

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.

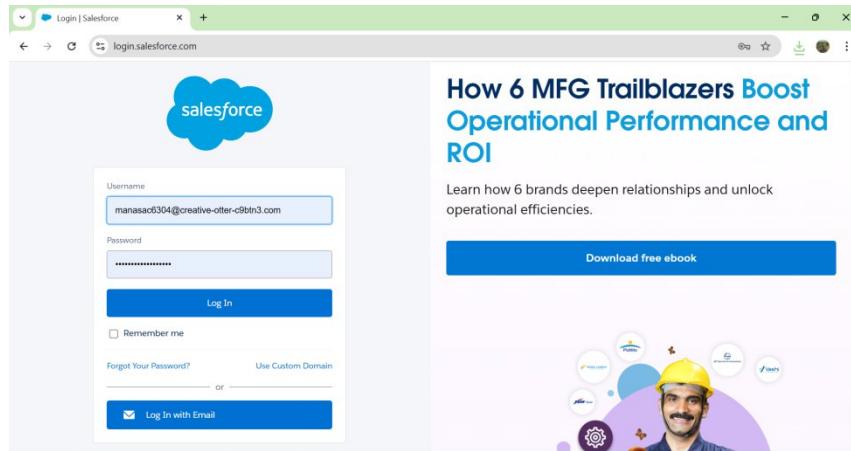
5. AppExchange Exploration

- 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

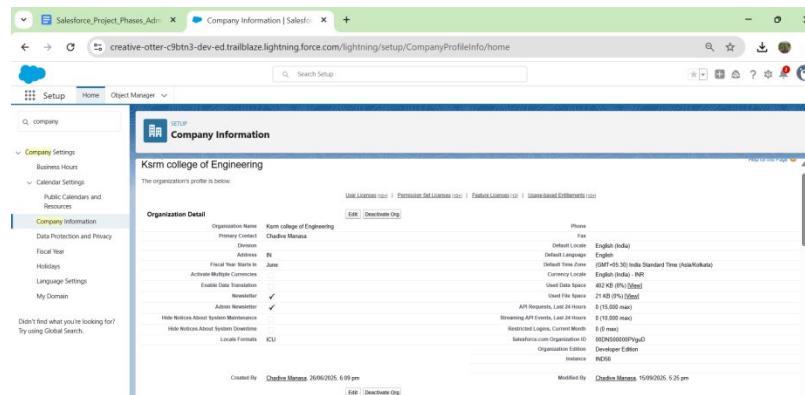
1. Salesforce Edition

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



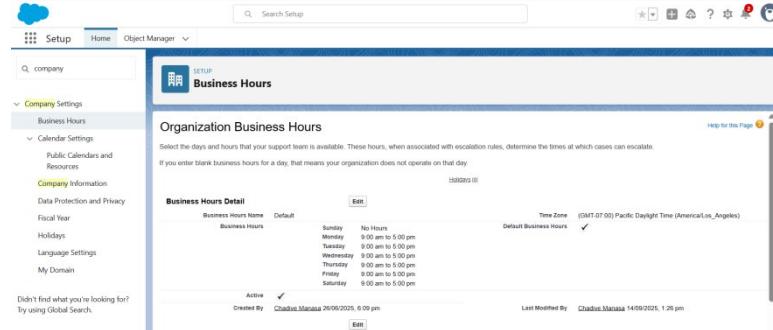
2. Company Profile Setup

- 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).

The screenshot shows the 'Holidays' setup page in Salesforce. The left sidebar includes 'Company Settings', 'Calendar Settings', 'Company Information', 'Data Protection and Privacy', 'Fiscal Year', 'Holidays' (which is selected), 'Language Settings', and 'My Domain'. The main content area displays a table for 'Holidays' with columns for Action, Holiday Name, Description, and Date and Time. Two entries are listed: 'Independence Day' (15/08/2026 All Day) and 'Republic Day' (26/01/2026 All Day). Below this is a section for 'Elapsed Holidays' which shows 'No records to display'.

4. Fiscal Year Settings

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

The screenshot shows the 'Fiscal Year' setup page in Salesforce. The left sidebar includes 'Company Settings', 'Calendar Settings', 'Company Information', 'Data Protection and Privacy', 'Fiscal Year' (selected), 'Holidays', 'Language Settings', and 'My Domain'. The main content area shows the 'Organization Fiscal Year Edit' screen for 'Ksm college of Engineering'. It allows specifying the fiscal year type (Standard Fiscal Year or Custom Fiscal Year) and changing the fiscal year period. A note states: 'Changing the fiscal year shifts fiscal periods and impacts opportunities and forecasts across your organization. If your forecast periods are set to quarterly, adjusting the fiscal year start month will erase existing forecast adjustments and quotas. Consider exporting a data backup before implementing this change.' The 'Change Fiscal Year Period' dialog is open, showing 'Name: Ksm college of Engineering', 'Fiscal Year Start Month: June', and 'Fiscal Year Is Based On: The ending month'.

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.

The screenshot shows the 'Users' setup page in Salesforce. The left sidebar includes 'Users' (selected), 'Permission Set Groups', 'Profiles', 'Public Groups', 'Queues', 'Roles', 'User Management Settings', 'Feature Settings', 'Data.com', 'Prospector Users', 'Service', and 'Embedded Service'. The main content area shows a table for 'All Users' with columns for Action, Full Name, Alias, Username, Role, Active, and Profile. Six users are listed: 'Chatter Expert' (larkin_teach), 'Manasa_Chadiv' (cmanna), 'Mikaelson_Fink' (fmika), 'integr' (integr), and 'sec' (sec). The 'Active' column shows checkboxes for each user, and the 'Profile' column lists their respective profiles: Teacher, Teacher, System Administrator, Finance Officer, and Analytics Cloud Integration User.

6. Profiles

- Created/Modified Profiles:

- System Administrator (default full access).

-

The screenshot shows the Salesforce Setup interface under the Profiles section. The profile displayed is 'System Administrator'. The 'Profile Detail' section shows the name 'System Administrator', user license 'Salesforce', and creation details ('Created By: salesforce.com, inc., 26/06/2025, 6:09 pm'). The 'Custom Profile' checkbox is checked. The 'Page Layouts' section lists various standard object layouts and custom layouts for objects like Invoice, Invoice Line, Lead, and Lead Entity.

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

The screenshot shows the Salesforce Setup interface under the Profiles section. The profile displayed is 'Teacher Profile'. The 'Profile Detail' section shows the name 'Teacher Profile', user license 'Salesforce', and creation details ('Created By: Chadiye Manasa, 14/09/2025, 5:54 pm'). The 'Custom Profile' checkbox is checked. The 'Page Layouts' section lists various standard object layouts and custom layouts for objects like Invoice, Invoice Line, Lead, and Lead Entity.

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

7. Roles

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

Profile Detail

| Name | Finance Staff Profile | Custom Profile | ✓ |
|--|-----------------------|----------------|----------------|
| User License | Salesforce Platform | | |
| Description | | | |
| Created By | Chadive Manasa | Modified By | Chadive Manasa |
| 15/09/2025, 5:10 pm 17/09/2025, 7:09 pm | | | |

Page Layouts

| Standard Object Layouts | Global | Data Use Purpose | Data Use Purpose Layout |
|-------------------------|--|------------------|---|
| Email Application | Not Assigned [View Assignment] | Email Message | Email Message Layout [View Assignment] |
| Home Page Layout | Home Page Default [View Assignment] | Event | Event Layout [View Assignment] |

- Teacher

Creating the Role Hierarchy

You can build on the existing role hierarchy shown on this page. To insert a new role, click **Add Role**.

Your Organization's Role Hierarchy

[Collapse All](#) [Expand All](#)

- [-] Ksrm college of Engineering
 - [+][CEO](#) [Edit](#) | [Del](#) | [Assign](#)
 - [+][Add Role](#)
 - [+][Principal](#) [Edit](#) | [Del](#) | [Assign](#)
 - [+][Add Role](#)
 - [+][Finance Officer](#) [Edit](#) | [Del](#) | [Assign](#)
 - [+][Add Role](#)
 - [+][HOD](#) [Edit](#) | [Del](#) | [Assign](#)
 - [+][Add Role](#)
 - [-] **Teacher** [Edit](#) | [Del](#) | [Assign](#)
 - [+][Add Role](#)

8

8.Permission Sets

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

The screenshot shows the Salesforce Setup interface. The left sidebar has 'Setup' selected. Under 'Users', 'Permission Sets' is highlighted. The main area shows the 'Fee Management Access' permission set. A table titled 'Current Assignments' lists one assignment: 'Fink Mikaelson' with 'Active' status, 'Finance Officer' role, 'Finance Staff Profile' profile, and 'Salesforce Platform' user license. The table has columns for Full Name, Active, Role, Profile, User License, and Expires On.

| Full Name ↑ | Active | Role | Profile | User License | Expires On |
|----------------|--------|-----------------|-----------------------|---------------------|------------|
| Fink Mikaelson | ✓ | Finance Officer | Finance Staff Profile | Salesforce Platform | |

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 →
- Course →
- Only
- Enrollment
- Attendance
- Grades →
- Fees →

Defaults)

Private

Public Read

→ Private

→ Private

Private

Private

The screenshots show the 'Details' tab for four custom objects in the Salesforce Setup Object Manager:

- Course:** API Name: Course, Singular Label: Course, Plural Label: Courses. Permissions: Enable Reports (checked), Track Activities (checked), Track Field History (checked). Deployment Status: Deployed, Help Setting: Standard Salesforce.com Help Window.
- Student:** API Name: Student, Singular Label: Student, Plural Label: Students. Permissions: Enable Reports (checked), Track Activities (checked), Track Field History (checked). Deployment Status: Deployed, Help Setting: Standard Salesforce.com Help Window.
- Fee:** API Name: Fee, Singular Label: Fee, Plural Label: Fees. Permissions: Enable Reports (checked), Track Activities (checked), Track Field History (checked). Deployment Status: Drafted, Help Setting: Standard Salesforce.com Help Window.
- Attendance:** API Name: Attendance, Singular Label: Attendance, Plural Label: Attendances. Permissions: Enable Reports (checked), Track Activities (checked), Track Field History (checked). Deployment Status: Deployed, Help Setting: Standard Salesforce.com Help Window.
- Enrollment:** API Name: Enrollment, Singular Label: Enrollment, Plural Label: Enrollments. Permissions: Enable Reports (checked), Track Activities (checked), Track Field History (checked). Deployment Status: Drafted, Help Setting: Standard Salesforce.com Help Window.
- Grades:** API Name: Grade, Singular Label: Grade, Plural Label: Grades. Permissions: Enable Reports (checked), Track Activities (checked), Track Field History (checked). Deployment Status: Drafted, Help Setting: Standard Salesforce.com Help Window.

The screenshot shows the Salesforce Sharing Settings page. The left sidebar has a search bar and navigation links for Setup, Home, and Object Manager. Under Security, it lists 'Sharing Settings' which is selected. A message says 'Didnt find what you're looking for? Try using Global Search.' The main content area is titled 'Sharing Settings' and contains a table of sharing rules. The table has two columns of dropdown menus for 'Share With' and 'Access Level'. Most rows have 'Private' selected for both. Some rows like 'Attendance' and 'Fees' have 'Public Read Only' or 'Public Read/Write' selected. There are also some checked checkboxes in the last column.

