workflow Actions
Edit
Immediate Workflow Actions
Type
Task
Description
Verify student's submitted documents
Time-Dependent Workflow Actions
See an example
You cannot add new time triggers to an active rule. Deactivate This Rule
Edit

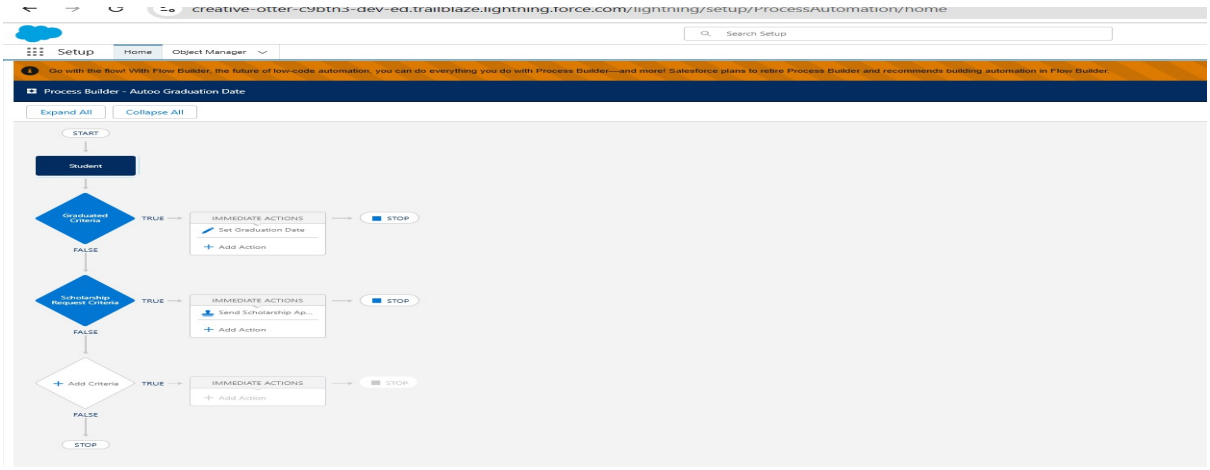

Setup
Home
Object Manager
Search Setup
workflow
Workflow Services
Batch Management
Monitor Workflow Services
Process Automation
Workflow Actions
Email Alerts
Field Updates
Outbound Messages
Send Actions
Tasks
Workflow Rules
Didn't find what you're looking for?
Try using Global Search.
SETUP
Workflow Rules
Workflow Rule
Course Allocation Task
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
Course Allocation Task
Active
Description
Rule Criteria
Student: Status EQUALS Enrolled
Created By
Chadive Manasa, 18/09/2025, 9:42 pm
Object
Student
Evaluation Criteria
Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
Modified By
Chadive Manasa, 18/09/2025, 9:44 pm
Workflow Actions
Edit
Immediate Workflow Actions
Type
Task
Description
Assign student to first semester courses
Time-Dependent Workflow Actions
See an example
You cannot add new time triggers to an active rule. Deactivate This Rule


creative-otter-c9btn3-dev-ed.trailblaze.lightning.force.com/lightning/setup/WorkflowRules/page?address=%2F01QNS0000045inh8&nodeId=WorkflowRules
Setup
Home
Object Manager
Search Setup
workflow
Workflow Services
Batch Management
Monitor Workflow Services
Process Automation
Workflow Actions
Email Alerts
Field Updates
Outbound Messages
Send Actions
Tasks
Workflow Rules
Didn't find what you're looking for?
Try using Global Search.
SETUP
Workflow Rules
Workflow Rule
Inactive Student Update
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
Inactive Student Update
Active
Description
Rule Criteria
Student: Enrollment Year LESS THAN TEXT(YEAR(TODAY()))
Created By
Chadive Manasa, 18/09/2025, 6:47 pm
Object
Student
Evaluation Criteria
Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
Modified By
Chadive Manasa, 18/09/2025, 9:45 pm
Workflow Actions
Edit
Immediate Workflow Actions
Type
Field Update
Description
Status
Time-Dependent Workflow Actions
See an example
You cannot add new time triggers to an active rule. Deactivate This Rule

3. Process Builder

Object: Student__c

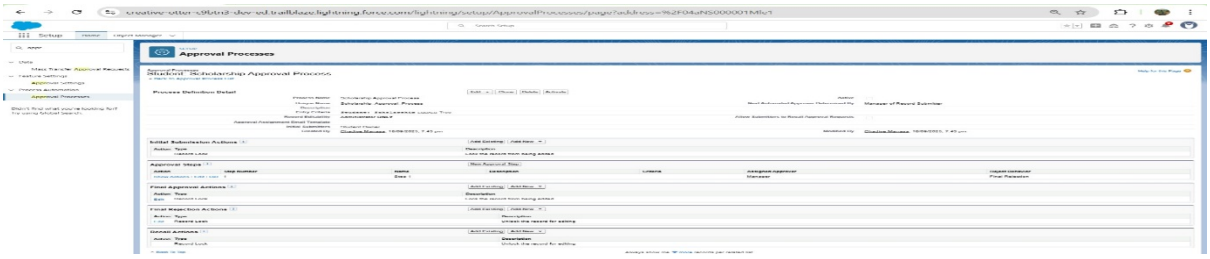
- **Process Name:** Graduation Update
- **Criteria:** When Status__c = "Graduated"
- **Action 1:** Update Field → Graduation_Date__c = TODAY()
- **Action 2:** Send Email Alert → "Graduation Congratulations Email" to student.



4. Approval Process

Object: Student__c

- **Process Name:** Scholarship Approval
- **Entry Criteria:** Scholarship_Requested__c = TRUE
- **Steps:**
    1. Initial Submission → Lock record.
    2. Assigned To → Finance Staff Queue.
    3. Final Approval → Unlock record + Update Scholarship_Status__c = "Approved".
    4. Rejection → Update Scholarship_Status__c = "Rejected".



5. Flow Builder

a) Screen Flow: Student Registration

- **Inputs:** First Name, Last Name, Email, Enrollment Year, Status.
- **Action:** Create Student record in Student__c.

b) Record-Triggered Flow: Auto Assign Status

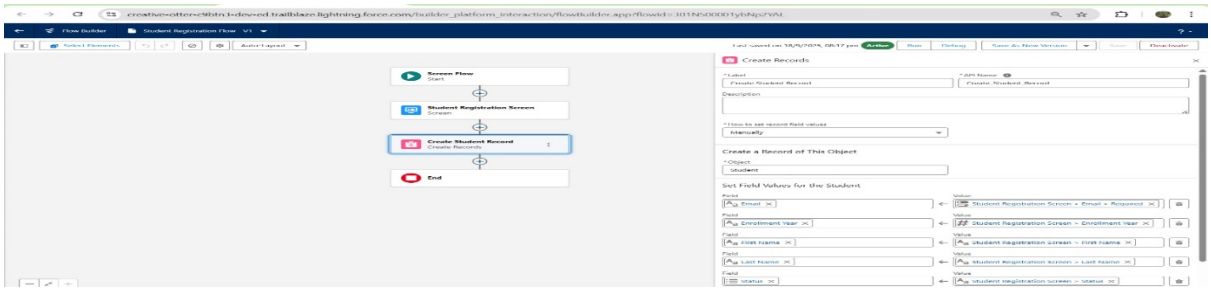
- **When:** A new Student record is created.
- **Condition:** If Enrollment_Year__c = TEXT(YEAR(TODAY()))
- **Action:** Update Status__c = "Enrolled".

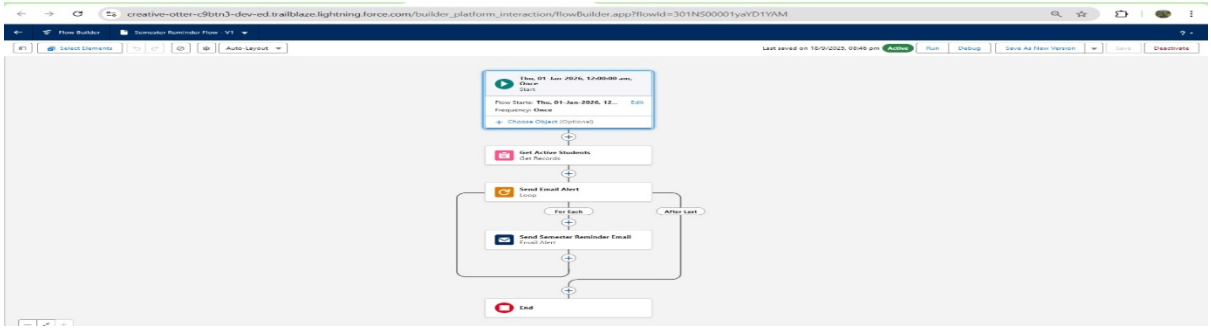
c) Scheduled Flow: Semester Reminder

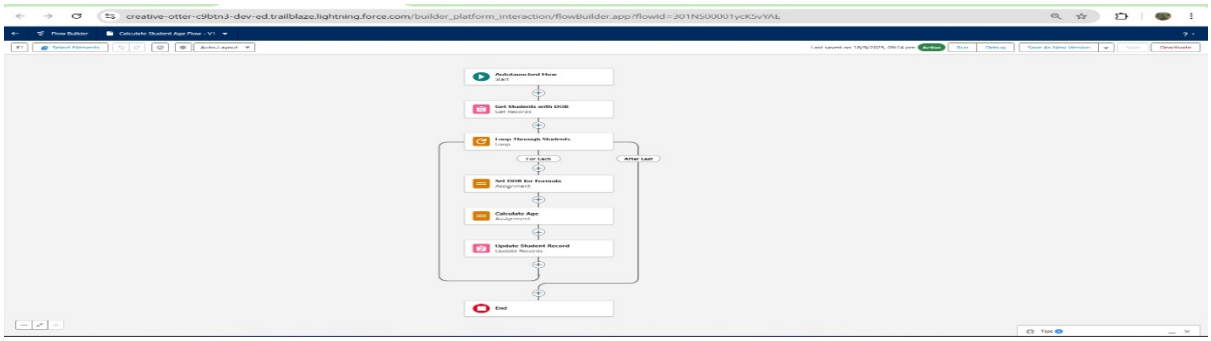
- **When:** Every semester start date (Scheduled path).
- **Action:** Send Email Alert to all Student__c records with Status__c = "Enrolled".

d) Auto-Launched Flow: Calculate Student Age

- **Formula:** TODAY() - Date_of_Birth__c
- **Action:** Update Age__c field in Student__c.



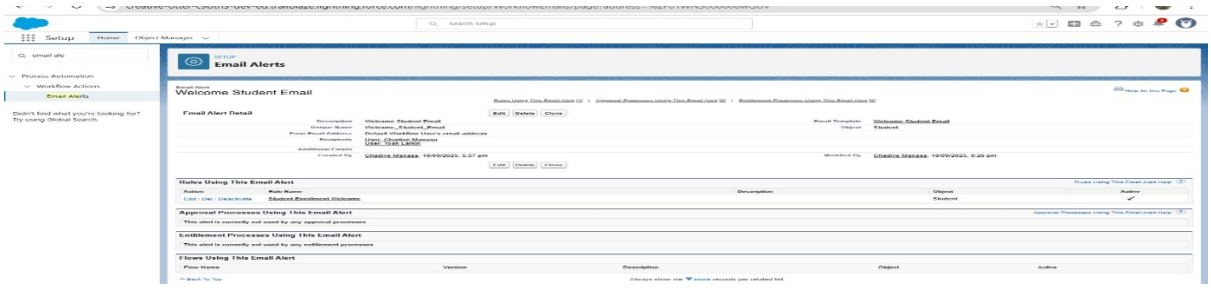


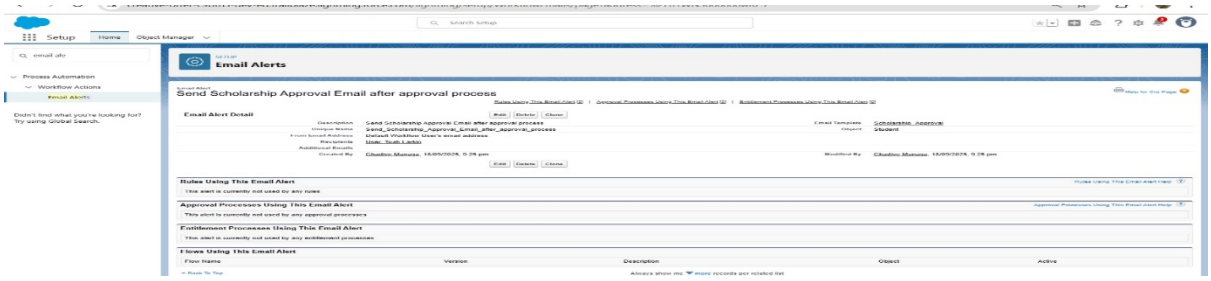


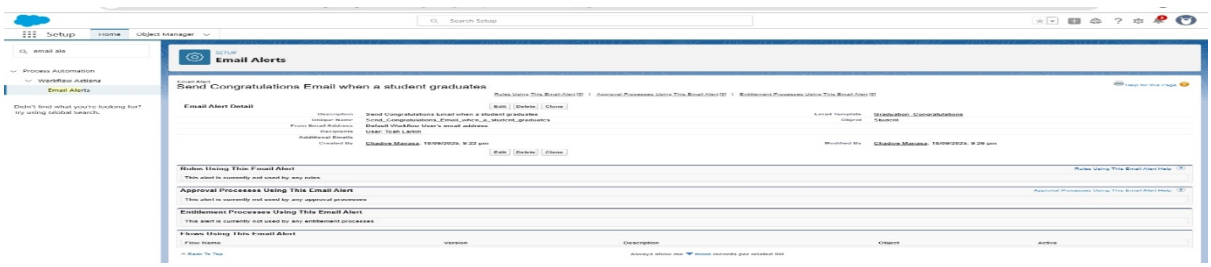
6. Email Alerts

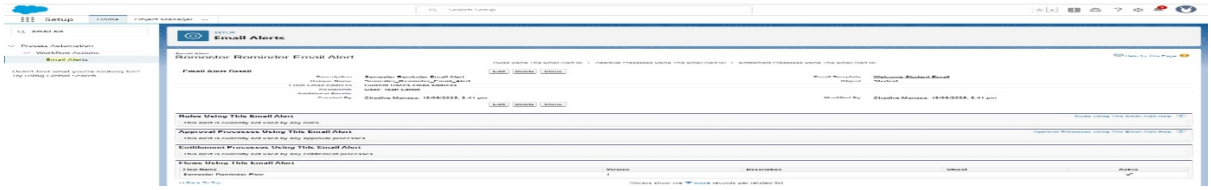
Object: Student__c

- **Templates Created:**
    1. Welcome Student Email (after registration).
    2. Semester Reminder Email.
    3. Graduation Congratulations Email.