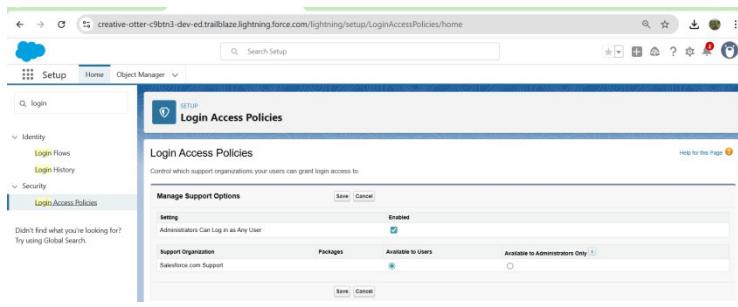
11. Sharing Rules

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

The screenshot shows the Salesforce Sharing Rules page. The left sidebar has a search bar and navigation links for Setup, Home, and Object Manager. Under Security, it lists 'Sharing Rules' which is selected. A message says 'Didnt find what you're looking for? Try using Global Search.' The main content area is titled 'Sharing Rules' and contains sections for different objects: Attendance, Course, Fees, Grades, Student, and Teacher. Each section has a 'New' button and a 'Recents' list. The 'Attendance Sharing Rules' section shows a rule for 'Owner' with 'Share With' set to 'Role: Teacher' and 'Access Level' to 'Read/Write'. The 'Course Sharing Rules' section shows a similar rule for 'Owner'. The 'Fees Sharing Rules' section shows a rule for 'Owner' with 'Share With' set to 'Role: Finance Office' and 'Access Level' to 'Read/Write'. The 'Grades Sharing Rules' section shows a rule for 'Owner' with 'Share With' set to 'Role: Teacher' and 'Access Level' to 'Read/Write'. The 'Student Sharing Rules' section shows a rule for 'Owner' with 'Share With' set to 'Role: Teacher' and 'Access Level' to 'Read/Write'. The 'Teacher Sharing Rules' section shows a rule for 'Owner' with 'Share With' set to 'Role: Teacher' and 'Access Level' to 'Read/Write'.

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.

The image displays four separate screenshots of the Salesforce Object Manager interface, each showing the creation and configuration of a custom object. The objects shown are:

- Fees**: A custom object with fields including Amount, Description, and Type. It has relationships with Account and Contact.
- Attendance**: A custom object with fields including Description, Date, and Status. It has relationships with Account and Contact.
- Enrollment**: A custom object with fields including Grade, Student, and Course. It has relationships with Account, Contact, and Course.
- Grades**: A custom object with fields including Grade, Description, and Score. It has relationships with Course and Enrollment.

Each screenshot shows the 'Details' tab of the object's setup page, including its API name, description, and various configuration options like field sets, buttons, and tracks.

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.

| 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) | | ✓ |

| FIELD LABEL | FIELD NAME | DATA TYPE | CONTROLLING FIELD | INDEXED |
|------------------|--------------------|--------------------|-------------------|---------|
| Address | Address__c | Text(Address) | | |
| Age | Age__c | Number(2, 0) | | |
| Assigned Course | Assigned_Course__c | Lookup(Course) | | ✓ |
| Created By | CreatedById | Lookup(User) | | ✓ |
| Date of Birth | Date_of_Birth__c | Date | | |
| Email | Email__c | Email | | |
| Enrollment Year | Enrollment_Year__c | Text(10) | | |
| First Name | First_Name__c | Text(255) | | |
| Graduation Date | Graduation_Date__c | Date | | |
| Last Modified By | LastModifiedById | Lookup(User) | | |
| Last Name | Last_Name__c | Text(255) | | |
| Owner | Owner | Lookup(User/Group) | | |
| Phone | Phone__c | Phone | | |
| Record Type | RecordTypeId | Record Type | | ✓ |
| Scholarship | Scholarship__c | Checkbox | | |
| Status | Status__c | Picklist | | |
| Student ID | Student_ID__c | Auto Number | | |

| FIELD LABEL | FIELD NAME | DATA TYPE | CONTROLLING FIELD | INDEXED |
|------------------|------------------|--------------------|-------------------|---------|
| Course | Course__c | Lookup(Course) | | ✓ |
| Course Code | Course_Code__c | Text(60) | | |
| Course ID | Course_ID__c | Auto Number | | |
| Course Name | Course_Name__c | Text(80) | | |
| Name | Name | Text(80) | | |
| Created By | CreatedById | Lookup(User) | | ✓ |
| Credits | Credits__c | Number(10, 0) | | |
| Duration | Duration__c | Number(10, 0) | | |
| Last Modified By | LastModifiedById | Lookup(User) | | |
| Owner | Owner | Lookup(User/Group) | | |
| Record Type | RecordTypeId | Record Type | | ✓ |
| Student | Student__c | Lookup(Student) | | ✓ |
| Teacher | Teacher__c | Lookup(Teacher) | | ✓ |

Setup > Object Manager

Teacher

| Fields & Relationships | | | | |
|------------------------|------------------|--------------------|-------------------|---------|
| 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) | | |

Setup > Object Manager

Fees

| Fields & Relationships | | | | |
|------------------------|--------------------|--------------------|-------------------|---------|
| 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) | | |

Setup > Object Manager

Enrollment

| Fields & Relationships | | | | |
|------------------------|--------------------|------------------------|-------------------|---------|
| 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.

The screenshot shows the Salesforce Object Manager interface for the 'Course' object. The left sidebar has a 'Record Types' section selected. The main area displays a table titled 'Record Types' with two items:

| RECORD TYPE LABEL | DESCRIPTION | ACTIVE | MODIFIED BY |
|-------------------|-----------------------------------|--------|-------------------------------------|
| Lab Course | Record type for lab-based courses | ✓ | Chadive Manasa, 17/09/2025, 5:26 pm |
| Theory Course | Record type for theory courses | ✓ | Chadive Manasa, 17/09/2025, 5:26 pm |

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.

The screenshot shows the Salesforce Object Manager interface for the 'Teacher' object. The left sidebar has a 'Page Layouts' section selected. The main area displays a table titled 'Page Layouts' with one item:

| PAGE LAYOUT NAME | CREATED BY | MODIFIED BY |
|------------------|-------------------------------------|-------------------------------------|
| Teacher Layout | Chadive Manasa, 17/09/2025, 6:36 pm | Chadive Manasa, 17/09/2025, 7:03 pm |

The screenshot shows the Salesforce Object Manager interface for the 'Student' object. The left sidebar has a 'Page Layouts' section selected. The main area displays a table titled 'Page Layouts' with three items:

| PAGE LAYOUT NAME | CREATED BY | MODIFIED BY |
|------------------------|-------------------------------------|-------------------------------------|
| Postgraduate Layout | Chadive Manasa, 17/09/2025, 5:28 pm | Chadive Manasa, 18/09/2025, 9:10 pm |
| Student Layout | Chadive Manasa, 14/09/2025, 5:58 pm | Chadive Manasa, 18/09/2025, 9:10 pm |
| Teacher Student Layout | Chadive Manasa, 15/09/2025, 5:37 pm | Chadive Manasa, 18/09/2025, 9:10 pm |

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.

The screenshot shows the Salesforce Setup interface. The left sidebar has a search bar and navigation links for Home, Object Manager, and various setup categories like Email, Apps, and Integrations. Under Integrations, 'External Data Sources' is selected, and 'External Objects' is highlighted. The main content area is titled 'External Objects' and contains a table of existing external objects. The table has columns for Action, Label, Deployed, External Data Source, and Description. The data is as follows:

| Action | Label | Deployed | External Data Source | Description |
|--------------|---------------|--------------------------|----------------------|----------------|
| Edit Erase | Advertisement | <input type="checkbox"/> | LibrarySystem | Advertisements |
| Edit Erase | Category | <input type="checkbox"/> | LibrarySystem | Categories |
| Edit Erase | PersonDetail | <input type="checkbox"/> | LibrarySystem | PersonDetails |
| Edit Erase | Product | <input type="checkbox"/> | LibrarySystem | Products |
| Edit Erase | Supplier | <input type="checkbox"/> | LibrarySystem | Suppliers |

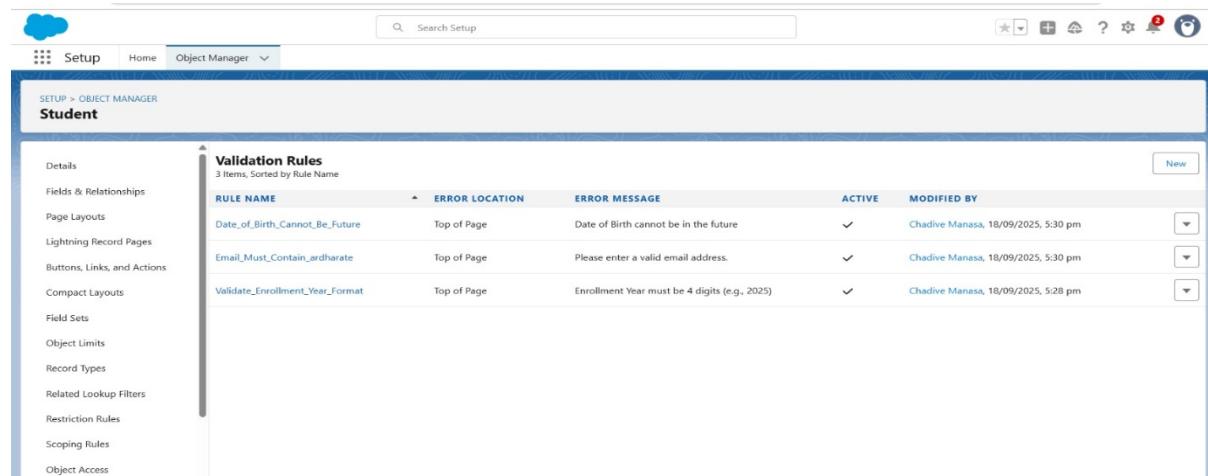
Phase 4: Process Automation (Admin)

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."



The screenshot shows the Salesforce Object Manager interface for the 'Student' object. On the left, there's a sidebar with various setup options like Details, Fields & Relationships, Page Layouts, etc. The main area is titled 'Validation Rules' and shows three items listed:

| RULE NAME | ERROR LOCATION | ERROR MESSAGE | ACTIVE | MODIFIED BY |
|---------------------------------|----------------|---|--------|-------------------------------------|
| Date_of_Birth_Cannot_Be_Future | Top of Page | Date of Birth cannot be in the future | ✓ | Chadive Manasa, 18/09/2025, 5:30 pm |
| Email_Must_Contain_@ndharate | Top of Page | Please enter a valid email address. | ✓ | Chadive Manasa, 18/09/2025, 5:30 pm |
| Validate_Enrollment_Year_Format | Top of Page | Enrollment Year must be 4 digits (e.g., 2025) | ✓ | Chadive Manasa, 18/09/2025, 5:28 pm |

2. Workflow Rules

Object: Student__c

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

Workflow Rules

Student Enrollment Welcome

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

| | | |
|--|--------------------------------------|---|
| Rule Name | Student Enrollment Welcome | Edit Close Deactivate |
| Active | ✓ | |
| Description | Student__Status EQUALS Active | |
| Rule Criteria | (Student__Status EQUALS Active) | |
| Created By | Chadive Manasa, 10/09/2025, 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, 10/09/2025, 11:16 pm | |

Workflow Actions

Immediate Workflow Actions

| | |
|-------------|-----------------------|
| Type | Description |
| Email Alert | Welcome Student Email |

Time-Dependent Workflow Actions See an example

Warning: You cannot add new time triggers to an active rule. [Deactivate This Rule](#)

[Edit](#)

Workflow Rules

Verify Student Documents

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

| | | |
|--|---|---|
| Rule Name | Verify Student Documents | Edit Close Deactivate |
| Active | ✓ | |
| Description | Verify student's submitted documents | |
| Rule Criteria | (Student__Status EQUALS Pending Docs) AND (Student__Status NOT EQUALS Enrolled) | |
| Created By | Chadive Manasa, 10/09/2025, 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, 10/09/2025, 11:16 pm | |