7. Field Updates

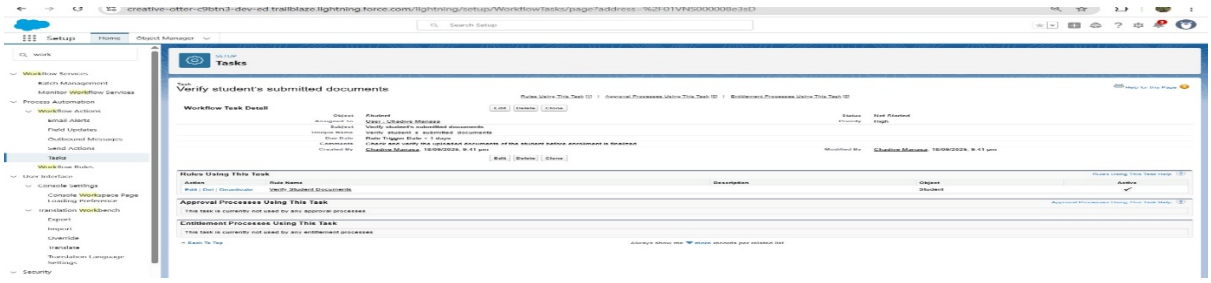
Objects & Fields:

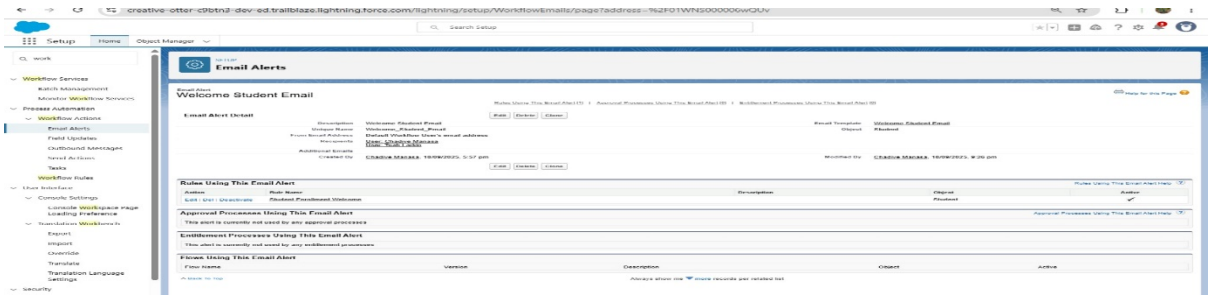
- Student__c.Status__c → "Inactive" (Workflow Rule).
- Student__c.Graduation_Date__c → TODAY() (Process Builder).
- Student__c.Age__c → Formula result (Flow).

8. Tasks

Object: Student__c

- **Example:** When Scholarship request is approved, create a Task for Finance Staff: "Process Scholarship Payment."

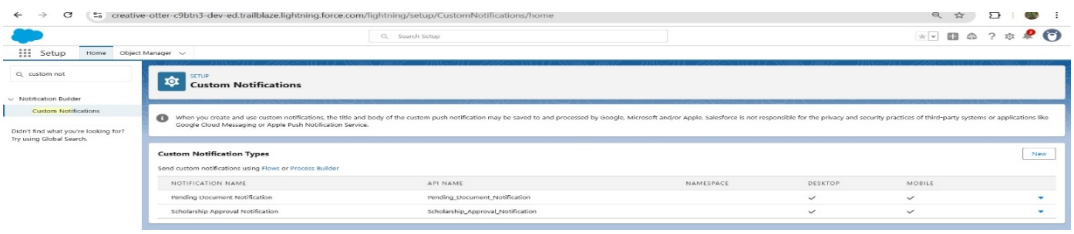




9. Custom Notifications

Object: Student__c

- **Example:** When a student is marked as "Graduated", send a **Custom Notification** to the Advisor/Staff to initiate alumni registration.



Phase 5: Apex Programming (Developer)

Goal of this Phase

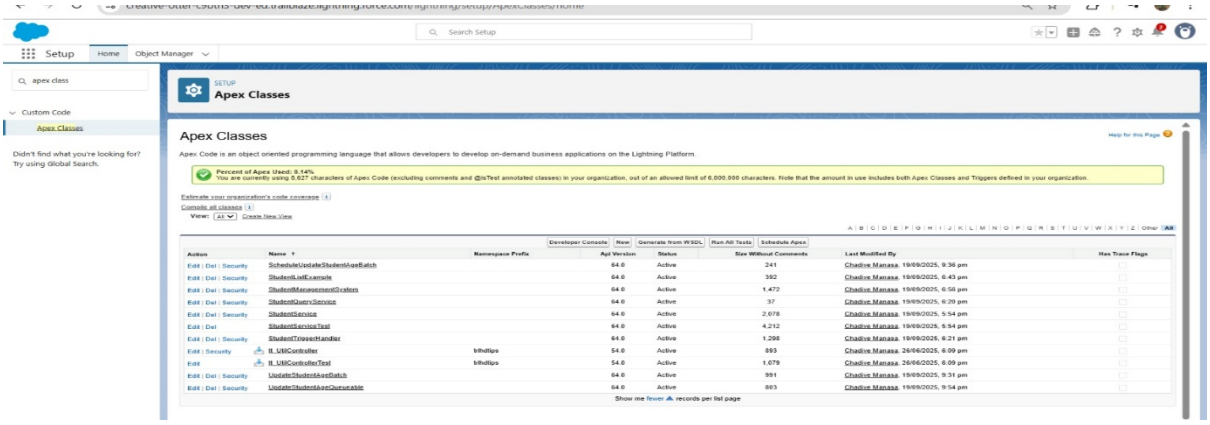
The goal of Phase 5 was to implement **Apex programming features** in Salesforce to support the Student Management System. This included creating **classes, triggers, SOQL/SOSL queries, collections, asynchronous processing, exception handling, and test classes** to ensure the application is efficient, scalable, and reliable.

1. Classes & Objects

- Created **StudentService** Apex class to handle business logic.
- Added methods for calculating student age based on Date of Birth and updating records.

2. Apex Triggers (Before/After Insert/Update/Delete)

- Implemented a **StudentTrigger** to:
    - Calculate Age before insert/update when Date of Birth is set.
    - Prevent invalid updates using before update triggers.
    - Maintain consistency when student records are deleted.

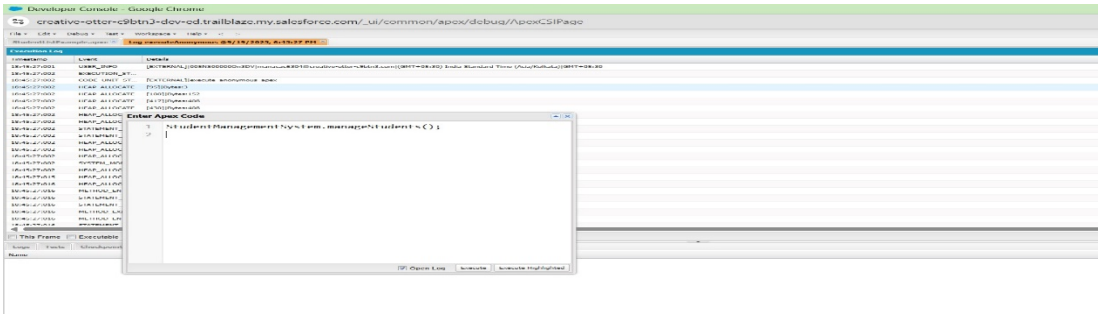


3. Trigger Design Pattern

- Applied **Handler Class Pattern**:
    - Trigger delegates logic to StudentTriggerHandler class.
    - Ensures clean separation of logic, reusable code, and easier maintenance.

4. SOQL & SOSL

- Used **SOQL queries** to fetch Student records with fields like Id, Date_of_Birth__c, and Age__c.
- Demonstrated **SOSL** for searching students by name across multiple fields.

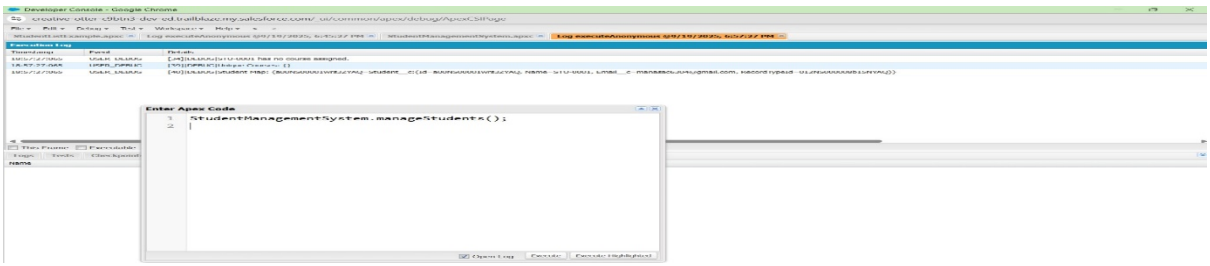


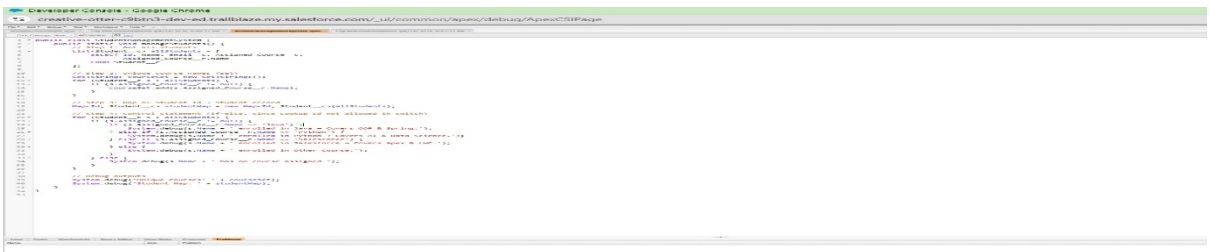
5. Collections (List, Set, Map)

- Used **List<Student__c>** for batch updates.
- Applied **Set<Id>** to handle unique record IDs.
- Implemented **Map<Id, Student__c>** to efficiently compare old and new trigger contexts.

6. Control Statements

- Used **If-Else conditions** for null checks.
- Applied **For loops** to process multiple student records.
- Incorporated **Try-Catch blocks** for exception handling.



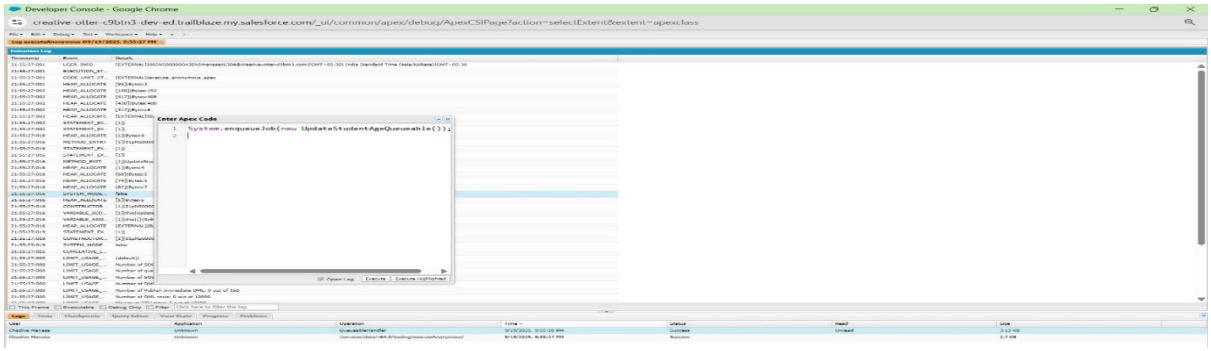


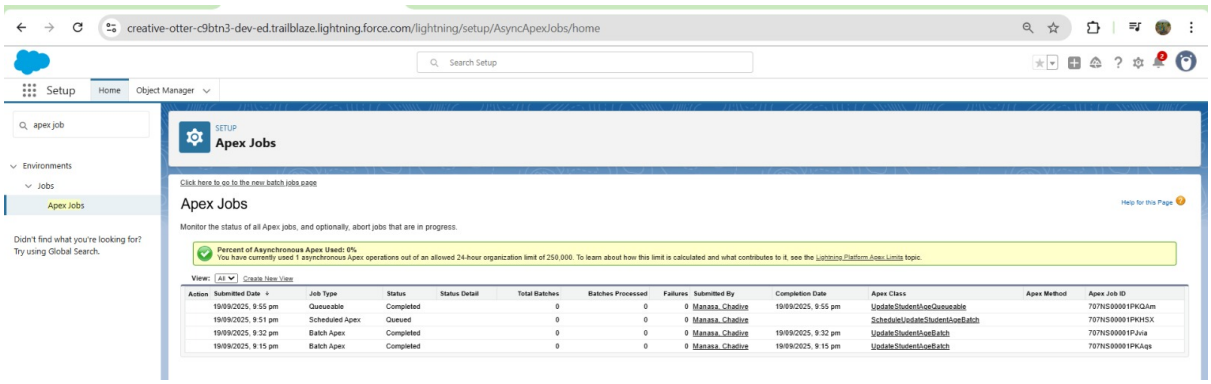
7. Batch Apex

- Created **UpdateStudentAgeBatch** class implementing Database.Batchable<SObject>.
- Processes students in batches to calculate and update Age.
- Scheduled and monitored using **Apex Jobs**.

8. Queueable Apex

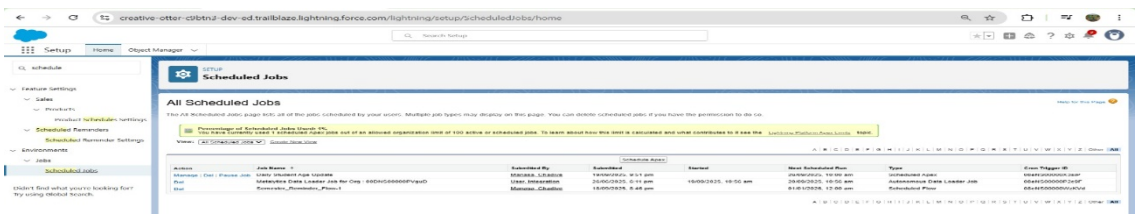
- Developed **UpdateStudentAgeQueueable** class implementing Queueable.
- Allows background execution of student age updates.
- Can be chained for sequential execution.





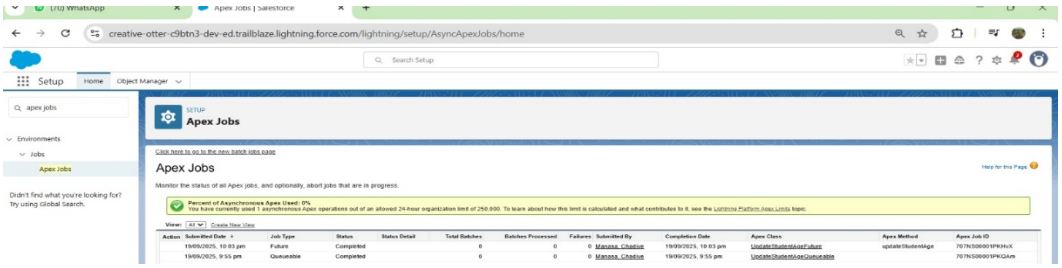
9. Scheduled Apex

- Created **ScheduleUpdateStudentAgeBatch** class implementing Schedulable.
- Scheduled batch jobs to run at predefined times.



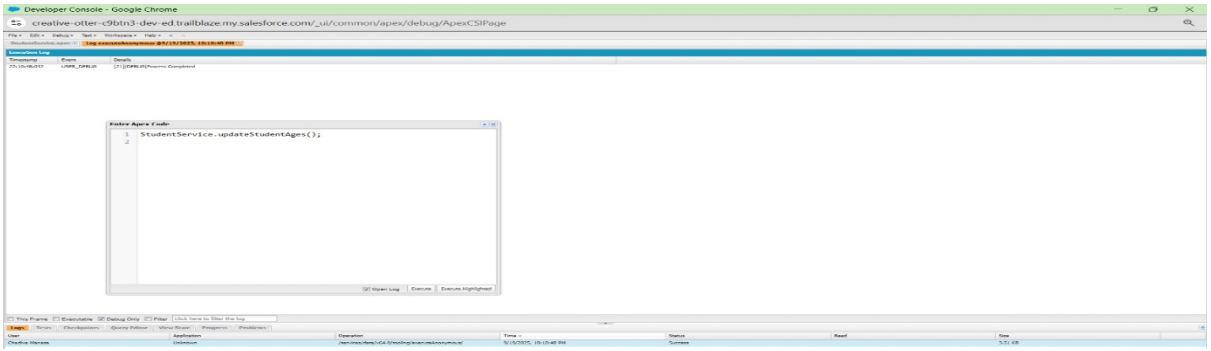
10. Future Methods

- Implemented **UpdateStudentAgeFuture** class with @future annotation.
- Runs background updates asynchronously for lightweight processing.



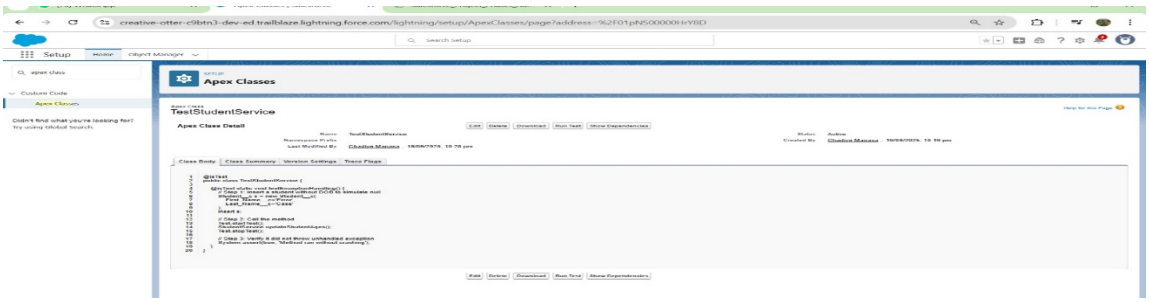
11. Exception Handling

- Wrapped logic in **try-catch** blocks.
- Handled null Date of Birth scenario to prevent runtime errors.
- Ensured no unhandled exceptions crash the system.



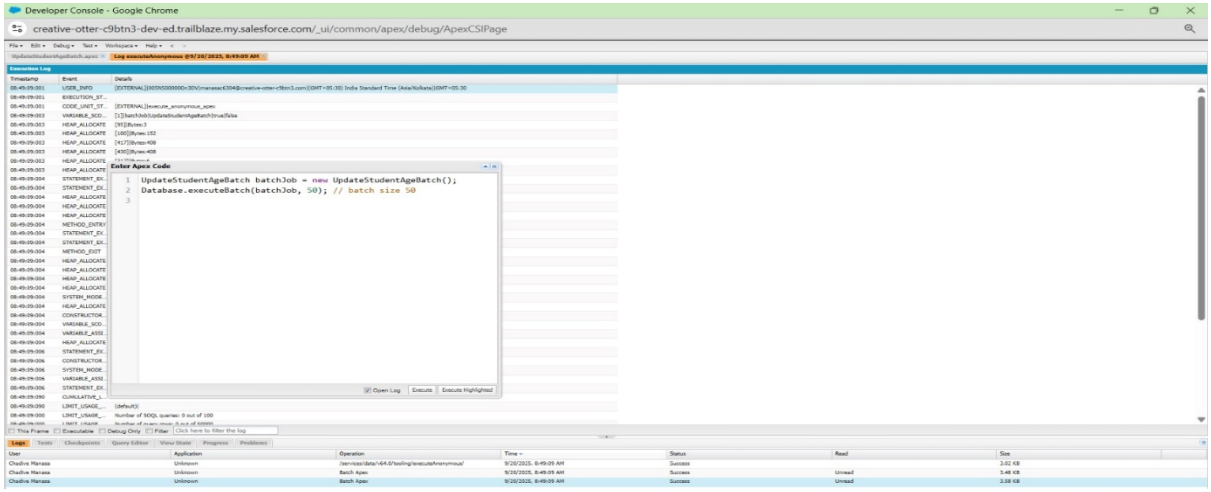
12. Test Classes

- Created **TestStudentService** and other test classes with @isTest.
- Verified Batch, Queueable, Future, and Exception Handling methods.
- Achieved **code coverage** and validated system reliability.



13. Asynchronous Processing

- Implemented and tested all four types: **Batch, Queueable, Scheduled, and Future Methods**.
- Verified job execution via **Apex Jobs** in Setup.
- Ensured scalability for handling large volumes of Student records.

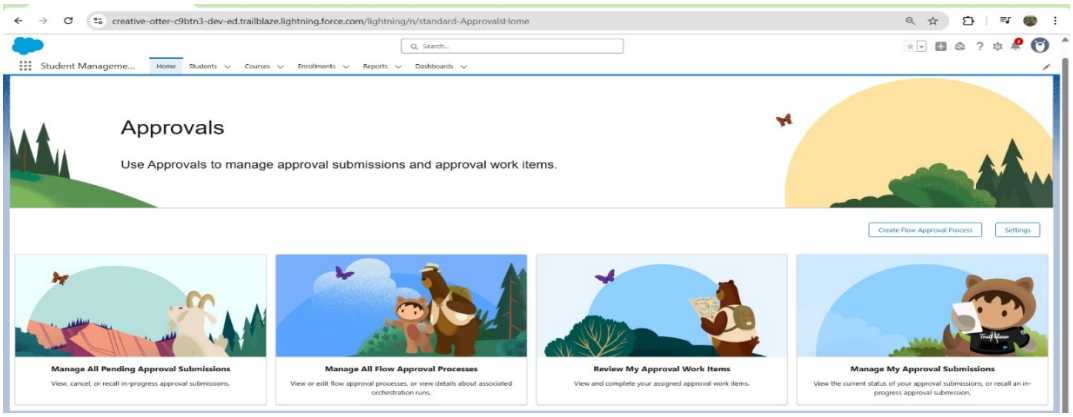


Phase 6: User Interface Development

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.

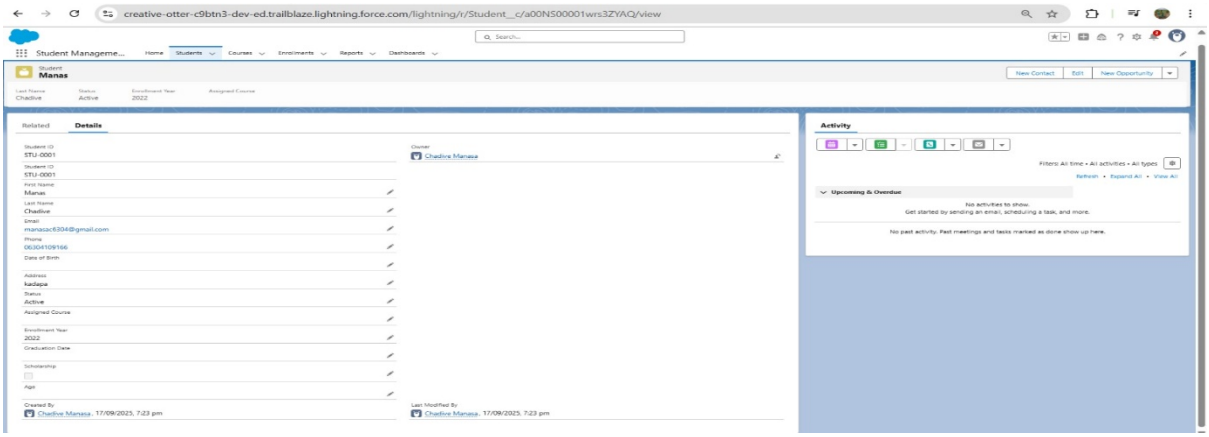
1.Lightning App Builder

- **Purpose:** Drag-and-drop tool to create custom pages without coding.
- **Key Uses:** Build **App Pages**, **Home Pages**, and **Record Pages**.
- **Details:** Place standard components or custom LWCs, save, and activate for users.



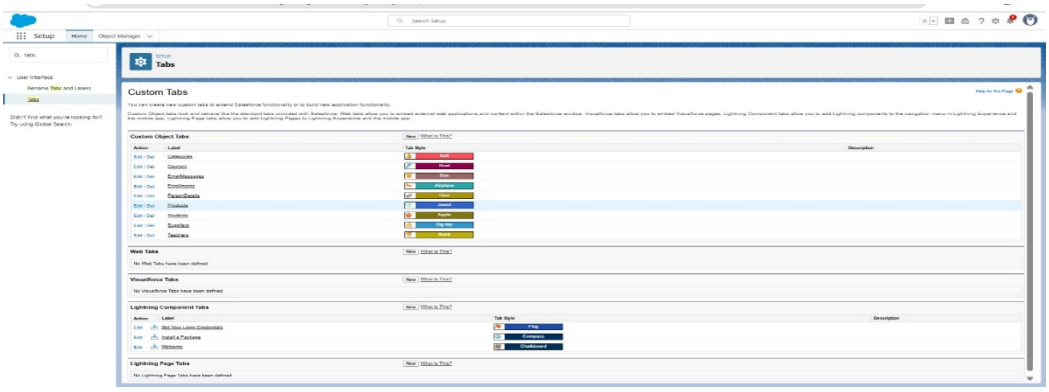
2.Record Pages

- **Purpose:** Customize the layout of individual records.
- **Key Uses:** Display key fields, related lists, and custom LWCs for objects like Student or Course.
- **Details:** Assign pages to apps or profiles to control visibility.



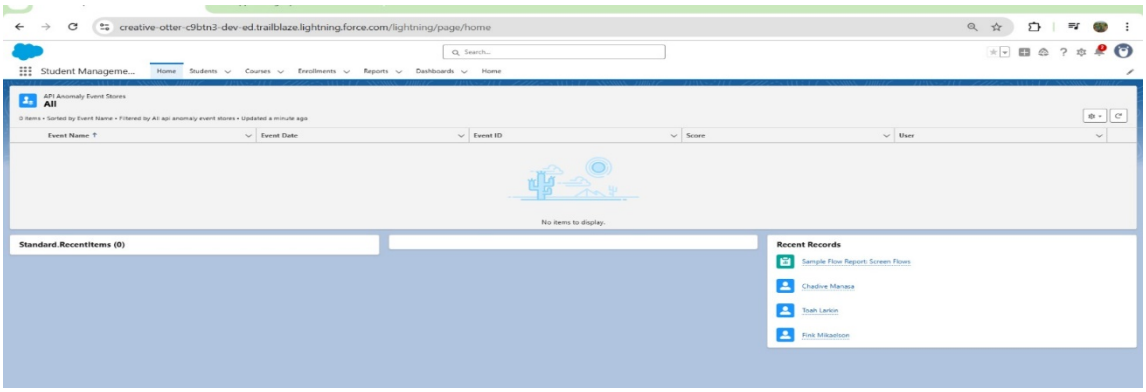
3.Tabs

- **Purpose:** Organize objects, pages, or components into tabs for easy navigation.
- **Key Uses:** Access **Students, Courses, Reports**, or custom pages quickly.
- **Details:** Tabs can host objects, Visualforce pages, web pages, or LWCs.



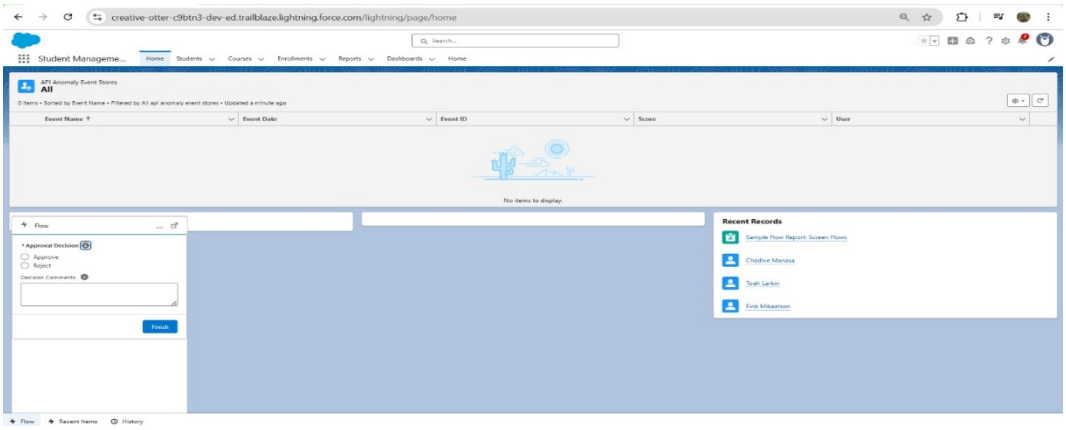
4. Home Page Layouts

- **Purpose:** Customize the Home Page for better user experience.
- **Key Uses:** Show dashboards, reports, navigation buttons, and custom LWCs.
- **Details:** Layouts define the structure and visibility of components on the Home Page.



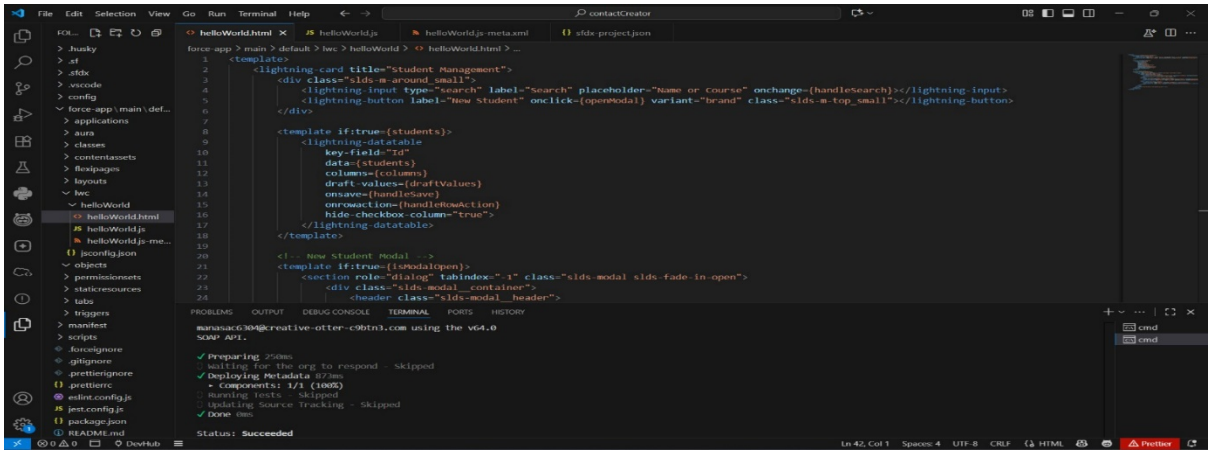
5. Utility Bar

- **Purpose:** Quick access to frequently used tools at the bottom of the app.
- **Key Uses:** Add Notes, Messages, or custom LWCs for easy reach.
- **Details:** Can configure behavior, size, and component order.