Workflow Actions

Immediate Workflow Actions

| | |
|------|--------------------------------------|
| Type | Description |
| Task | Verify student's submitted documents |

Time-Dependent Workflow Actions See an example

Warning: You cannot add new time triggers to an active rule. [Deactivate This Rule](#)

[Edit](#)

Workflow Rules

Course Allocation Task

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

| | | |
|--|--|---|
| Rule Name | Course Allocation Task | Edit Close Deactivate |
| Active | ✓ | |
| Description | Assign student to first semester courses | |
| Rule Criteria | (Student__Status EQUALS Enrolled) | |
| Created By | Chadive Manasa, 10/09/2025, 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, 10/09/2025, 9:44 pm | |

Workflow Actions

Immediate Workflow Actions

| | |
|------|--|
| Type | Description |
| Task | Assign student to first semester courses |

Time-Dependent Workflow Actions See an example

Warning: You cannot add new time triggers to an active rule. [Deactivate This Rule](#)

[Edit](#)

Workflow Rules

Inactive Student Update

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

| | | |
|--|--|---|
| Rule Name | Inactive Student Update | Edit Close Deactivate |
| Active | ✓ | |
| Description | Student__Enrollment_Year LESS THAN TEXT(YEAR(TODAY())) | |
| Rule Criteria | (Student__Enrollment_Year LESS THAN TEXT(YEAR(TODAY()))) | |
| Created By | Chadive Manasa, 10/09/2025, 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, 10/09/2025, 9:46 pm | |

Workflow Actions

Immediate Workflow Actions

| | |
|--------------|-------------|
| Type | Description |
| Field Update | Status |

Time-Dependent Workflow Actions See an example

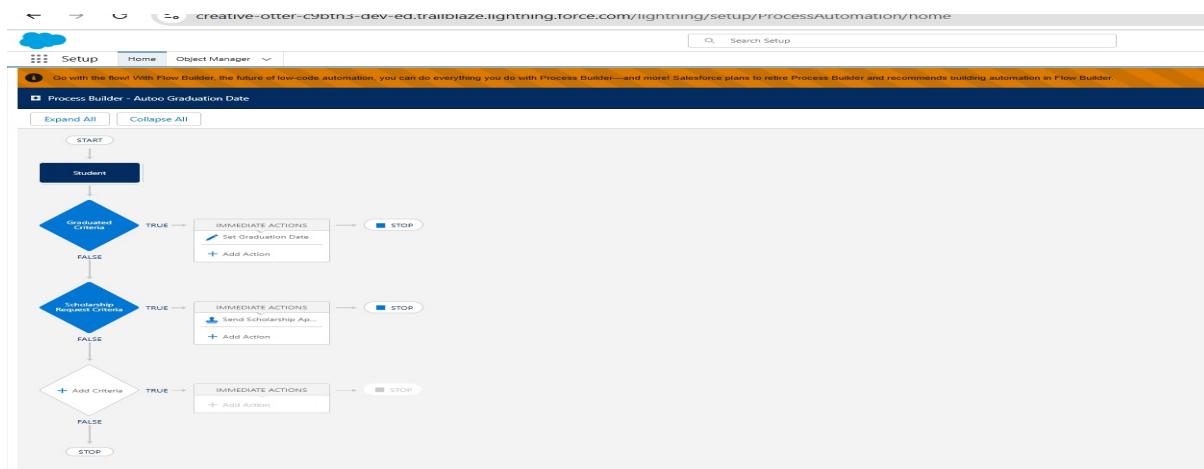
Warning: You cannot add new time triggers to an active rule. [Deactivate This Rule](#)

[Edit](#)

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".

The screenshot shows the 'Approval Processes' configuration page for the 'Student Scholarship Approval Process'. It displays various settings and steps:

- Process Definition Detail:** Shows the process name and description.
- Initial Submission Actions:** Includes 'Lock Record' and 'Update Record' steps.
- Assignment Settings:** Shows the 'Finance Staff' queue assigned to the process.
- Final Approval Actions:** Includes 'Unlock Record' and 'Update Record' steps.
- Final Rejection Actions:** Includes 'Update Record' steps.
- General Actions:** Includes 'Email Message' and 'Update Record' steps.

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

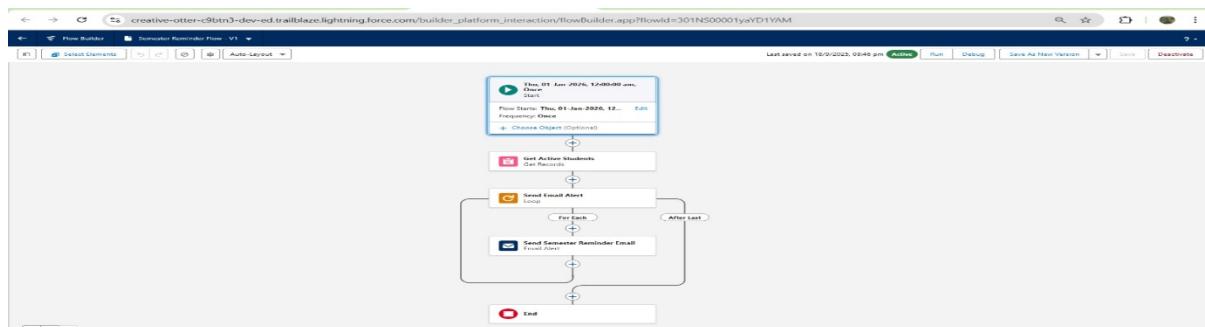
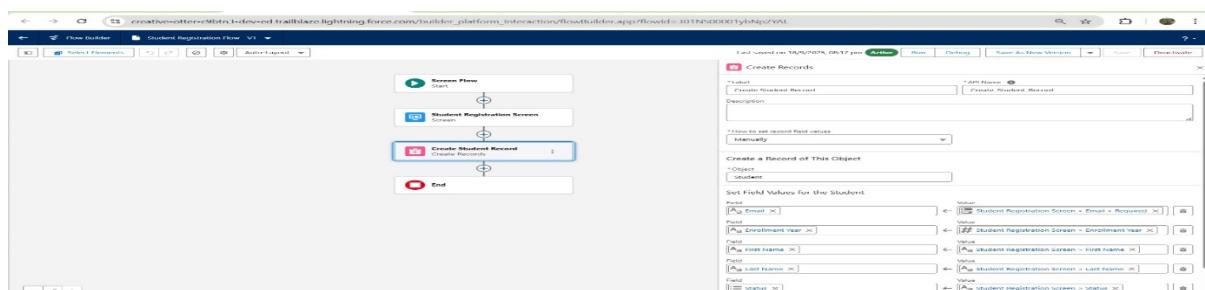
- When:** A new Student record is created.
- Condition:** If Enrollment_Year__c = TEXT(YEAR(TODAY()))
- Action:** Update Status__c = "Enrolled".

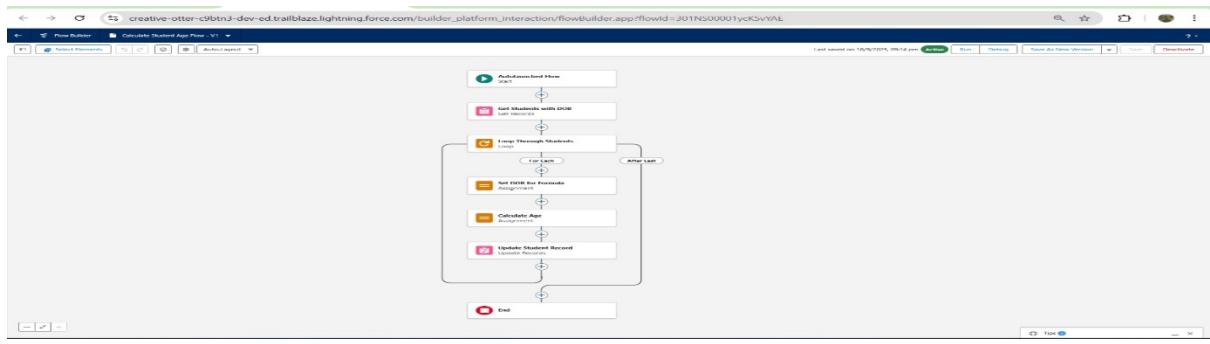
c) Scheduled Flow: Semester Reminder

- 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.

| Email Alert Detail | | Modified By | |
|--------------------|---------------------------------|---------------|---------------------|
| Description | Welcome Student Email | Object | Shadow Manager |
| From Email Address | ShadowManager_email@outlook.com | Created Date | 10/09/2020, 9:29 AM |
| To Email Address | ShadowManager_email@outlook.com | Modified Date | 10/09/2020, 9:29 AM |
| Automatic Response | None | Active | ✓ |
| Custom Obj | | | |

Rules Using This Email Alert

This alert is currently not used by any rules.

Approval Processes Using This Email Alert

This alert is currently not used by any approval processes.

Entitlement Processes Using This Email Alert

This alert is currently not used by any entitlement processes.

Flows Using This Email Alert

No flows.

| Email Alert Detail | | Modified By | |
|--------------------|--|---------------|---------------------|
| Description | Send Scholarship Approval Email after approval process | Object | Shadow Manager |
| From Email Address | ShadowManager_email@outlook.com | Created Date | 10/09/2020, 9:29 AM |
| To Email Address | ShadowManager_email@outlook.com | Modified Date | 10/09/2020, 9:29 AM |
| Automatic Response | None | Active | ✓ |
| Custom Obj | | | |

Rules Using This Email Alert

This alert is currently not used by any rules.

Approval Processes Using This Email Alert

This alert is currently not used by any approval processes.

Entitlement Processes Using This Email Alert

This alert is currently not used by any entitlement processes.

Flows Using This Email Alert

No flows.

| Email Alert Detail | | Modified By | |
|--------------------|---|---------------|---------------------|
| Description | Send Congratulations Email when a student graduates | Object | Shadow Manager |
| From Email Address | ShadowManager_email@outlook.com | Created Date | 10/09/2020, 9:29 AM |
| To Email Address | ShadowManager_email@outlook.com | Modified Date | 10/09/2020, 9:29 AM |
| Automatic Response | None | Active | ✓ |
| Custom Obj | | | |

Rules Using This Email Alert

This alert is currently not used by any rules.

Approval Processes Using This Email Alert

This alert is currently not used by any approval processes.

Entitlement Processes Using This Email Alert

This alert is currently not used by any entitlement processes.

Flows Using This Email Alert

No flows.

The screenshot shows the 'Email Alerts' configuration screen. It displays the alert details, including the subject line 'Reminder: Remind me about my scholarship application', recipient 'Student__c', and message body. Below the alert details, sections for 'Rules Using This Email Alert', 'Approved Processes Using This Email Alert', and 'Entitlement Processes Using This Email Alert' are visible. A note states that this alert is currently not used by any entitlement processes.

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."

The screenshot shows the creation of a new Workflow Task. The task is titled 'Verify student's submitted documents'. It is set to run on 'Student__c' and has a scheduled time of 'Now'. The task description is 'Verify student's submitted documents'. The status is 'Not Started' and the priority is 'High'. The task is assigned to 'Student__c'. A note indicates that this task is currently not used by any entitlement processes.

The screenshot shows the configuration of an 'Email Alerts' rule for 'Welcome Student Email'. The rule is triggered by 'Student__c' and 'Student__c' being updated. The email template is 'Welcome Student Email'. The recipient is 'Student__c'. The message body includes 'Welcome Student Email' and 'Default Workflow User's email address'. The rule is assigned to 'Student__c'. A note states that this rule is currently not used by any entitlement processes.