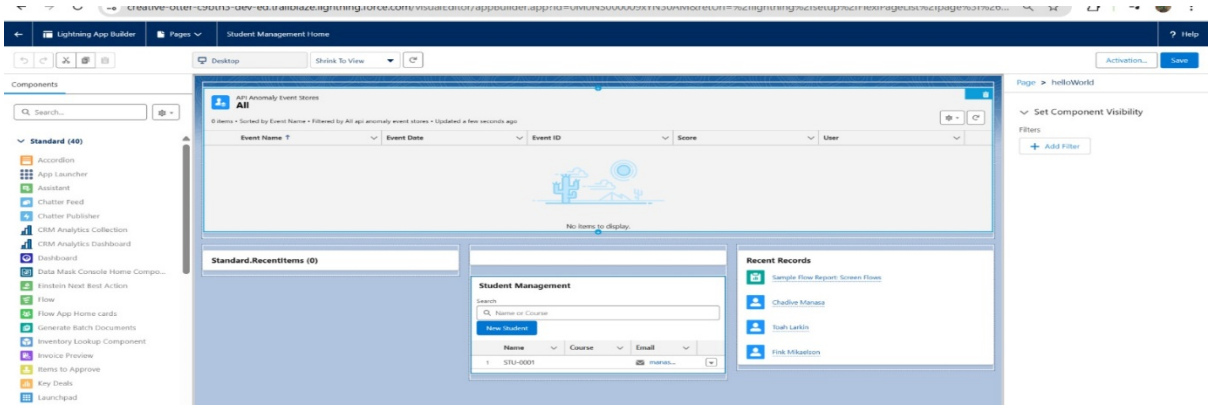
6. Lightning Web Components (LWC)

- **Purpose:** Build reusable, interactive components for the UI.
- **Key Uses:** Display student lists, details, assignments, certificates, and navigation buttons.
- **Details:** Components consist of HTML (template), JS (functionality), CSS (styling), and meta file (configuration).



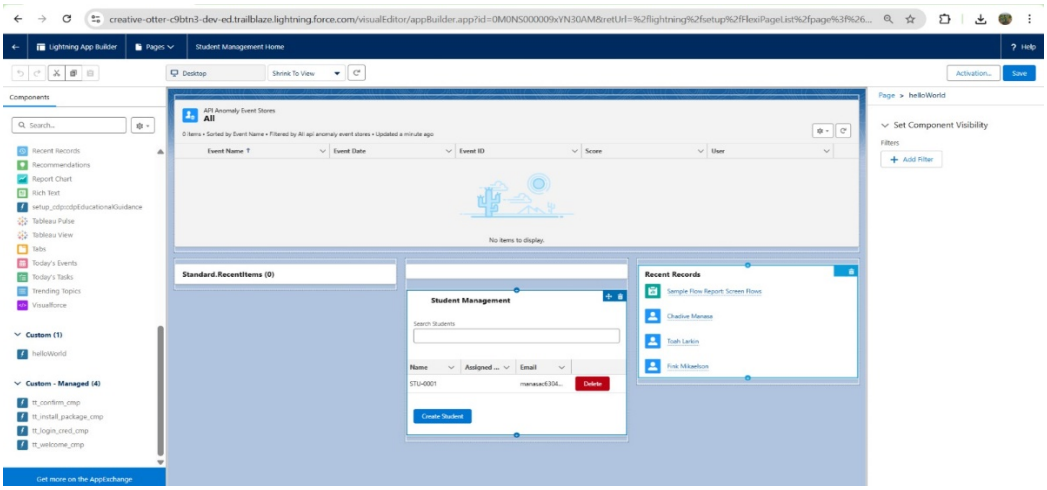
7. Apex with LWC

- **Purpose:** Fetch or manipulate Salesforce data using Apex in LWCs.
- **Key Uses:** Display dynamic data like students or courses.
- **Details:** Use **@AuraEnabled methods** in Apex, then call them in JS (Wire or Imperative).



8. Events in LWC

- **Purpose:** Enable communication between components.
- **Key Uses:** Pass data from child to parent, or across sibling components.
- **Details:** Use **Custom Events** (dispatchEvent) or **Lightning Message Service** for cross-component communication.



9. Wire Adapters

- **Purpose:** Automatically fetch Salesforce data reactively.
- **Key Uses:** Display records or lists without manual refresh.
- **Details:** Example: @wire(getRecord, { recordId: '$recordId', fields }) record;

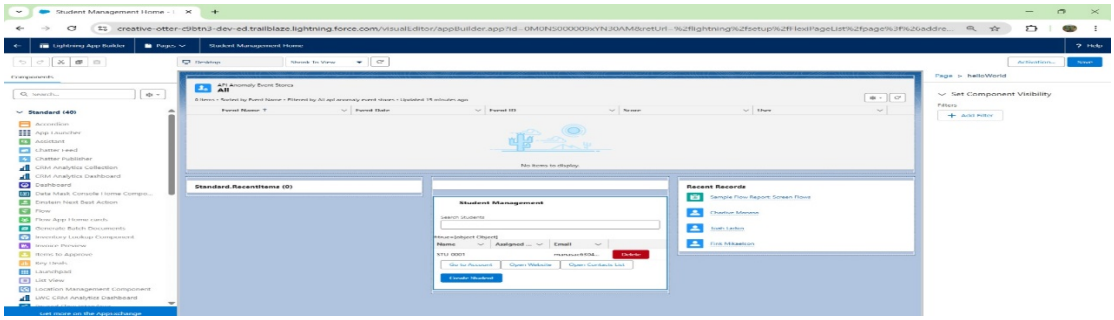
10. Imperative Apex Calls

- **Purpose:** Call Apex methods manually from JS when needed.
- **Key Uses:** Fetch or update data on button click or action.

- **Details:** Example:

```
import getStudents from '@salesforce/apex/StudentController.getStudents';


handleLoadStudents() {
    getStudents()
        .then(result => { this.students = result; })
        .catch(error => { this.error = error; });
}
```



11. Navigation Service

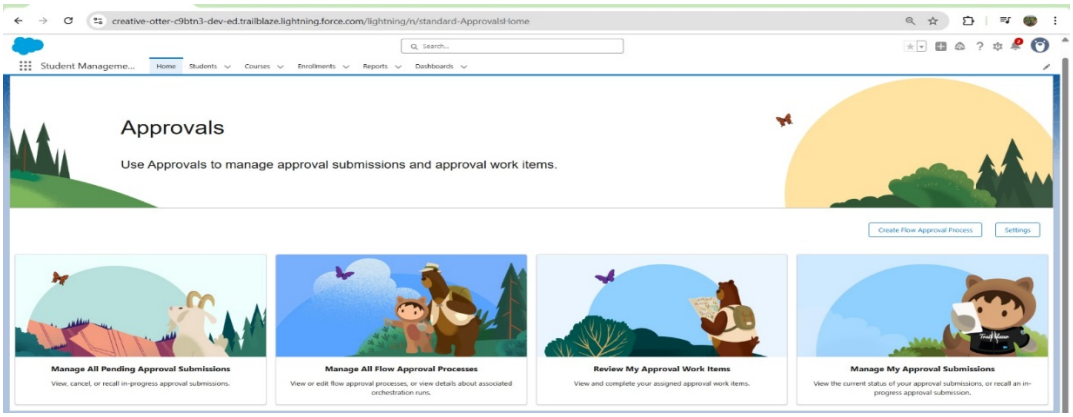
- **Purpose:** Programmatically navigate to records, lists, or external URLs.
- **Key Uses:** Navigate users to **Student details, Course list, or external websites**.
- **Details:** Use NavigationMixin.Navigate() inside LWC methods. Works on **App Pages and Record Pages**, but not standard Home Pages.

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.

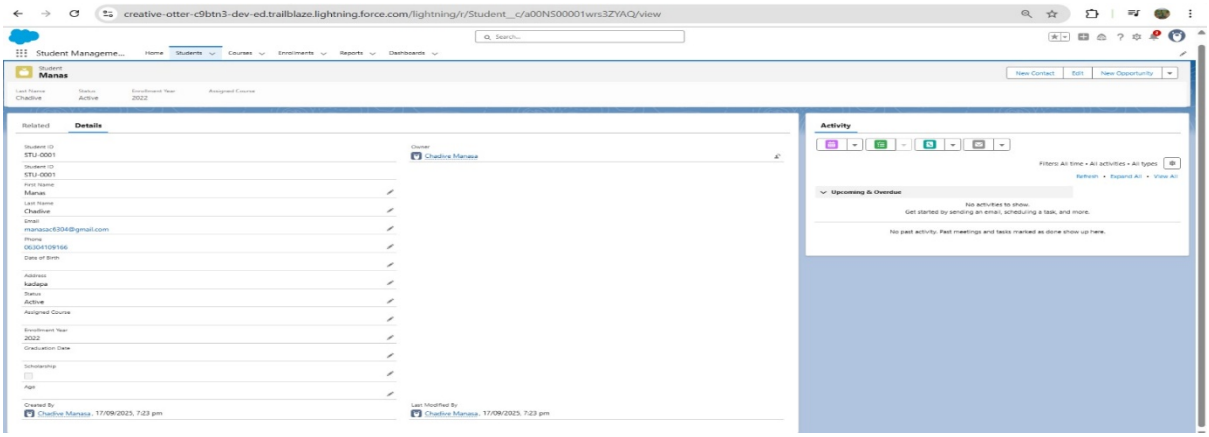
1.Lightning App Builder

- **Purpose:** Drag-and-drop tool to create custom pages without coding.
- **Key Uses:** Build **App Pages**, **Home Pages**, and **Record Pages**.
- **Details:** Place standard components or custom LWCs, save, and activate for users.



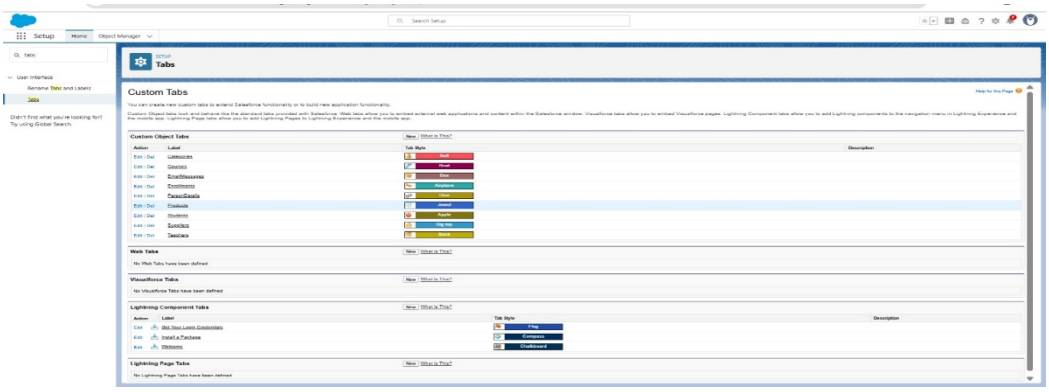
2.Record Pages

- **Purpose:** Customize the layout of individual records.
- **Key Uses:** Display key fields, related lists, and custom LWCs for objects like Student or Course.
- **Details:** Assign pages to apps or profiles to control visibility.



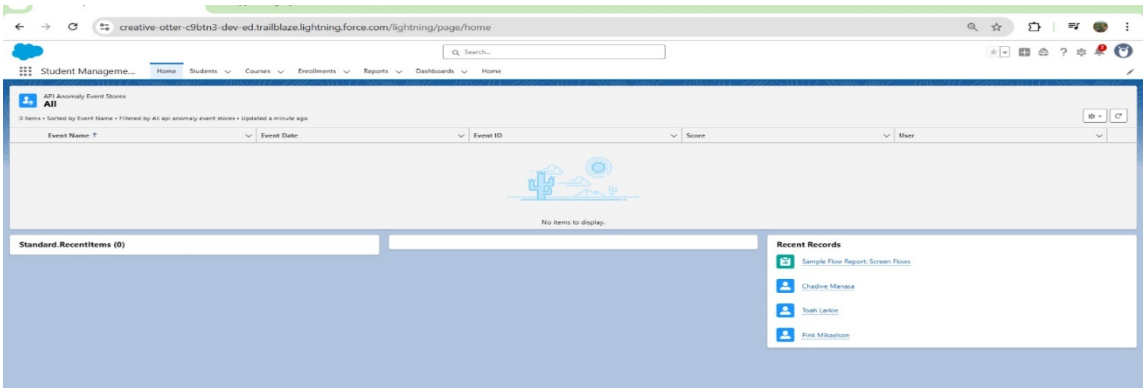
3.Tabs

- **Purpose:** Organize objects, pages, or components into tabs for easy navigation.
- **Key Uses:** Access **Students, Courses, Reports**, or custom pages quickly.
- **Details:** Tabs can host objects, Visualforce pages, web pages, or LWCs.



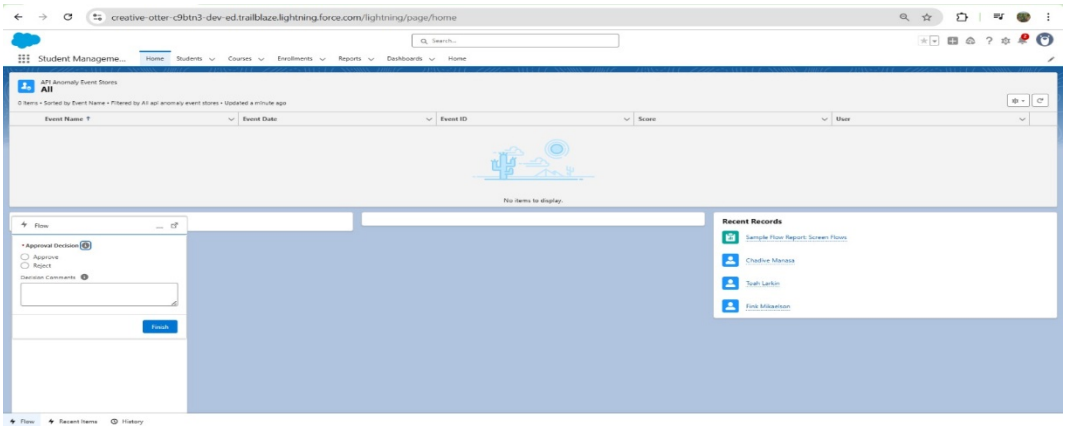
4. Home Page Layouts

- **Purpose:** Customize the Home Page for better user experience.
- **Key Uses:** Show dashboards, reports, navigation buttons, and custom LWCs.
- **Details:** Layouts define the structure and visibility of components on the Home Page.



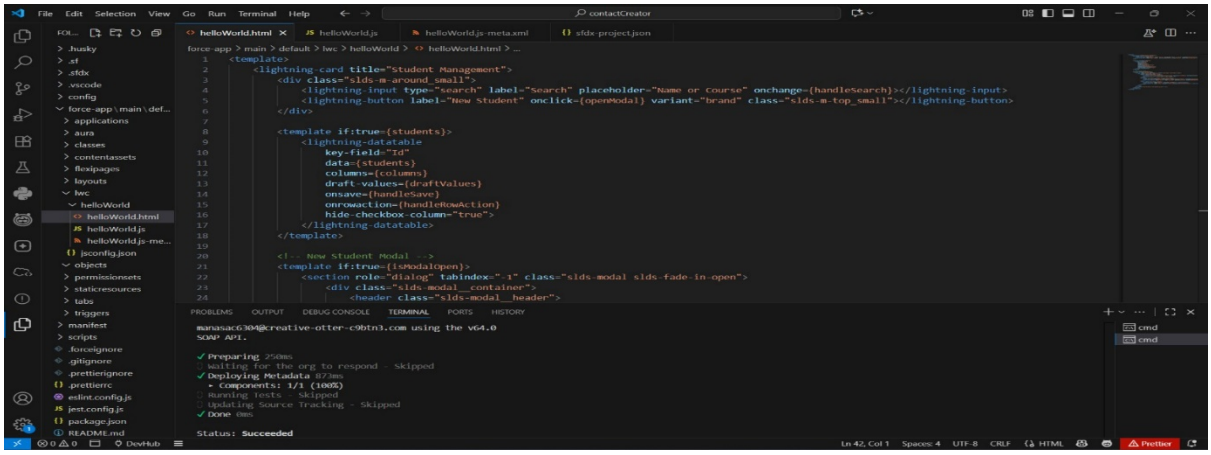
5. Utility Bar

- **Purpose:** Quick access to frequently used tools at the bottom of the app.
- **Key Uses:** Add Notes, Messages, or custom LWCs for easy reach.
- **Details:** Can configure behavior, size, and component order.



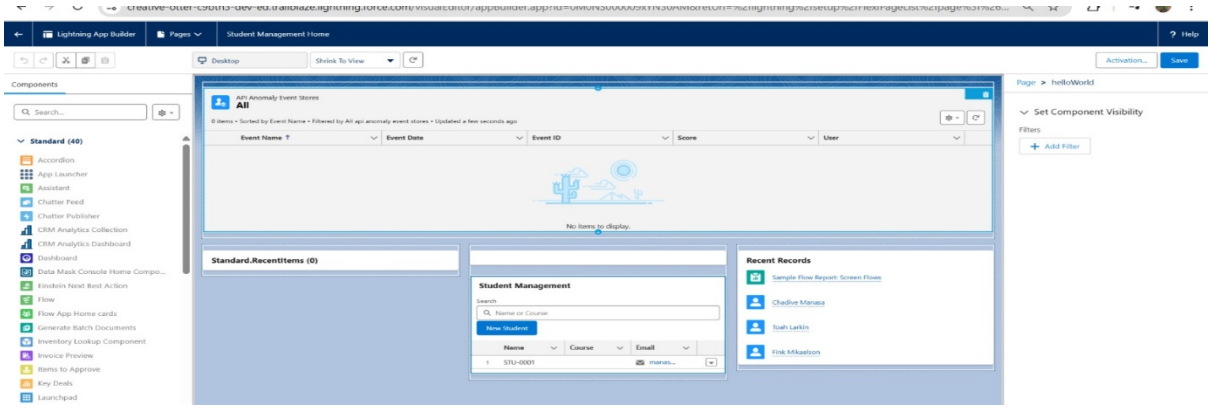
6. Lightning Web Components (LWC)

- **Purpose:** Build reusable, interactive components for the UI.
- **Key Uses:** Display student lists, details, assignments, certificates, and navigation buttons.
- **Details:** Components consist of HTML (template), JS (functionality), CSS (styling), and meta file (configuration).



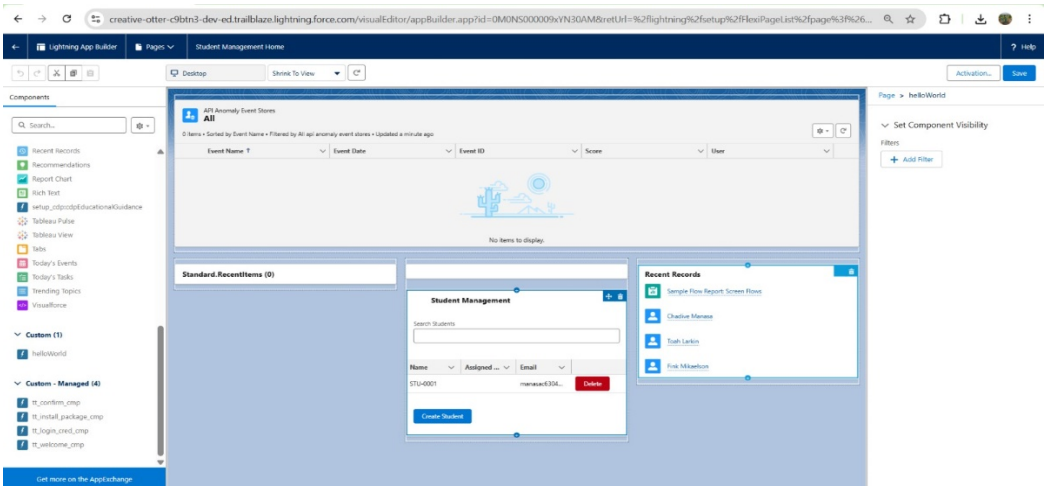
7. Apex with LWC

- **Purpose:** Fetch or manipulate Salesforce data using Apex in LWCs.
- **Key Uses:** Display dynamic data like students or courses.
- **Details:** Use **@AuraEnabled methods** in Apex, then call them in JS (Wire or Imperative).



8. Events in LWC

- **Purpose:** Enable communication between components.
- **Key Uses:** Pass data from child to parent, or across sibling components.
- **Details:** Use **Custom Events** (dispatchEvent) or **Lightning Message Service** for cross-component communication.



9. Wire Adapters

- **Purpose:** Automatically fetch Salesforce data reactively.
- **Key Uses:** Display records or lists without manual refresh.
- **Details:** Example: @wire(getRecord, { recordId: '$recordId', fields }) record;

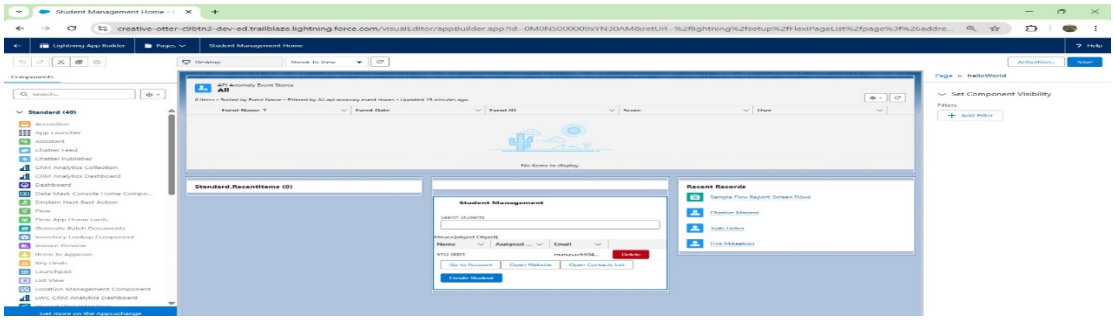
10. Imperative Apex Calls

- **Purpose:** Call Apex methods manually from JS when needed.
- **Key Uses:** Fetch or update data on button click or action.

- **Details:** Example:

```
import getStudents from '@salesforce/apex/StudentController.getStudents';


handleLoadStudents() {
    getStudents()
        .then(result => { this.students = result; })
        .catch(error => { this.error = error; });
}
```



11. Navigation Service

- **Purpose:** Programmatically navigate to records, lists, or external URLs.
- **Key Uses:** Navigate users to **Student details, Course list, or external websites**.
- **Details:** Use NavigationMixin.Navigate() inside LWC methods. Works on **App Pages and Record Pages**, but not standard Home Pages.

Phase 7: Integration & External Access – Student Management System

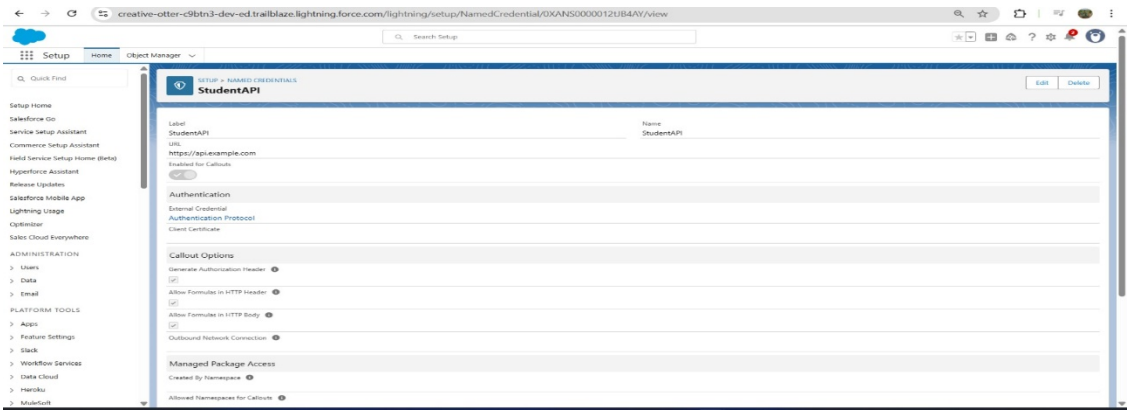
Goal of This Phase

The primary goal of this phase was to enable **seamless data exchange and interaction** between Salesforce and external systems.
This ensures that the Student Management System can:

- Synchronize student data with other platforms (ERP, Learning Systems, etc.)
- Trigger real-time updates when changes happen inside Salesforce.
- Provide secure and reliable API endpoints for external applications.
- Enforce best practices for authentication, callouts, and integration limits.

1. Named Credentials

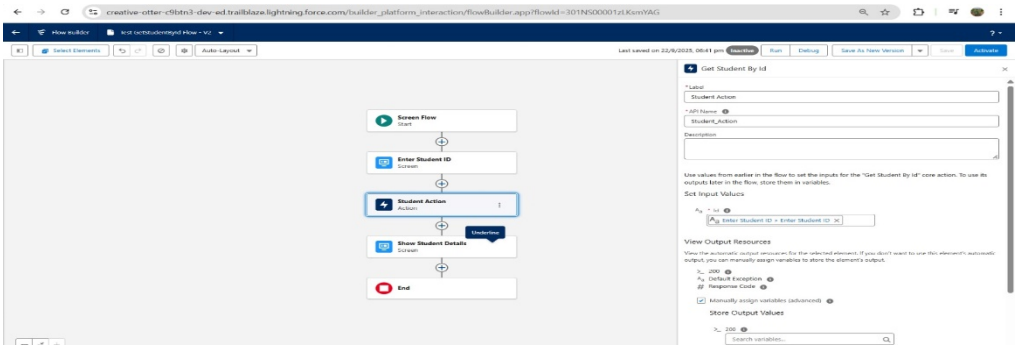
- **Created Named Credential:** Student_API_Named_Credential
- Configured with:
    - Base URL of the external Student API.
    - Authentication type (if required – basic or OAuth).
- Purpose: Simplified authentication and callout management without exposing credentials in Apex code.



2. External Services

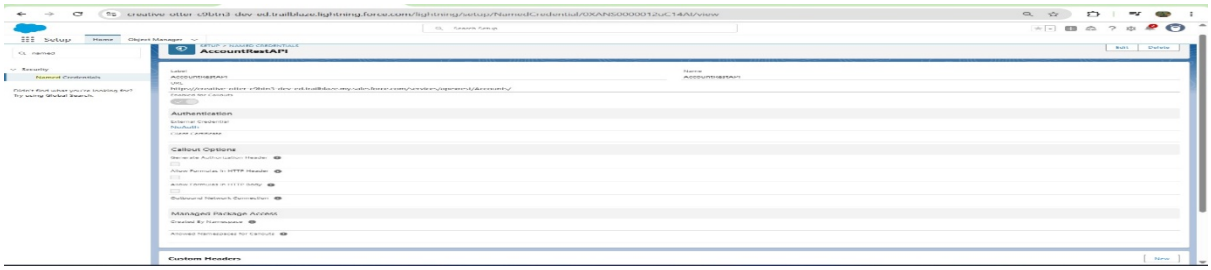
- **Registered External Schema:** Connected to the external Student REST API using the schema definition.
- Auto-generated Apex classes for:
    - Fetching student details.
    - Creating or updating student records externally.

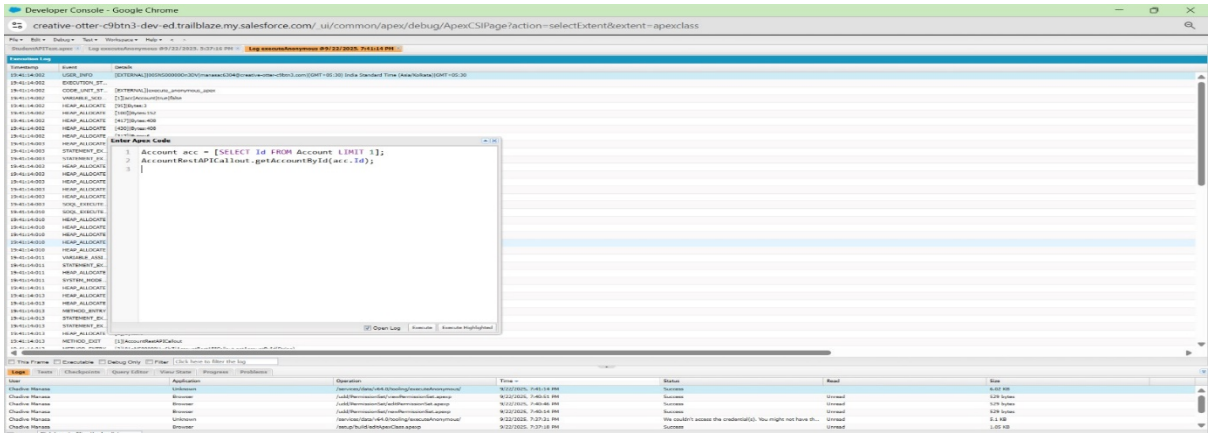
- Purpose: Allow declarative integration (Flow & Apex) without manually writing callout code.