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.

The screenshot shows the 'Custom Notifications' configuration screen. It displays the notification details, including the notification name 'Pending Document Notification' and API name 'Pending_Document_Notification'. The namespace is 'Student__c'. The desktop and mobile settings are both checked. A note states that when you create and use custom notifications, the title and body of the custom push notification may be saved to and processed by Google, Microsoft and/or Apple. Salesforce is not responsible for the privacy and security practices of third-party systems or applications like Google Cloud Messaging or Apple Push Notification Service.

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.

The screenshot shows the Salesforce Setup Apex Classes page. The URL is [https://**XXXXXXXXXX**.salesforce.com/setup/objects/apex/classes](#). The page title is "Apex Classes". It displays a list of 11 Apex classes, each with a "Edit" link and a "Delete" link. The columns include Name, Namespace Prefix, API Version, Status, Last Modified By, and Has Trace Flags. The classes listed are:

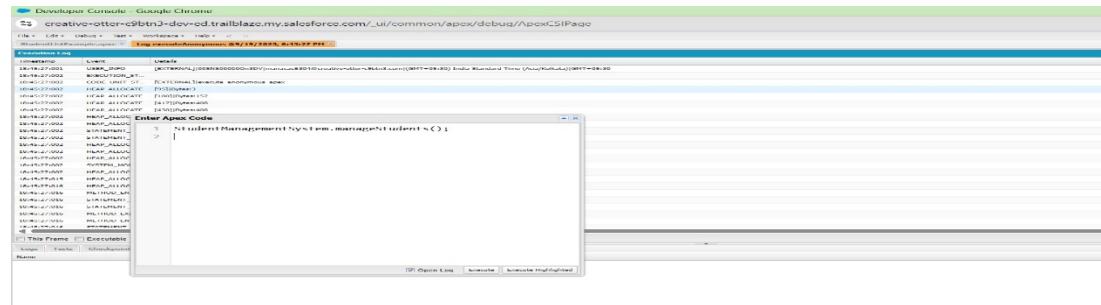
| Action | Name | Namespace Prefix | API Version | Status | Last Modified By | Has Trace Flags |
|-----------------------|-------------------------------|------------------|-------------|--------|------------------|-----------------|
| Edit Del Security | ScheduleUpdateStudentAgeBatch | | 54.0 | Active | 241 | |
| Edit Del Security | StudentInfoExample | | 54.0 | Active | 392 | |
| Edit Del Security | StudentManagementSystem | | 54.0 | Active | 1,472 | |
| Edit Del Security | StudentQueryService | | 54.0 | Active | 37 | |
| Edit Del Security | StudentService | | 54.0 | Active | 2,035 | |
| Edit Del Security | StudentServiceTest | | 54.0 | Active | 4,212 | |
| Edit Del Security | StudentTriggerHandler | | 54.0 | Active | 1,298 | |
| Edit Test Only | ILNHCController | ilnhtips | 54.0 | Active | 893 | |
| Edit Del Security | ILNHCControllerTest | ilnhtips | 54.0 | Active | 1,079 | |
| Edit Del Security | UpdateStudentAgeBatch | | 54.0 | Active | 991 | |
| Edit Del Security | UpdateStudentAgeCancellable | | 54.0 | Active | 863 | |

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.

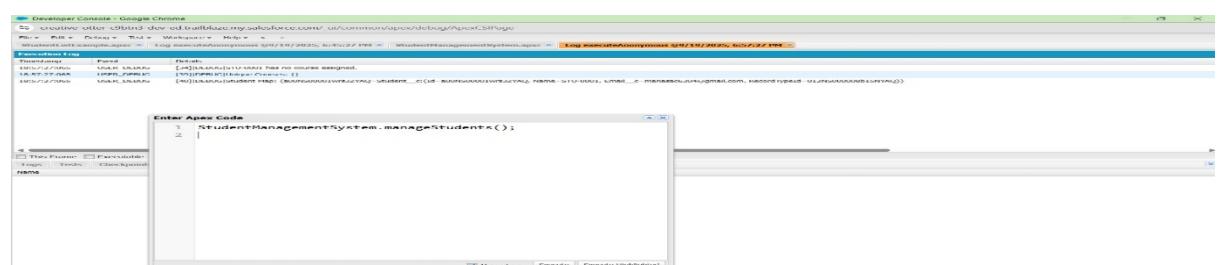


5. Collections (List, Set, Map)

- 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.

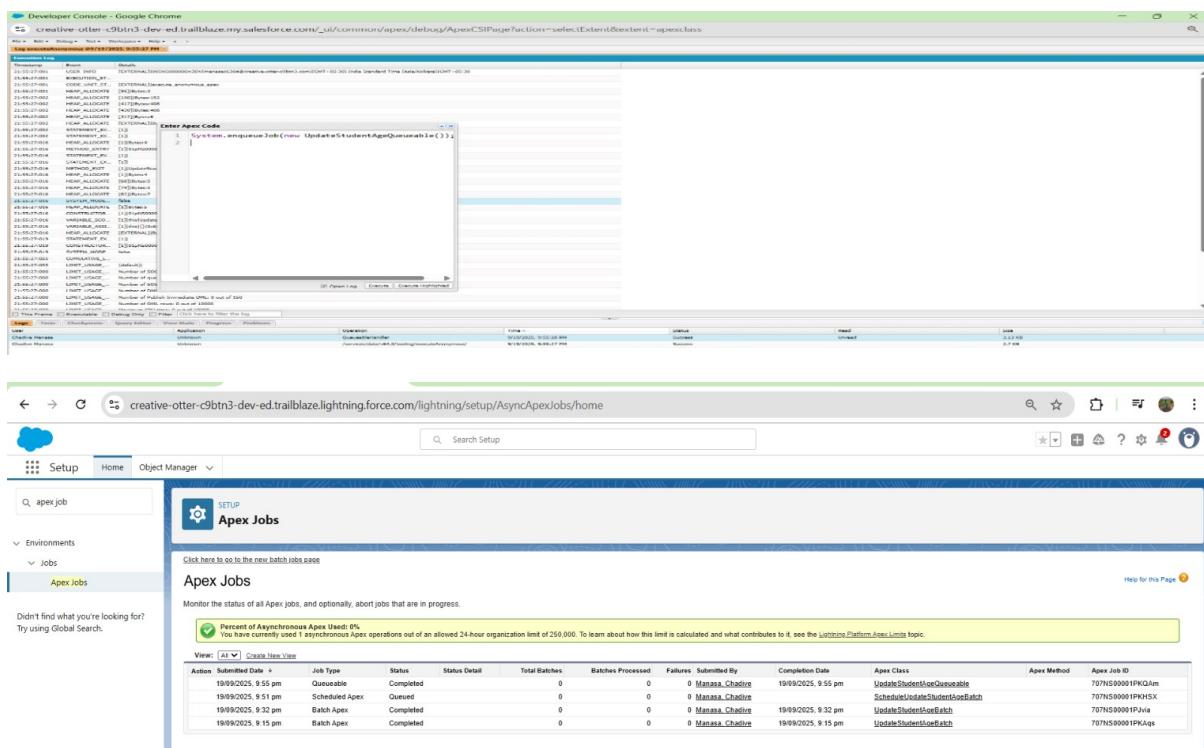


7. Batch Apex

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

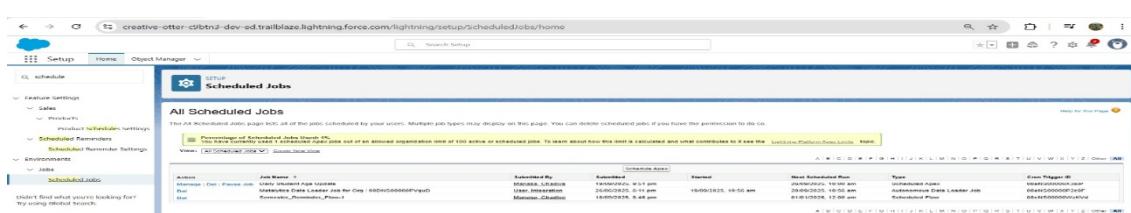
8. Queueable Apex

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



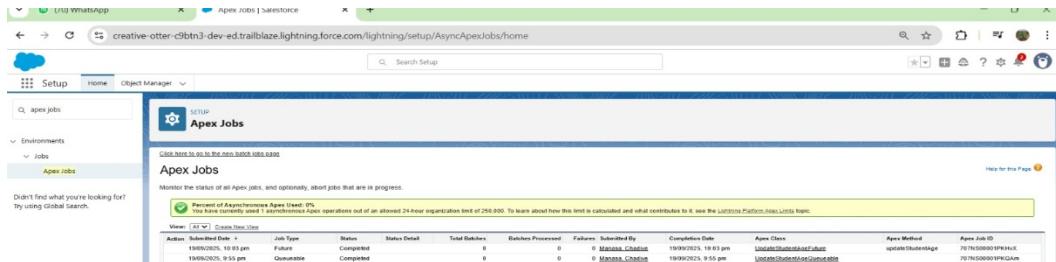
9. Scheduled Apex

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



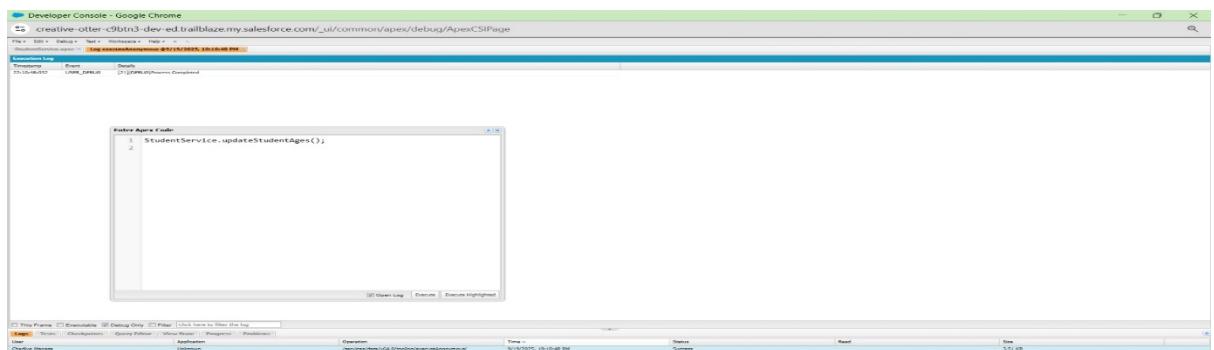
10. Future Methods

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



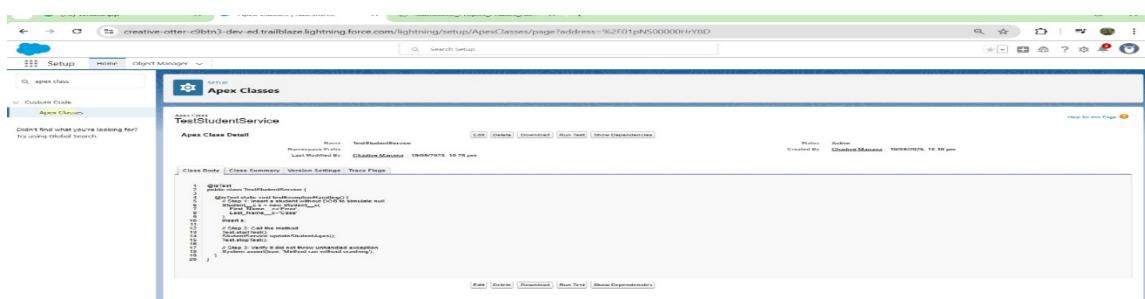
11. Exception Handling

- 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.

The screenshot shows the Salesforce Developer Console interface. The top navigation bar includes File, Edit, Debug, Test, Workflows, Help, and a user icon. Below the bar, the URL is `creative-otter-91tn3-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. The main area is divided into two sections: 'Execution Log' and 'Apex Jobs'.

Execution Log: This section displays a timeline of events with timestamps from 08:49:09:991 to 08:49:09:996. It shows various log entries such as EXECUTION_STARTED, CODE_UNIT_STARTED, EXTERNAL_TRIGGERED, and multiple HEAP_ALLOCATE entries. A modal window titled 'Enter Apex Code' is open, containing the following code:

```
1 UpdateStudentAgeBatch batchJob = new UpdateStudentAgeBatch();
2 Database.executeBatch(batchJob, 50); // batch size 50
```

Apex Jobs: This section lists three jobs under the 'Recent' tab. All three jobs are of type 'Batch Apex' and were run by 'Creative Heros'. The first job was successful and has a size of 3.02 kB. The second and third jobs are unread and have sizes of 3.48 kB and 3.08 kB respectively.

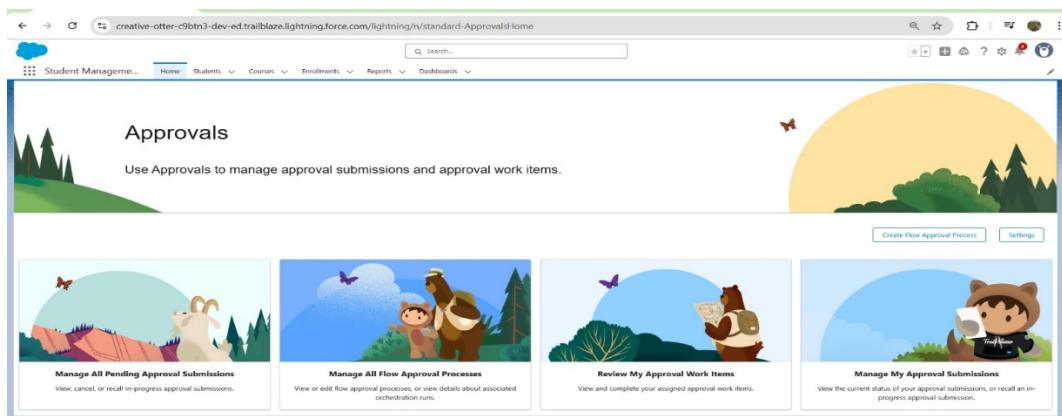
| User | Chickpeas | Query Editor | Value Store | Progress | Problems | Operation | Time | Status | Read | Size |
|----------------|-----------|--------------|-------------|----------|----------|---|-----------------------|---------|------|---------|
| Creative Heros | | | | | | /services/data/v44.0/testing/resourceAnonymous/ | 9/20/2020, 8:49:09 AM | Success | | 3.02 kB |
| Creative Heros | | | | | | Batch Apex | 9/20/2020, 8:49:09 AM | Success | | 3.48 kB |
| Creative Heros | | | | | | Batch Apex | 9/20/2020, 8:49:09 AM | Success | | 3.08 kB |

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.

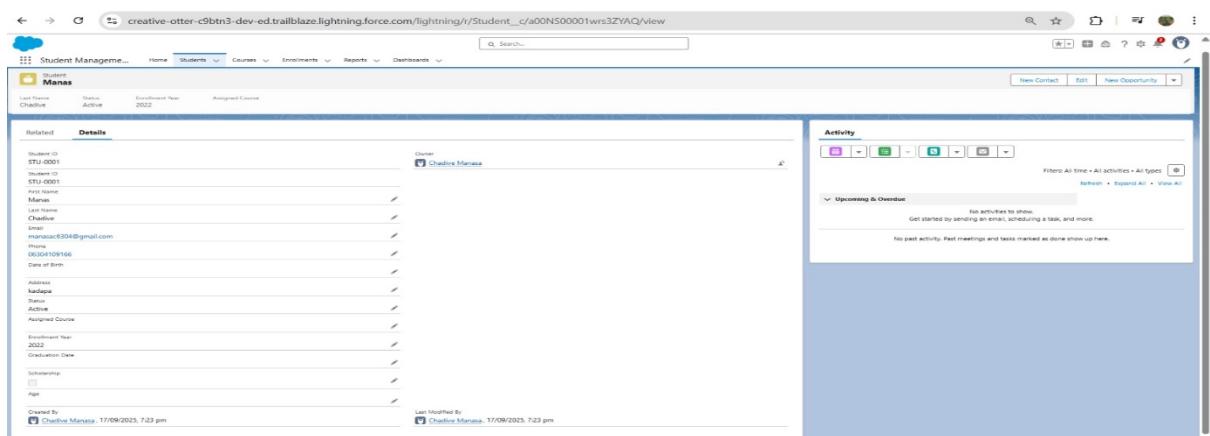
1.Lightning App Builder

- 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.



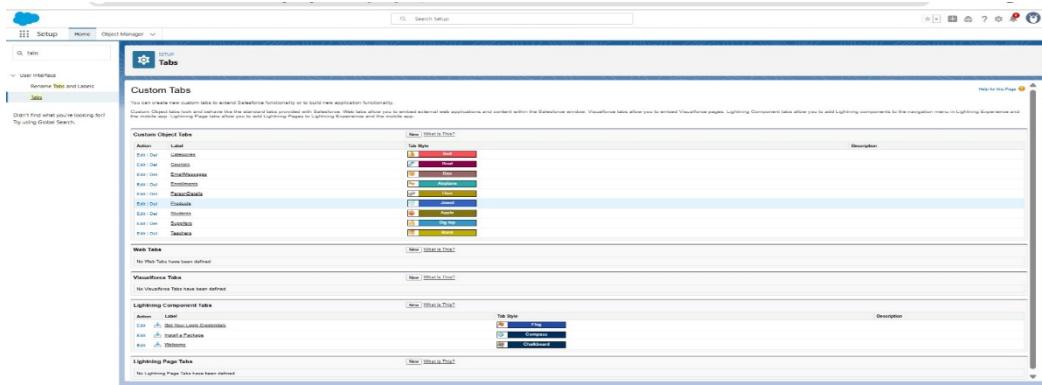
2.Record Pages

- 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.



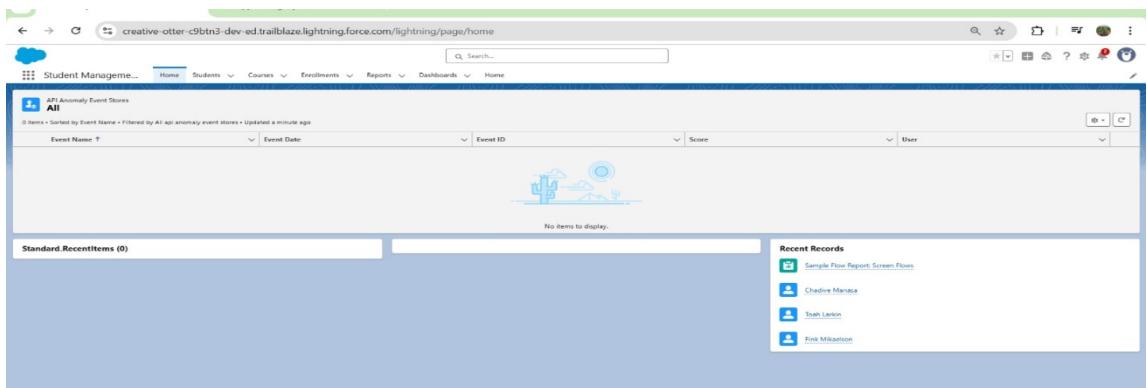
3.Tabs

- 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.



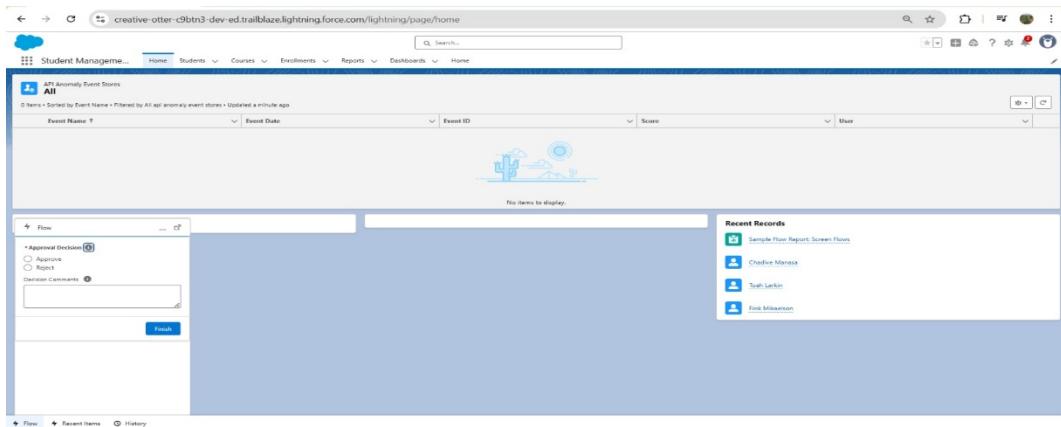
4. Home Page Layouts

- 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).

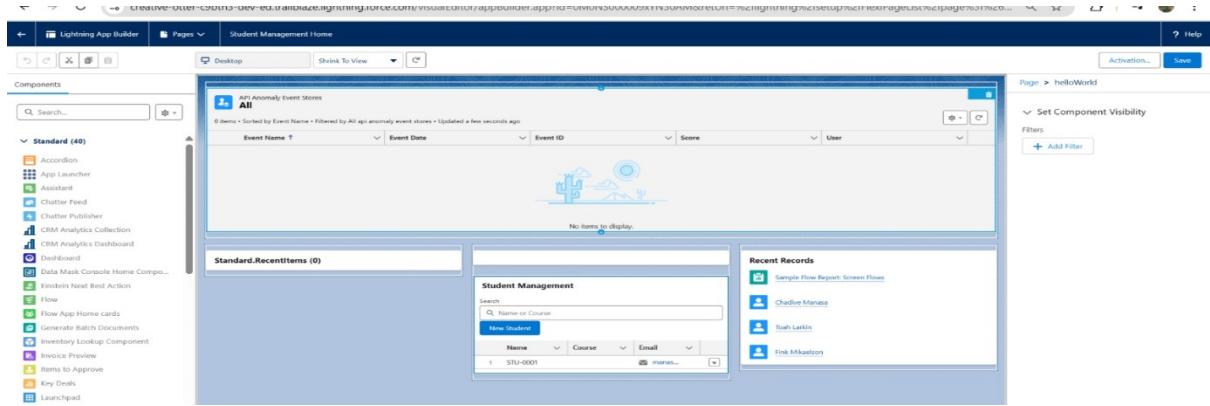
```

File Edit Selection View Go Run Terminal Help <- | > contactCreator
> _husky
> _sf
> _sfdx
> _vscode
> config
> force-app/main/default...
> applications
> aura
> classes
> contentsets
> flexipages
> layouts
> pages
> helloWorld
> helloWorld.html
> helloWorld.js
> helloWorld.js-meta.xml
> jsconfig.json
> objects
> permissions
> staticresources
> tabs
> triggers
> manifest
> scripts
> eslintrc.json
> .gitignore
> .prettierignore
> prettier
> eslint.config.js
> jest.config.js
> package.json
> README.md
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS HISTORY
manusar630@creative-otter-c9btn3.com using the v64.0
SOAP API
Preparing 250ms
  Waiting for the org to respond - skipped
  Deploying Meta Data 97ms
    Components: 1/1 (100%)
      Running Tests - Skipped
        Using Source Tracking - Skipped
      Done 0ms
cmd cmd
Status: Succeeded
In 42, Col 1  Spaces: 4  UTR: 8  CRLF  ⌂ HTML  ⌂ Prettier