3. Web Services (REST/SOAP)

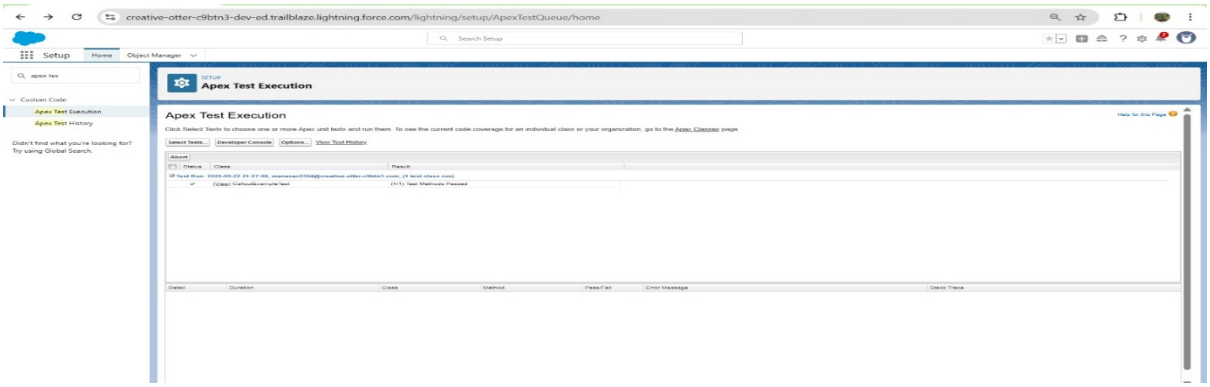
- **Developed REST Apex Class:** StudentRESTService
- Exposed endpoints:
  - /students → Create or update student records.
  - /students/{id} → Fetch student details by Student_ID__c.
- Added proper authentication and exception handling.
- Purpose: Allow external systems to push and pull student data securely.





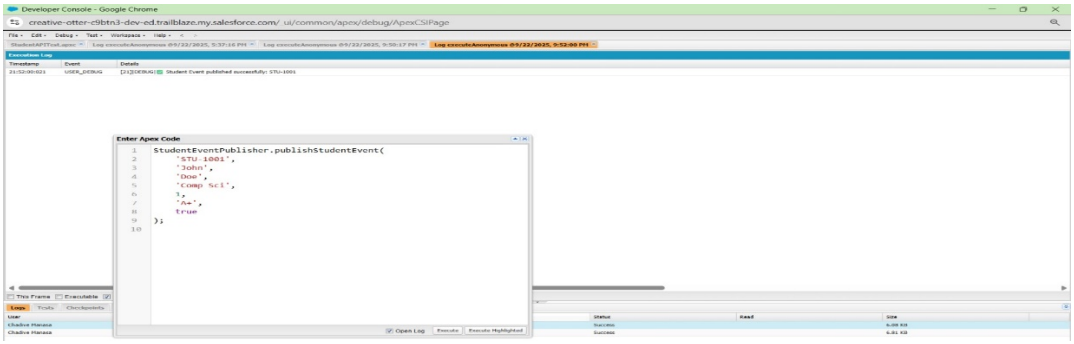
4. Callouts

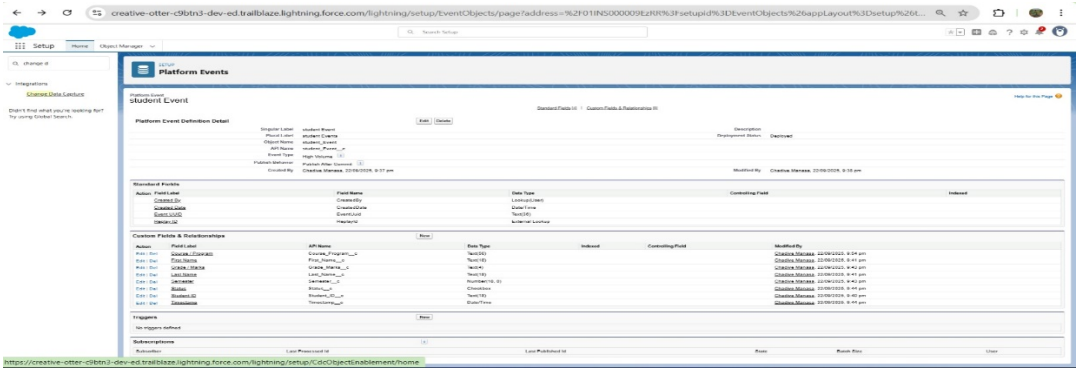
- **Created Apex Callout Class:** CalloutExample
- Added test class with HttpCalloutMock to enable test coverage.
- Verified callout works with Remote Site Settings and Named Credentials.
- Purpose: Send student updates from Salesforce to external systems when a record is created or updated.



5. Platform Events

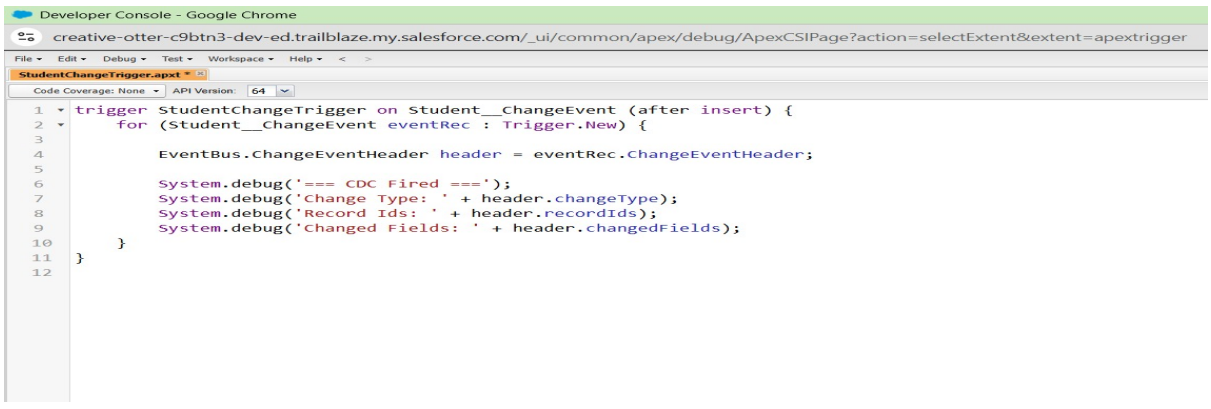
- **Created Platform Event:** Student_Event__e
- Custom Fields:
  - Student_ID__c (Text)
  - First_Name__c (Text)
  - Last_Name__c (Text)
  - Course_Program__c (Text)
  - Grade_Marks__c (Text)
  - Semester__c (Number)
  - Status__c (Checkbox)
  - Timestamp__c (Date/Time)
- Purpose: Publish student data changes to subscribers (internal or external) in real time.

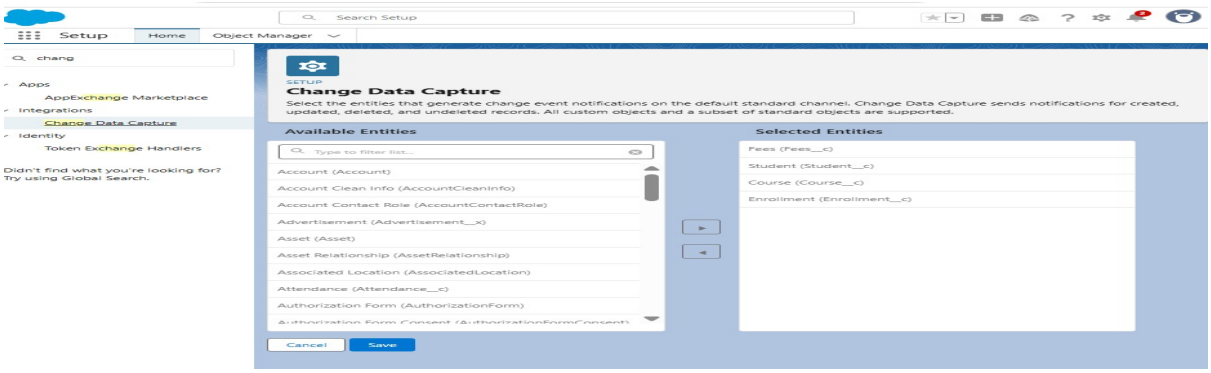




6. Change Data Capture (CDC)

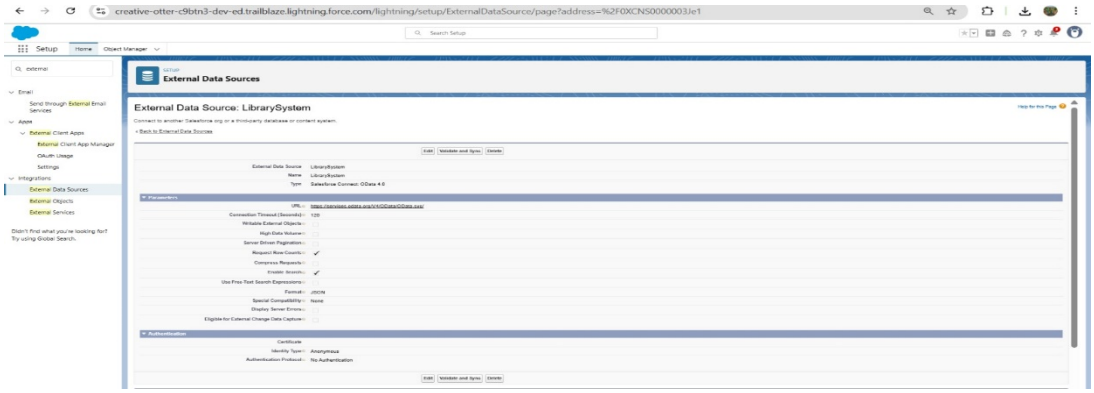
- **Enabled CDC for Student__c object**
- **Created Apex Trigger:** StudentChangeTrigger
  - Logs change type, changed fields, and commit timestamp.
  - Takes action when Grade_Marks__c is updated.
- Purpose: Monitor and respond to changes in real time without writing complex logic.





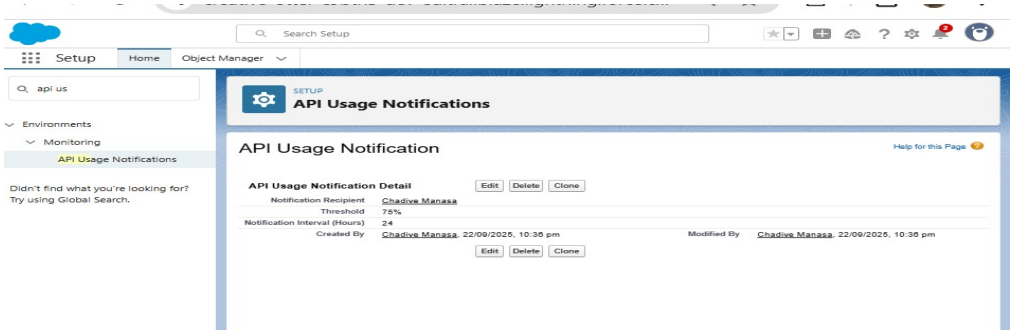
7. Salesforce Connect

- Configured **External Object** (Student_External__x) to display external student data within Salesforce.
- Verified CRUD operations (read-only or writable depending on requirement).
- Purpose: Provide a unified view of external student records directly in Salesforce UI.



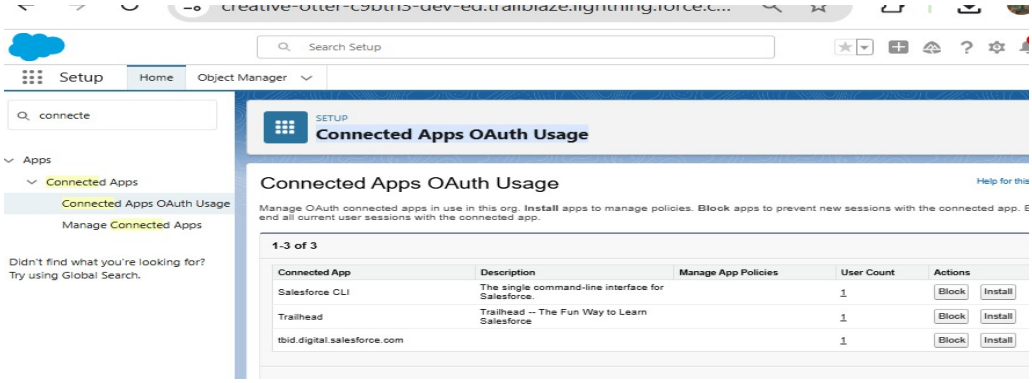
8. API Limits

- Enabled **API Usage Notifications** to alert when API usage reaches 75% and 90%.
- Monitored API calls through **System Overview** and **Reports**.
- Purpose: Prevent API over-consumption and avoid service interruptions.



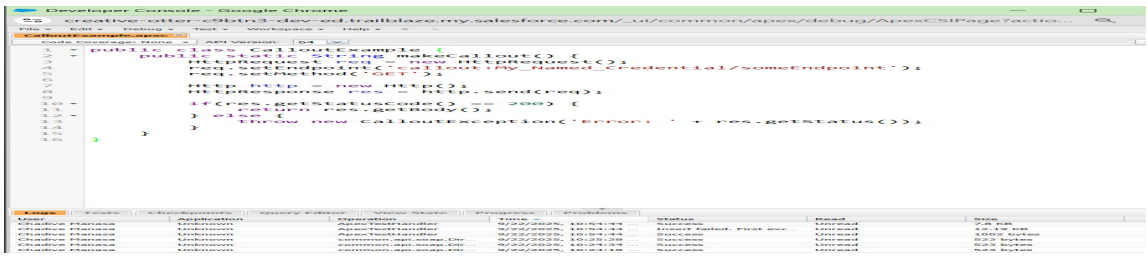
9. OAuth & Authentication

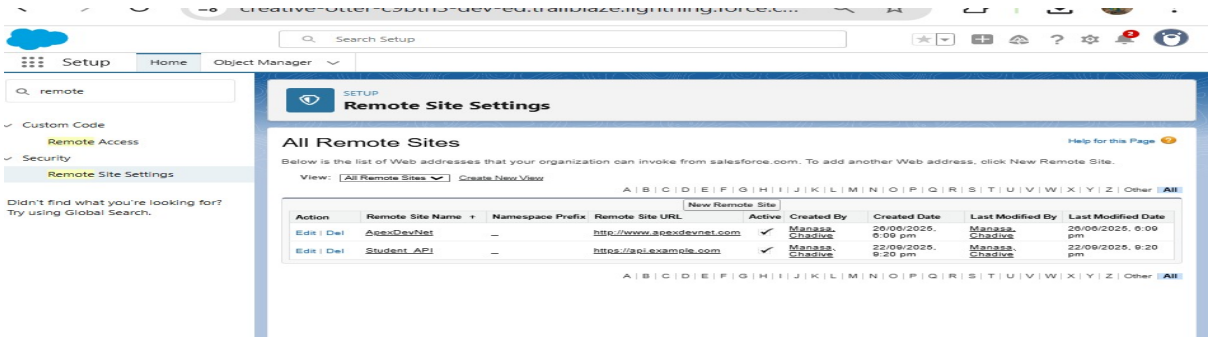
- **Created Connected App:** StudentManagementIntegration
- Configured OAuth scopes:
    - Full Access (full)
    - Perform requests on your behalf at any time (refresh_token, offline_access)
- Assigned profiles/permission sets.
- Tested authentication using Postman and verified in **Connected Apps OAuth Usage**.
- Purpose: Securely allow external applications to authenticate and integrate.