```

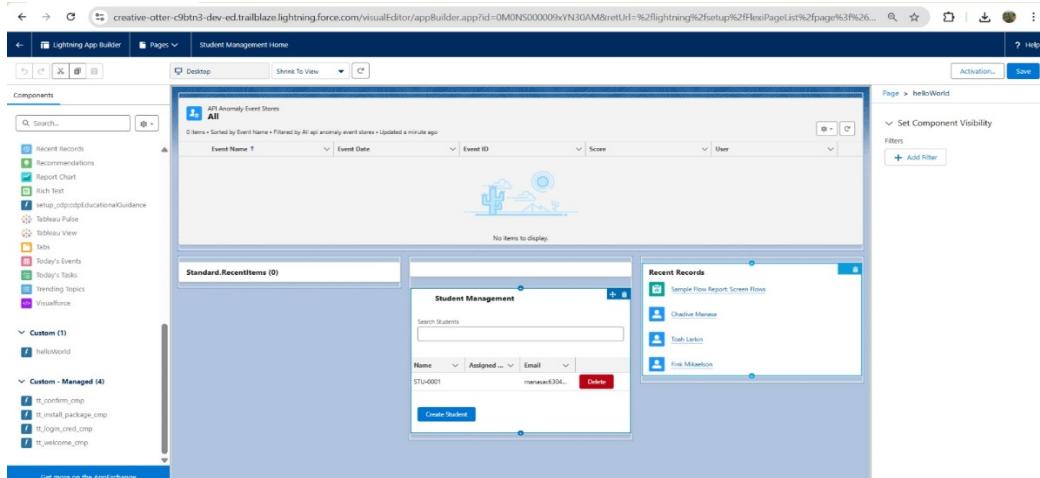
7. Apex with LWC

- 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;`

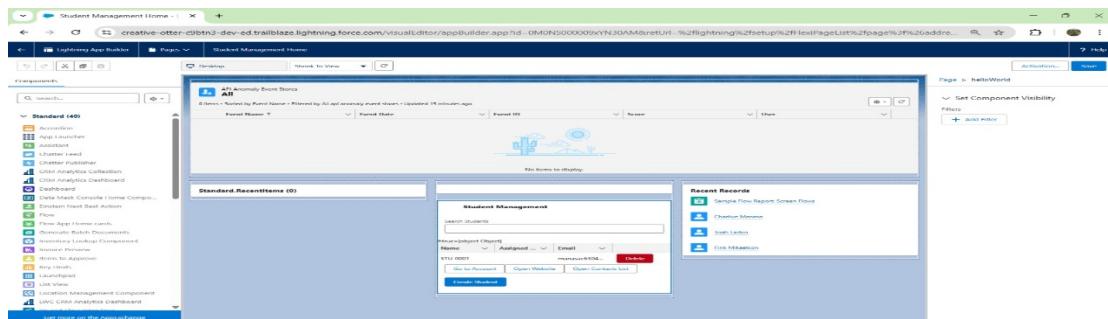
10. Imperative Apex Calls

- 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

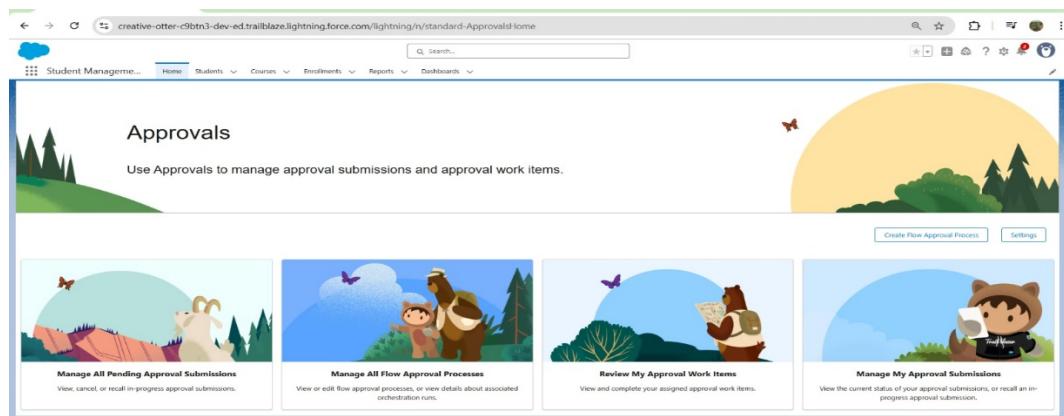
- **Purpose:** Programmatically navigate to records, lists, or external URLs.
- **Key Uses:** Navigate users to **Student details, Course list, or external websites**.
- **Details:** Use `NavigationMixin.Navigate()` inside LWC methods. Works on **App Pages and Record Pages**, but not standard Home Pages.

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.

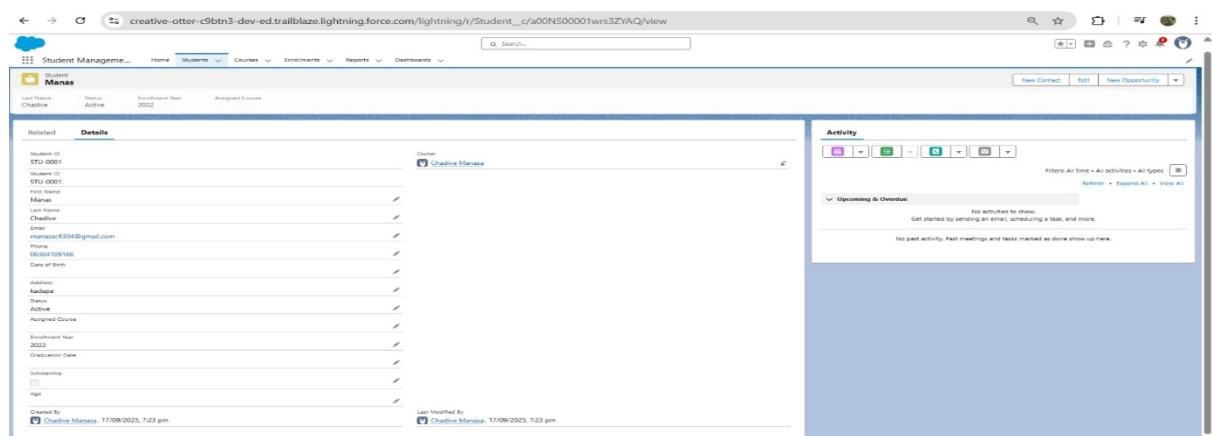
1.Lightning App Builder

- 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.



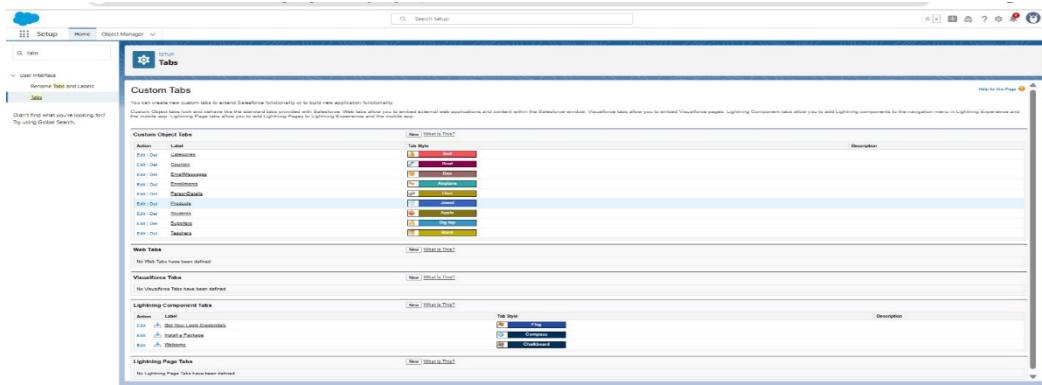
2.Record Pages

- 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.



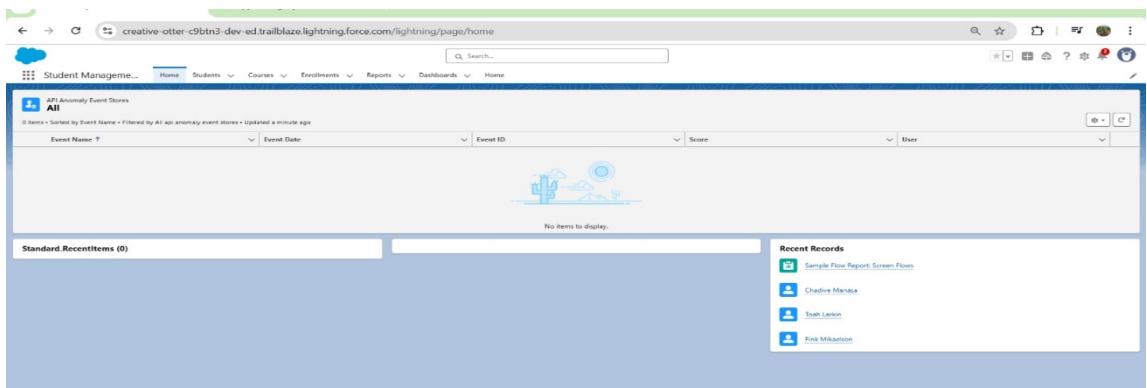
3.Tabs

- 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.



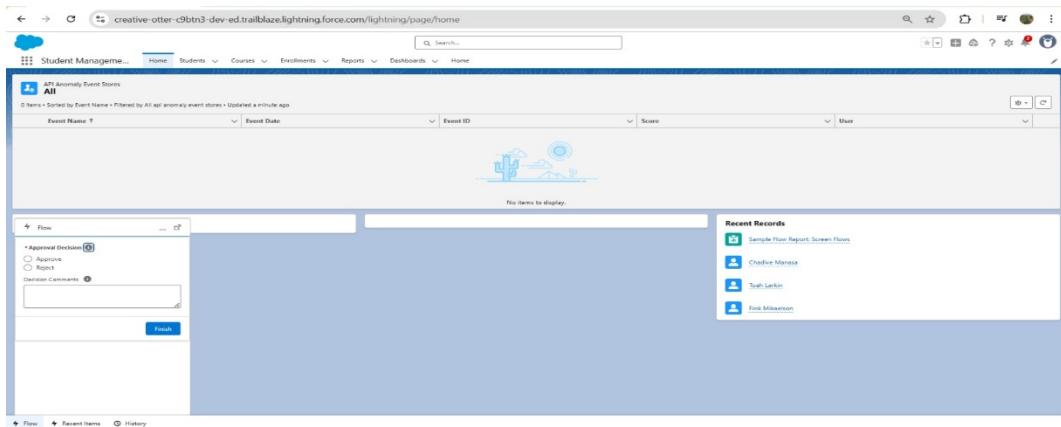
4. Home Page Layouts

- 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).

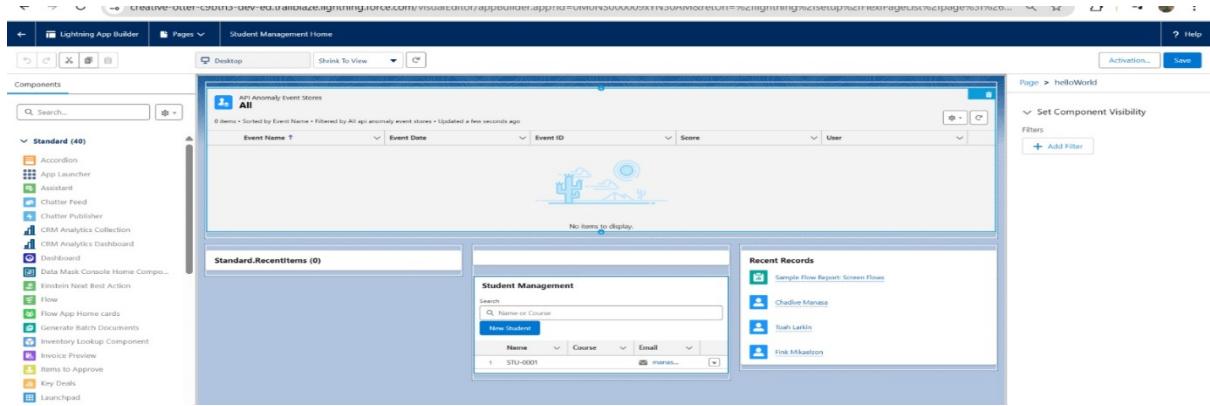
```

File Edit Selection View Go Run Terminal Help <- | > contactCreator
> _husky
> _sf
> _sfdx
> _vscode
> config
> force-app/main/default...
> applications
> aura
> classes
> contentsets
> flexipages
> layouts
> pages
> helloWorld
> helloWorld.html
> helloWorld.js
> helloWorld.js-meta.xml
> jsconfig.json
> objects
> permissions
> staticresources
> tabs
> triggers
> manifest
> scripts
> eslintrc.json
> .gitignore
> .prettierignore
> prettier
> eslint.config.js
> jest.config.js
> package.json
> README.md
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS HISTORY
manusar630@creative-otter-c9btn3.com using the v64.0
SOAP API
Preparing 250ms
  Waiting for the org to respond - skipped
  Deploying Meta Data 97ms
    Components: 1/1 (100%)
      Running Tests - Skipped
        Using Source Tracking - Skipped
      Done 0ms
cmd cmd
Status: Succeeded
In 42, Col 1  Spaces: 4  UTR: 8  CRLF  ⌂ HTML  ⌂ Prettier

```

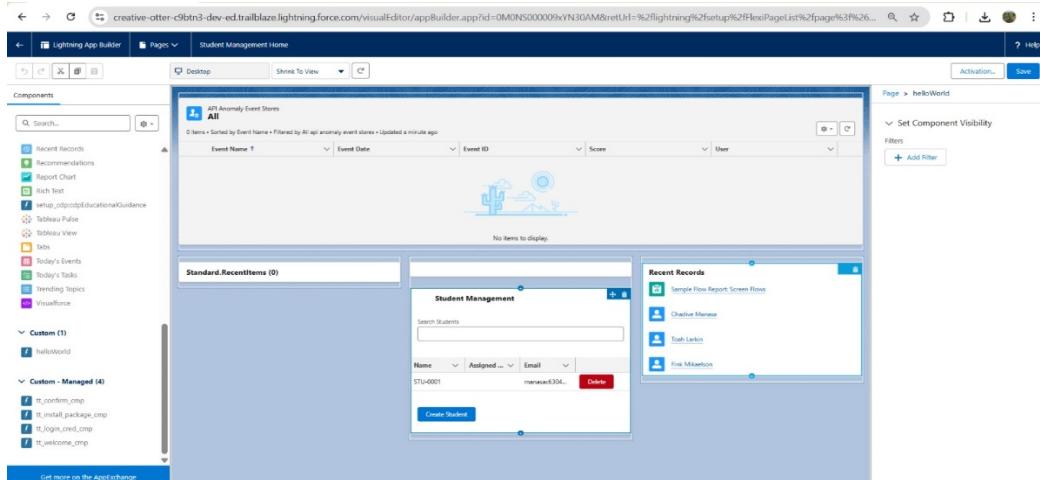
7. Apex with LWC

- 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;`

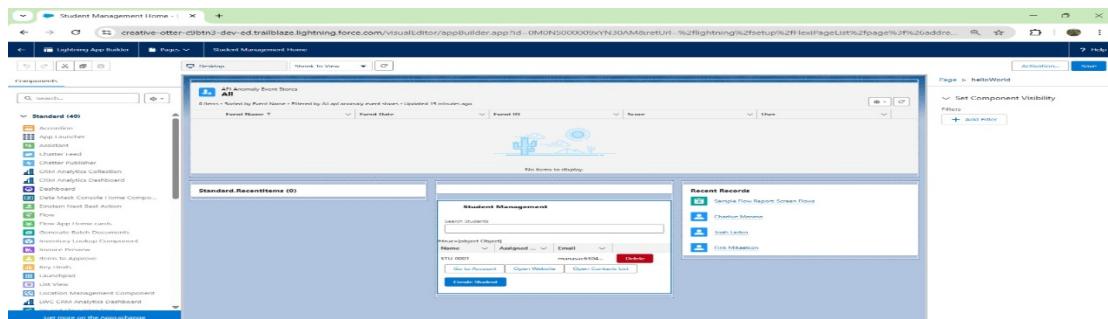
10. Imperative Apex Calls

- 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

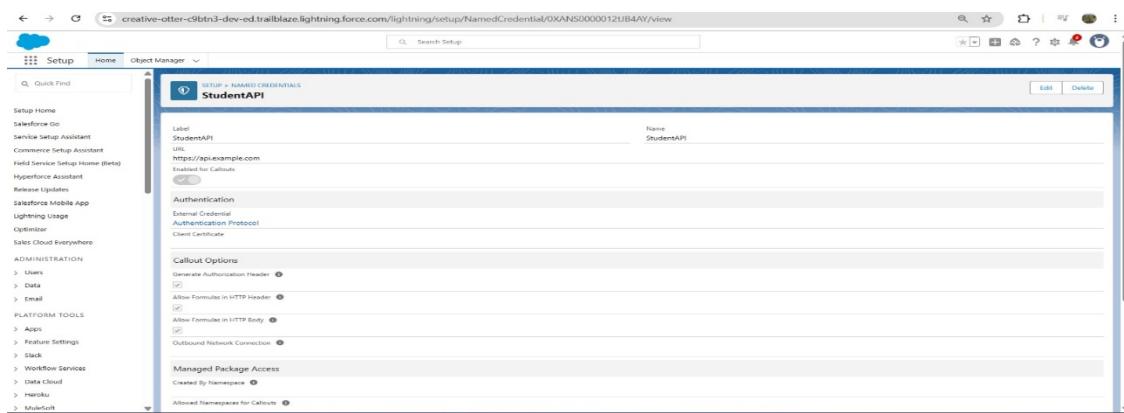
The primary goal of this phase was to enable **seamless data exchange and interaction** between Salesforce and external systems.

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

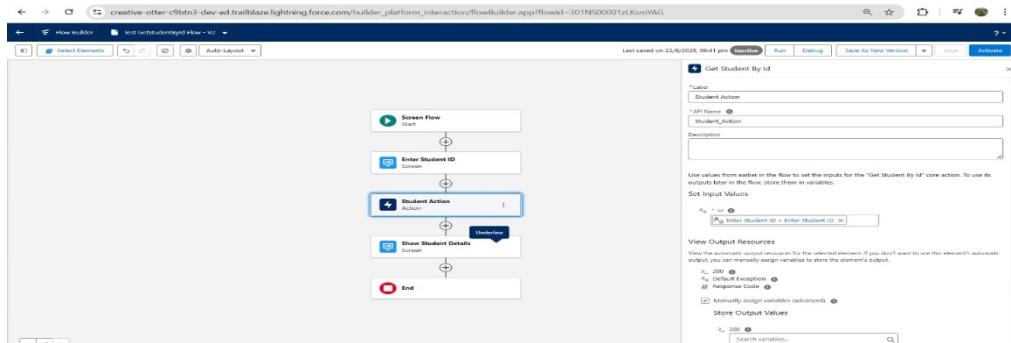
- **Created Named Credential:** Student_API_Named_Credential
- Configured with:
 - Base URL of the external Student API.
 - Authentication type (if required – basic or OAuth).
- Purpose: Simplified authentication and callout management without exposing credentials in Apex code.



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.

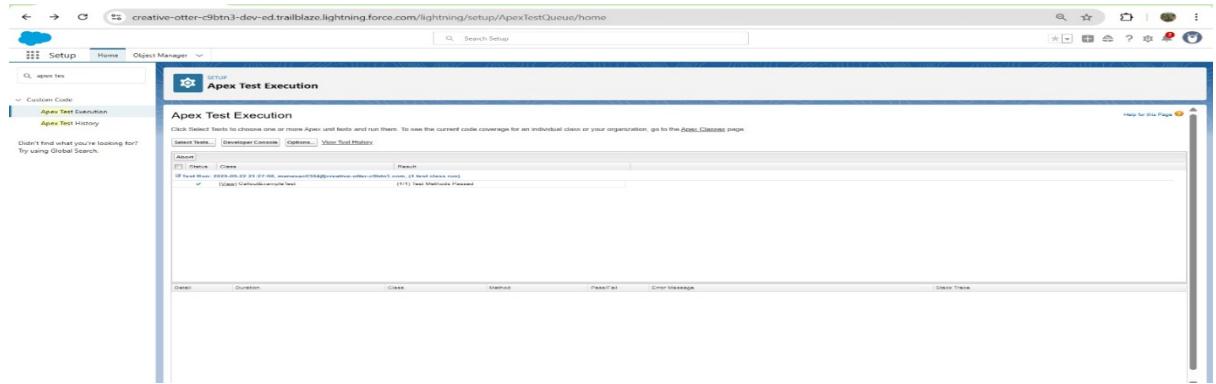


3. Web Services (REST/SOAP)

- 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.

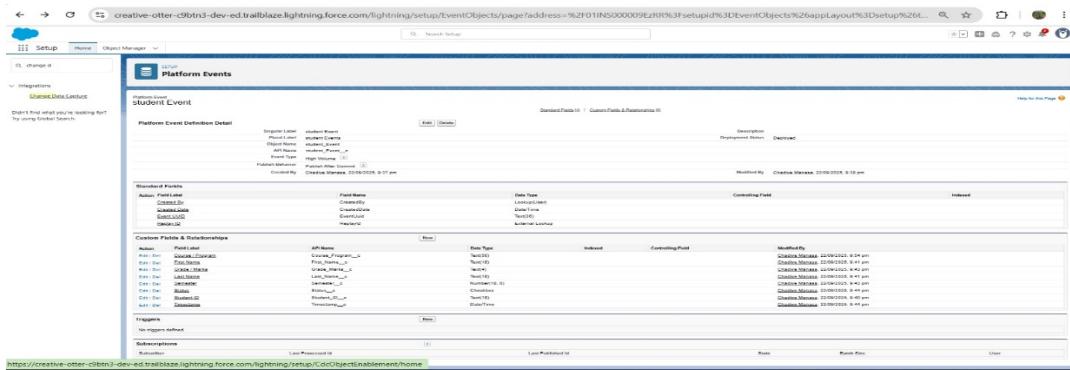
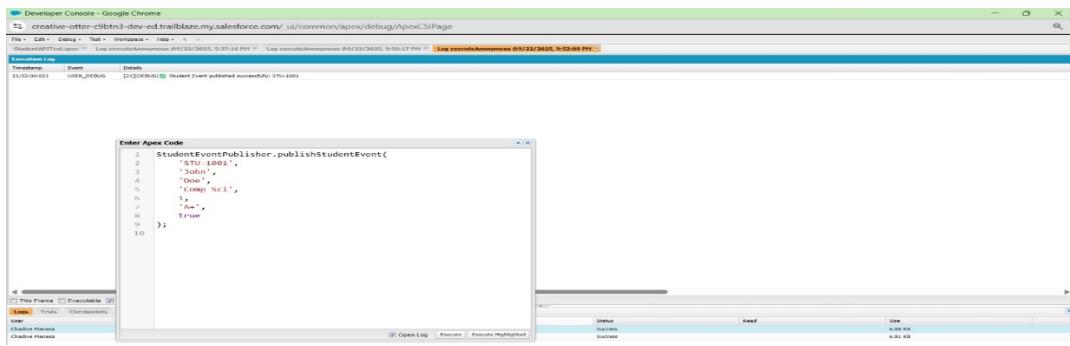
4. Callouts

- **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.