10. Remote Site Settings

- **Created Remote Site:** Student_API_Remote_Site
- Added base URL of external API to allow callouts.
- Purpose: Ensure Salesforce can make HTTPS callouts to the external Student API.



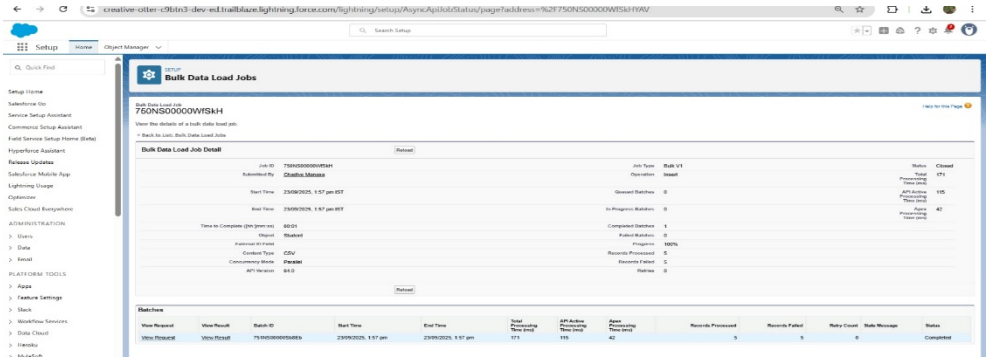


Phase 8: Data Management & Deployment

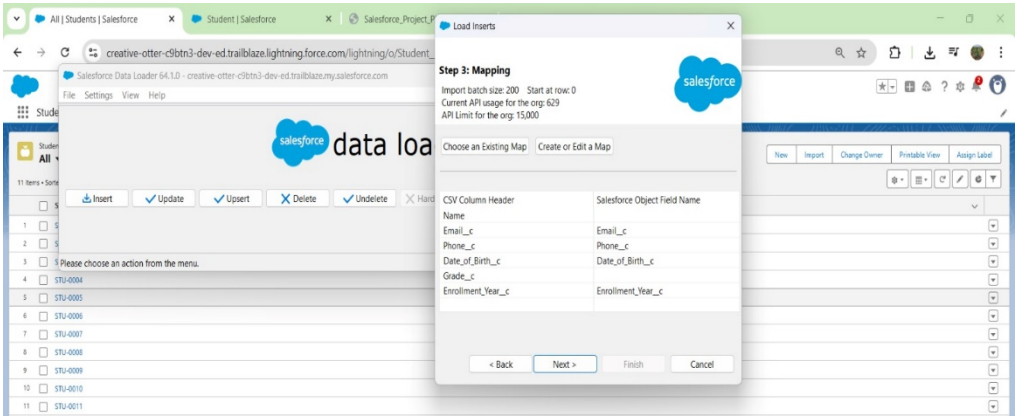
Goal :

To efficiently manage, migrate, and secure student data and metadata across Salesforce environments using native tools, packages, and modern deployment methods.

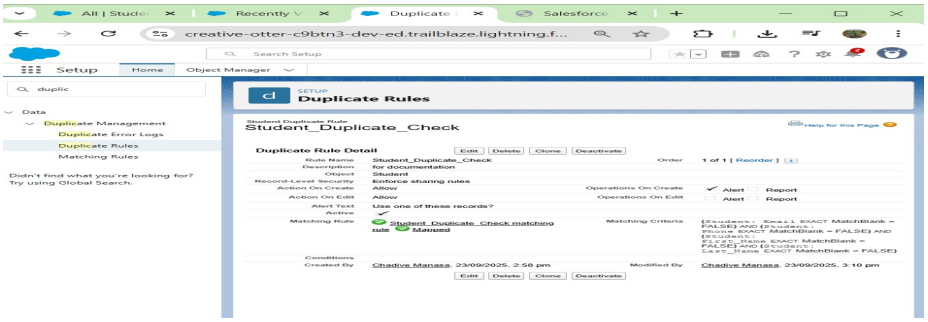
1. **Data Import Wizard**
    - Imported sample student records and related data through the Salesforce UI.
    - Validated successful import with list views and reports.
    - Documented mapping fields for reusability.

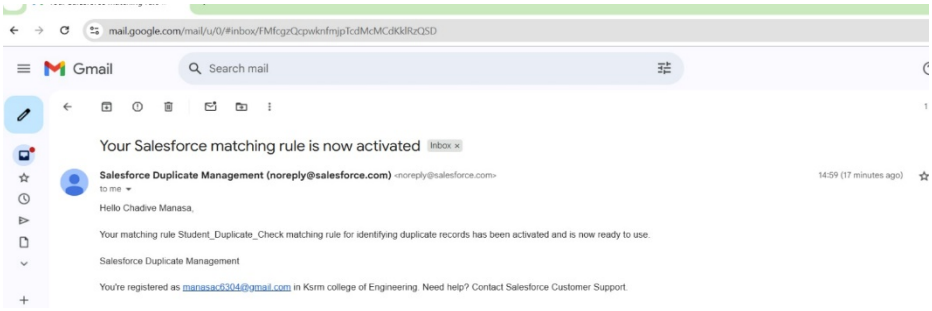


2. **Data Loader**
    - Performed bulk inserts, updates, and deletes for large datasets.
    - Created CSV templates for Student and Course objects.
    - Logged and reviewed success/error CSVs for audit.

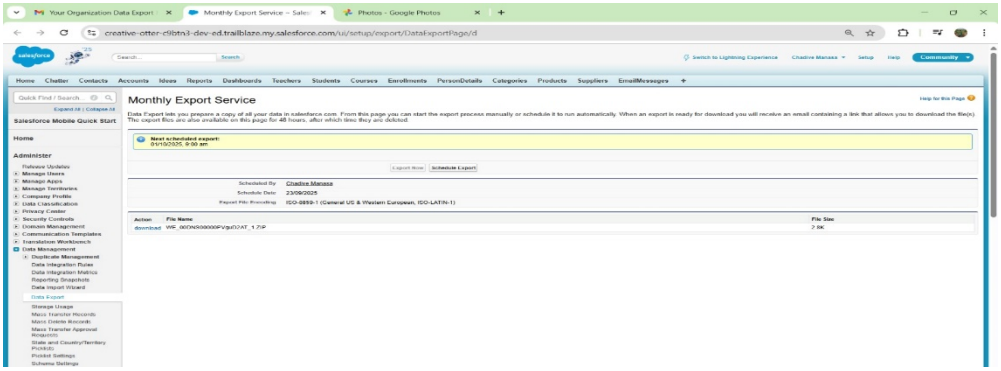


3. **Duplicate Rules**
    - Defined matching rules to detect duplicate student records.
    - Configured duplicate rules to block or allow with alerts.
    - Tested with sample data to ensure data quality.



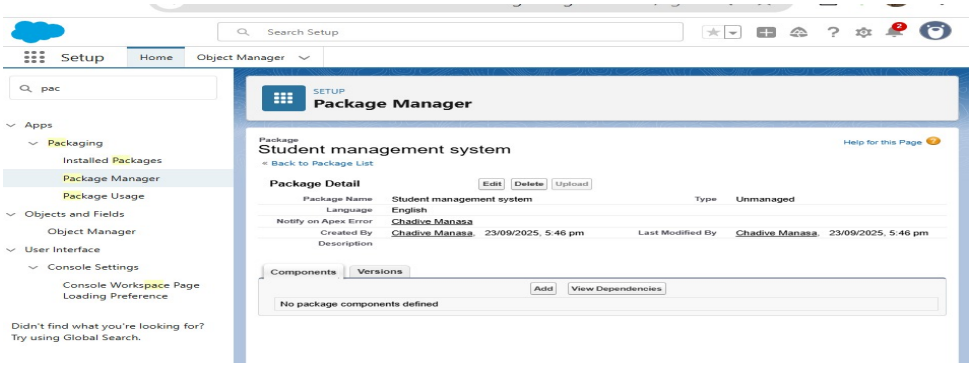


4. **Data Export & Backup**
    - Scheduled weekly data export from Setup for backup.
    - Downloaded CSVs of critical objects (Students, Courses, Enrollments).
    - Stored backup securely for rollback scenarios.

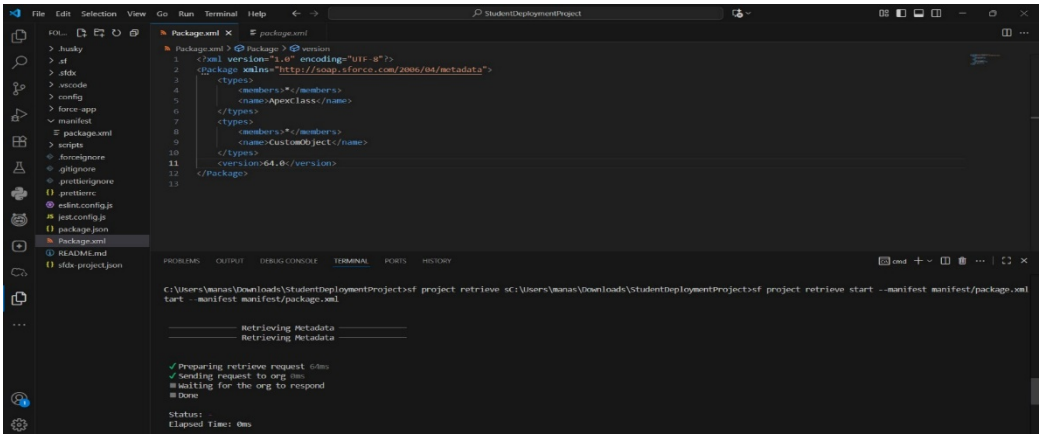


5. **Change Sets**
    - Created outbound change sets with custom objects and fields.
    - Uploaded change sets to target sandbox and deployed.
    - Validated deployment results post-migration.

6. **Unmanaged vs Managed Packages**
    - Built an unmanaged package containing custom objects for sharing.
    - Created a managed package to test namespace and code protection.
    - Installed both in a new org and documented differences.



7. **ANT Migration Tool**
    - Configured build.xml and package.xml for metadata retrieval.
    - Retrieved custom object and Apex class metadata from source org.
    - Deployed changes to target org and validated success logs.
8. **VS Code & SFDX**
    - Set up VS Code project with Salesforce CLI and authorized orgs.
    - Retrieved metadata using sf project retrieve start.
    - Modified and deployed changes back using sf project deploy start.



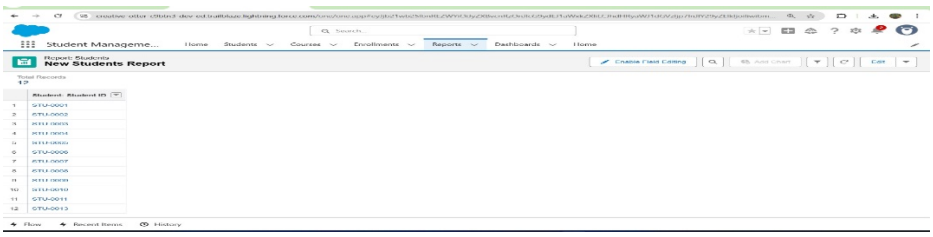
Phase 9: Reporting, Dashboards & Security Review

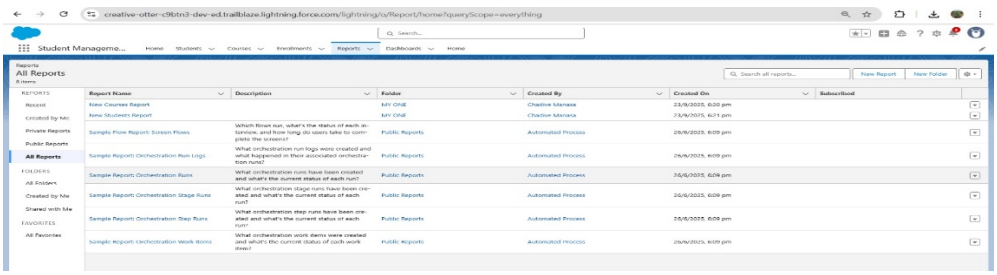
Goal :

Deliver **powerful reporting & analytics** (Reports, Dashboards, Dynamic Dashboards) and implement **robust security controls** (Sharing Settings, Field-Level Security, Session Settings, IP Ranges, Audit Trail) to ensure data is both actionable and protected.

1. Reports (Tabular, Summary, Matrix, Joined)

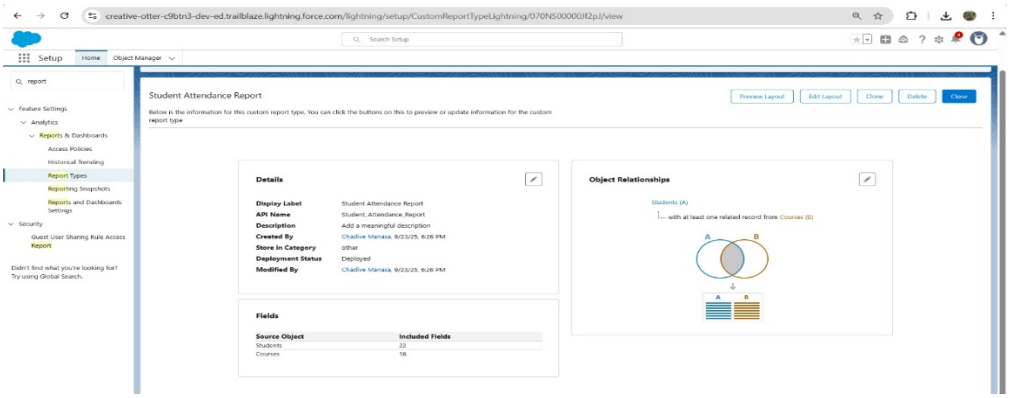
- **Tabular Reports:**
  - Student List Report (Basic student details)
  - Attendance List Report (Daily/Monthly attendance)
- **Summary Reports:**
  - Class-wise Student Count
  - Attendance Percentage by Class
  - Grade Distribution Summary
- **Matrix Reports:**
  - Attendance Matrix (Rows = Classes, Columns = Months)
  - Grade Performance Matrix (Rows = Subjects, Columns = Terms)
- **Joined Reports:**
  - Student + Attendance + Grades combined view for holistic performance analysis
  - Course Enrollment + Completion Rate joined report

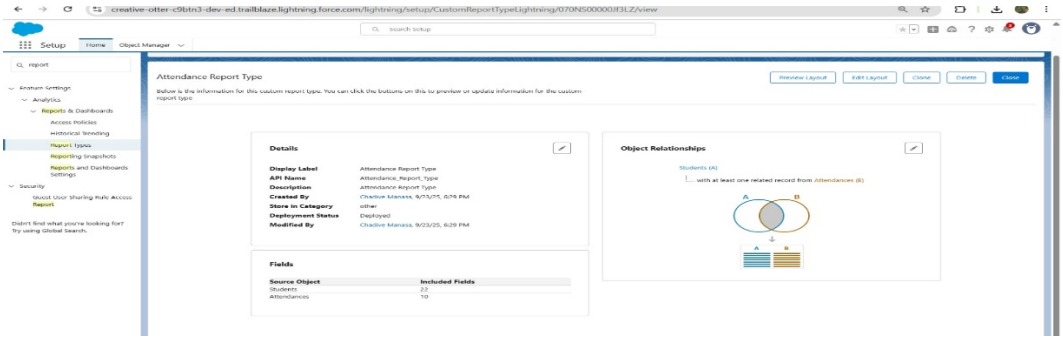




2. Report Types

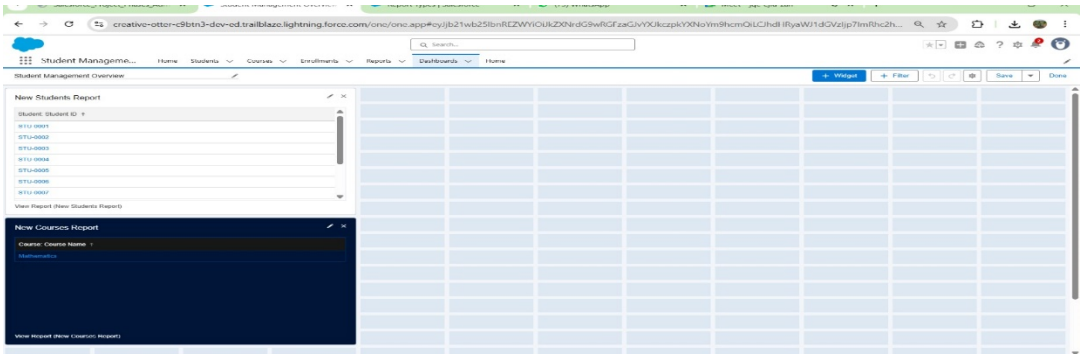
- **Student Report Type:** Includes student personal details, contact info
- **Attendance Report Type:** Links Student → Attendance records
- **Grades Report Type:** Links Student → Grades with term/subject info
- **Courses Report Type:** Links Student → Courses enrolled, status, teacher details





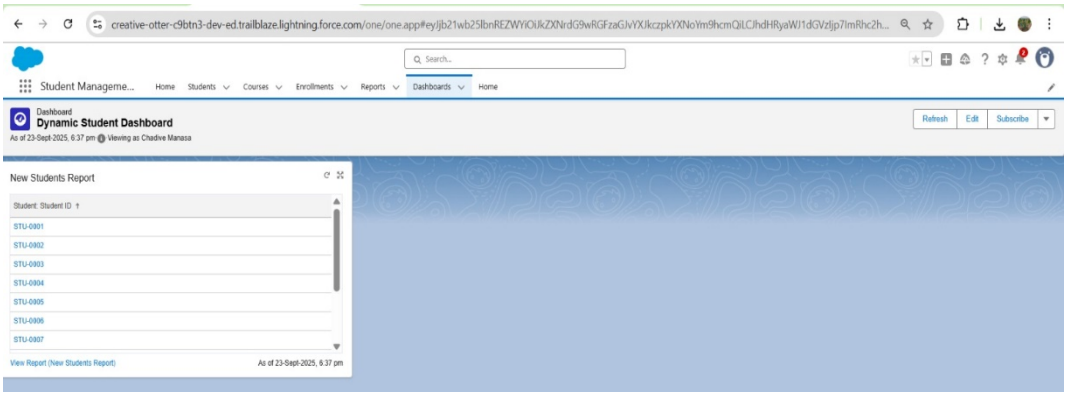
3. Dashboards

- **Student Overview Dashboard:** Total Students, New Admissions, Dropouts
- **Attendance Dashboard:** Present %, Absent %, Class-wise charts
- **Grades Dashboard:** Average Grades per Class, Top Performers, At-Risk Students
- **Course Dashboard:** Enrolment Trends, Most Popular Courses



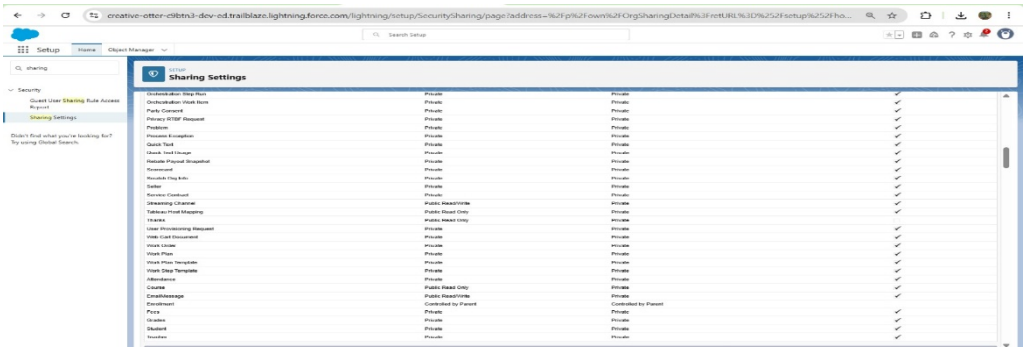
4. Dynamic Dashboards

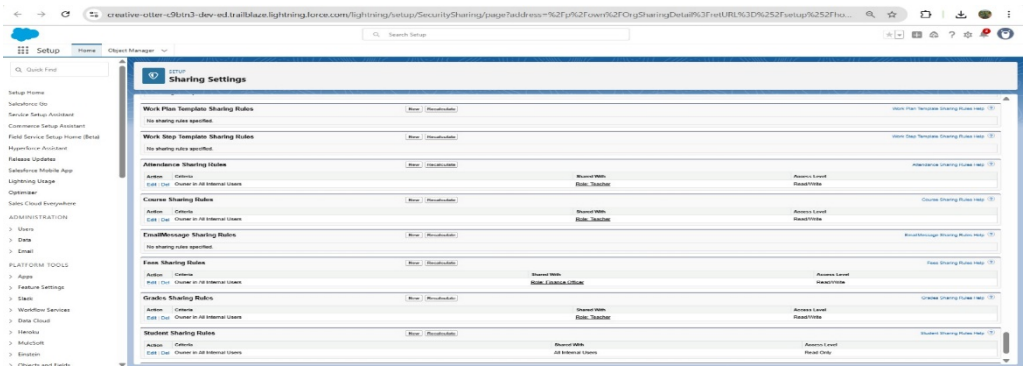
- Configured dashboards to **run as logged-in user** so each Teacher sees only their class data, Principal sees their school data, and Admin sees all data.
- Applied **dashboard filters** (Class, Term, Subject) for real-time drill-down.



5. Sharing Settings

- **Organization-Wide Defaults (OWD):**
  - Students, Attendance, Grades = Private
  - Courses = Controlled by Parent

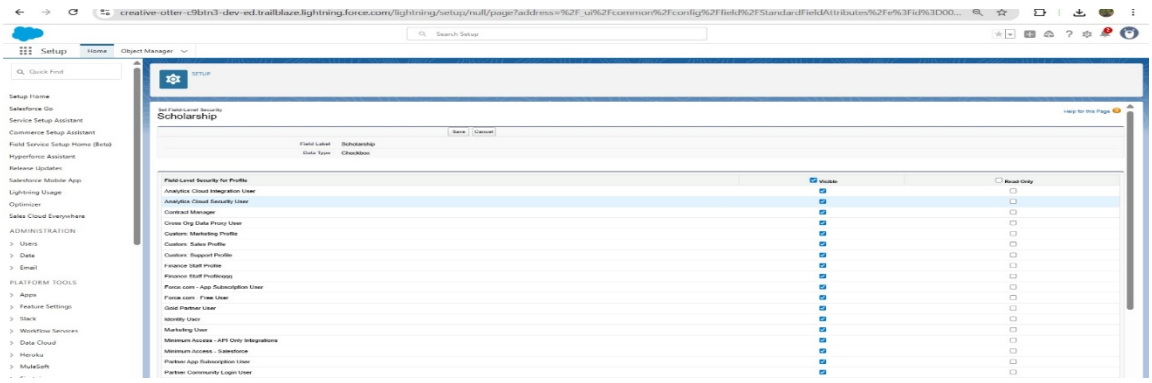




- **Role Hierarchy:**
    - Admin > Principal > Teacher > Student
- **Sharing Rules:**
    - Criteria-based rules to share records with Teacher Groups based on Class/Section
    - Owner-based sharing for Principals to access all data under their school

6. Field-Level Security

- **Sensitive Fields Restricted:**
    - SSN, Parent Contact Info → Visible to Admin & Principal only
    - Fee Details → Visible to Admin only
    - Grades → Read-Only for Teachers, Visible to Principal/Admin
- **Profiles Configured:** Admin, Principal, Teacher, Student with tailored field visibility



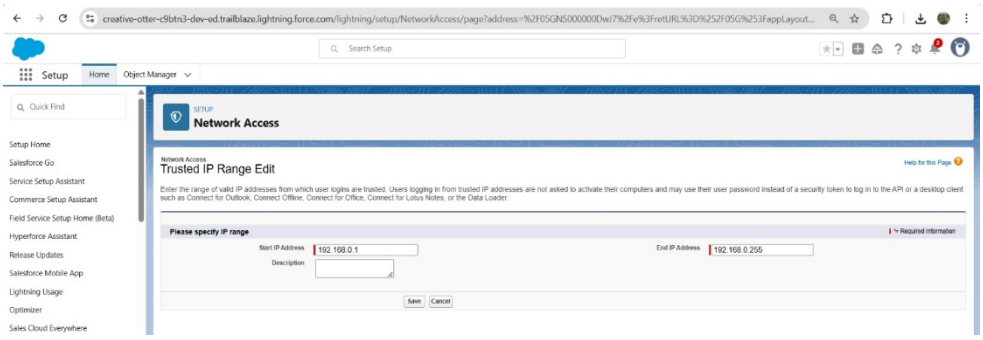
7. Session Settings

- **Session Timeout:** 30 minutes
- **Logout on Timeout:** Enabled
- **Lock Session to IP/Domain:** Enabled
- **Clickjack & CSRF Protection:** Enabled
- **HTTPS Required:** Enforced for secure communication

8. Login IP Ranges

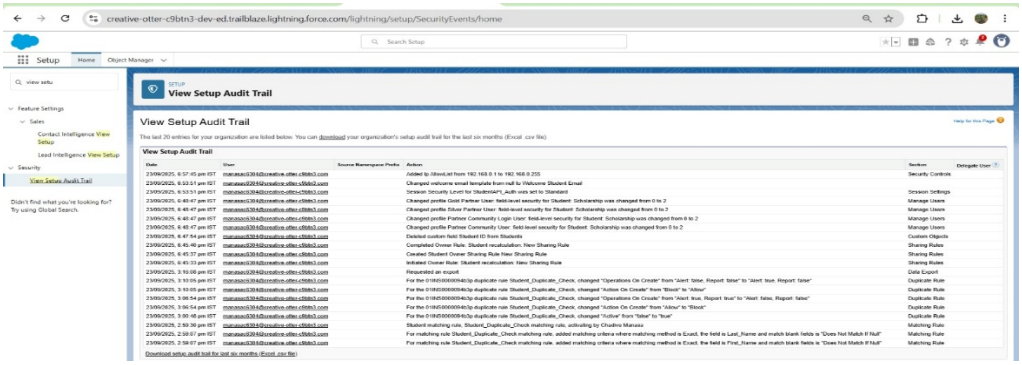
- **Network Access (Org-Wide):** Added school's LAN/Wi-Fi IP range
- **Profile-Specific Ranges:**
    - Teachers: Allowed only from school network

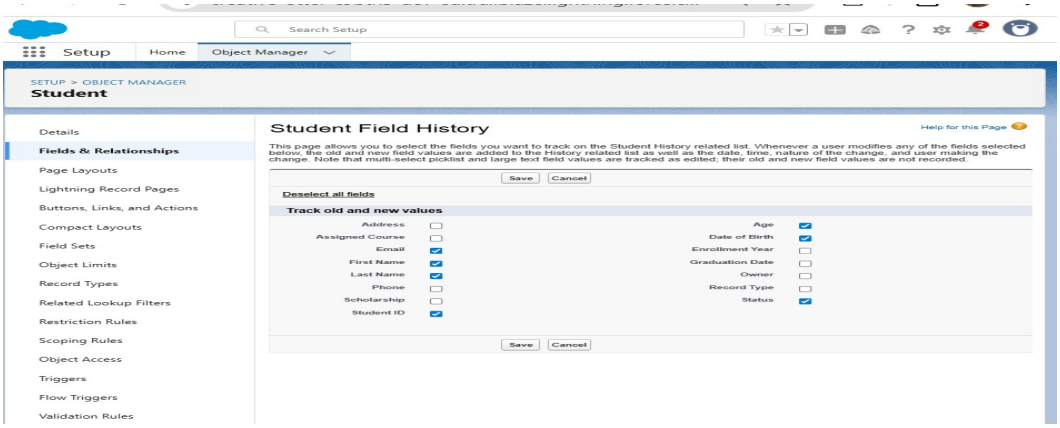
- Admin: Allowed from office network + VPN
- Students: Allowed from broader IP range (school + home if needed)

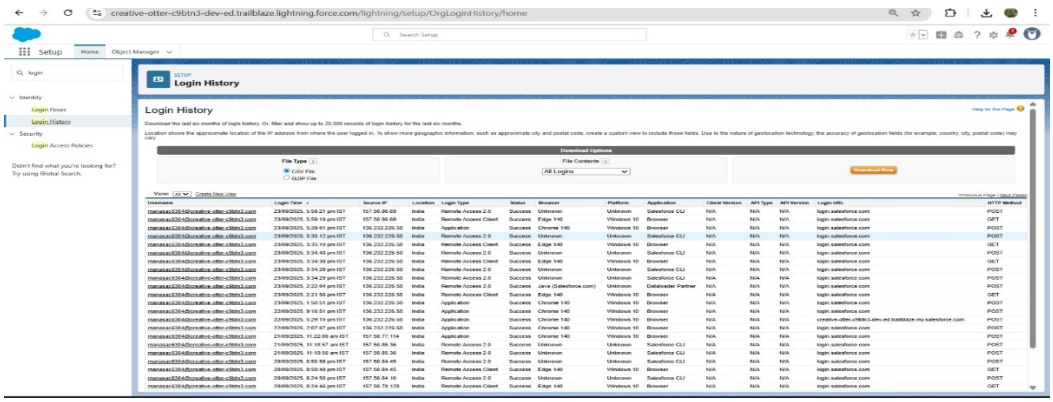


9. Audit Trail

- **Setup Audit Trail:** Enabled and reviewed for changes in roles, profiles, and settings
- **Field History Tracking:**
  - Tracked Fields: Student Status, Assigned Teacher, Grade Changes, Attendance Updates
- **Login History:** Monitored failed logins and suspicious attempts






creative-otter-c9btn3-dev-ed.trailblaze.lightning.force.com/lightning/setup/OrgLoginHistory/home
Setup
Home
Object Manager
Login History
Login History
Download Options
File Type
CSV File
GZIP File
File Contents
All Logins
Download Now