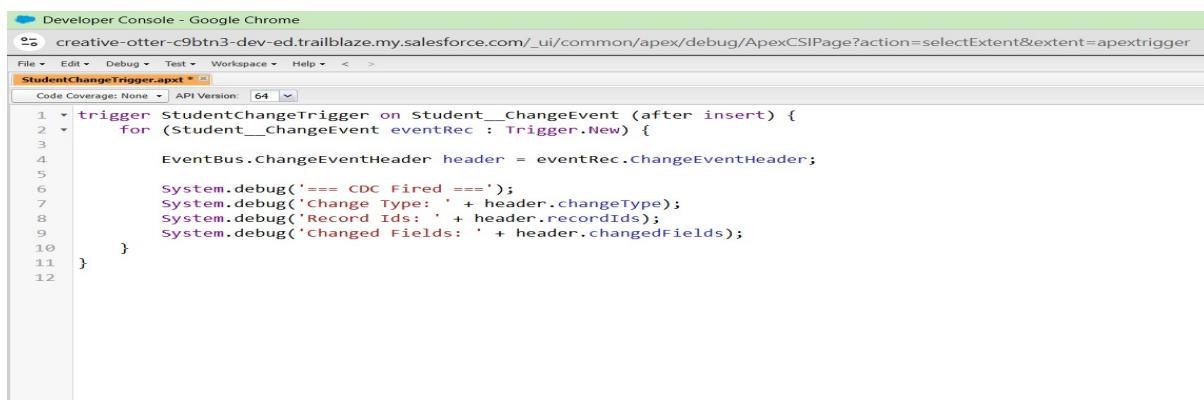
5. Platform Events

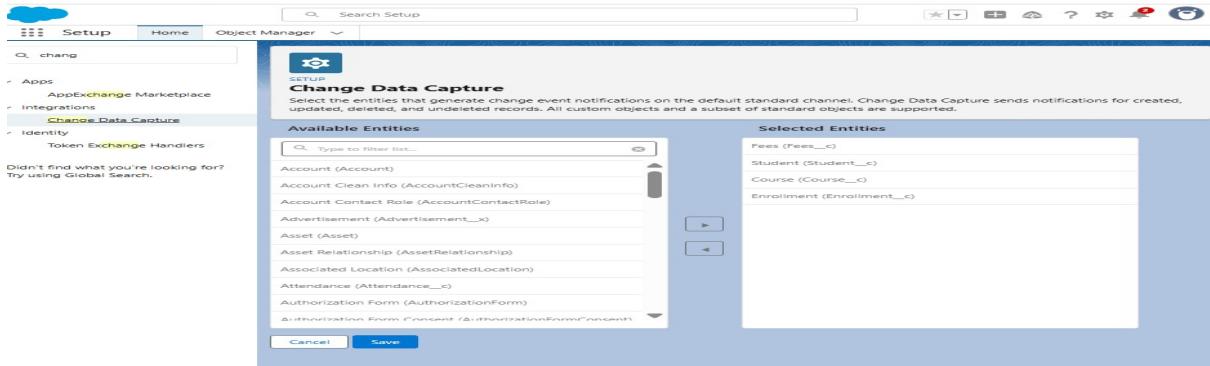
- **Created Platform Event:** Student_Event__e
- Custom Fields:
 - Student_ID__c (Text)
 - First_Name__c (Text)
 - Last_Name__c (Text)
 - Course_Program__c (Text)
 - Grade_Marks__c (Text)
 - Semester__c (Number)
 - Status__c (Checkbox)
 - Timestamp__c (Date/Time)
- Purpose: Publish student data changes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



6. Change Data Capture (CDC)

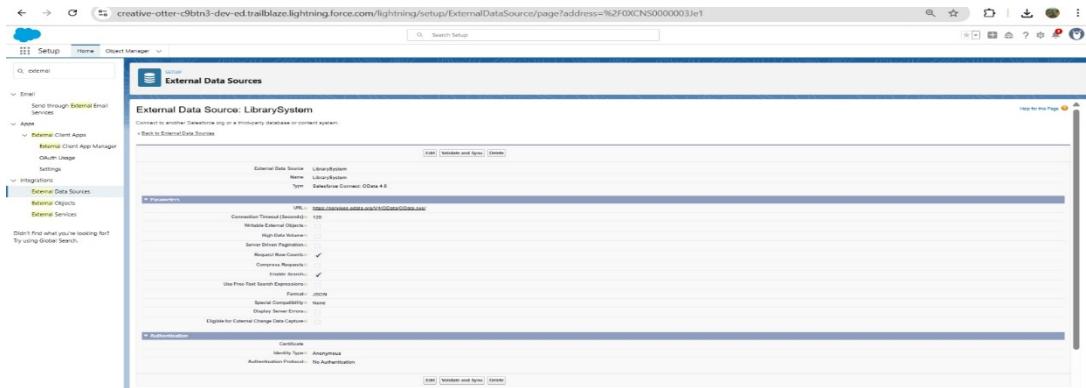
- **Enabled CDC for Student__c object**
 - **Created Apex Trigger:** StudentChangeTrigger
 - Logs change type, changed fields, and commit timestamp.
 - Takes action when Grade_Marks__c is updated.
 - Purpose: Monitor and respond to changes in real time without writing complex logic.





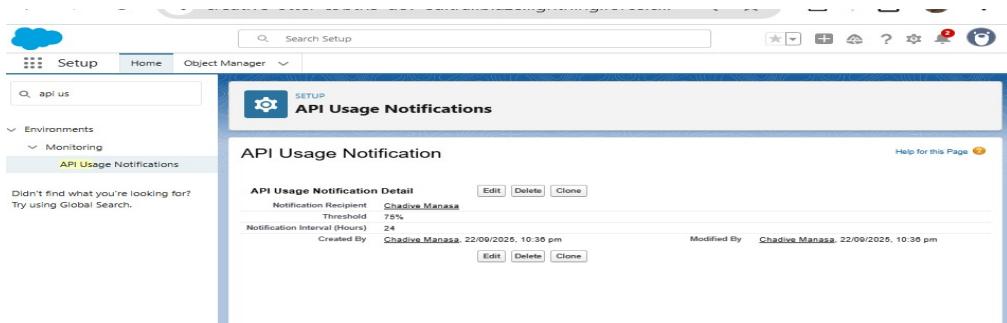
7. Salesforce Connect

- 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.

The screenshot shows the Salesforce Setup interface with the search bar set to "connect". Under the "Connected Apps" section, the "Connected Apps OAuth Usage" option is selected. The main content area displays the "Connected Apps OAuth Usage" page, which lists three connected apps: "Salesforce CLI", "Trailhead", and "tbid.digital.salesforce.com". Each entry includes a description, user count (1), and actions buttons for "Block" and "Install".

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.

The screenshot shows the Developer Console with an Apex class named "CalloutExample". The code implements a static method "makeCallout()" that sends a POST request to a specified endpoint using the HTTP class. It checks the response status code and returns the response body if successful, or throws an exception if it fails.

```
public static String makeCallout() {
    String endpoint = 'https://apitest.salesforce.com/500';
    CalloutExample.setEndpoint(endpoint);
    RestCallout callout = new RestCallout();
    callout.setEndpoint(endpoint);
    callout.setMethod('POST');
    callout.setHeader('Content-Type', 'application/json');
    callout.setBody('{"name": "My Name"}');
    HttpResponse res = callout.send();
    if(res.getStatusCode() == 200) {
        return res.getBody();
    } else {
        throw new CalloutException('Error: ' + res.getStatusCode());
    }
}
```

The screenshot shows the Salesforce Setup interface with the search bar set to "remote". Under the "Custom Code" section, the "Remote Site Settings" option is selected. The main content area displays the "Remote Site Settings" page, which lists "All Remote Sites". It shows two entries: "ApexDevNet" (Active, Namespace Prefix: -, Remote Site URL: http://www.apexdevnet.com, Created By: Manasa Chavade, Created Date: 26/06/2025, 6:09 pm) and "Student API" (Active, Namespace Prefix: -, Remote Site URL: https://ap.example.com, Created By: Manasa Chavade, Created Date: 26/06/2025, 6:20 pm).

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.

The screenshot shows the Bulk Data Load Jobs page in the Salesforce Setup interface. It displays a list of jobs, with one job highlighted: '7600000000WFSH'. The job details show it was submitted by 'Creative Otter' at '2020/02/25, 1:57 pm EST' and completed at '2020/02/25, 1:57 pm EST'. The job status is 'Completed' with a total of 105 records processed. The 'Batches' section shows a single batch with 105 records processed. The left sidebar includes links for Setup Home, Object Manager, and various system administration and platform tools.

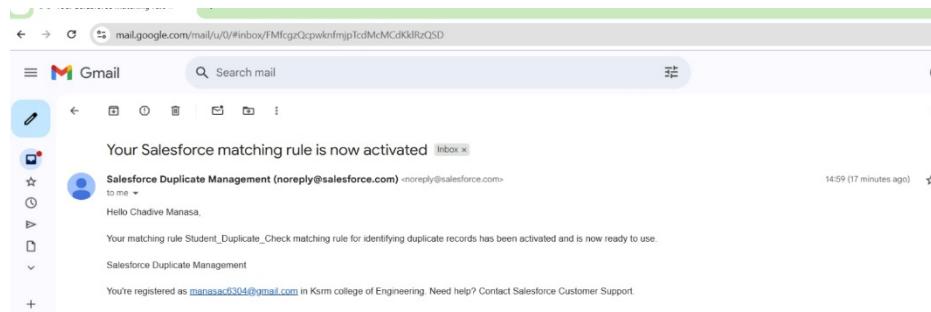
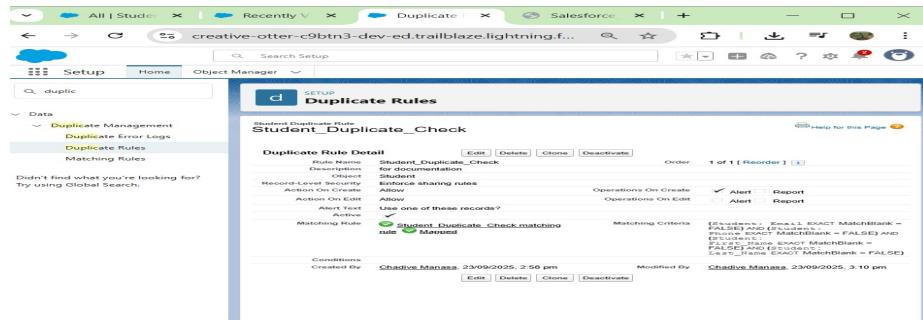
2. Data Loader

- 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.

The screenshot shows the Salesforce Data Loader interface. A modal window titled 'Step 3: Mapping' is open, prompting the user to 'Choose an Existing Map' or 'Create or Edit a Map'. The mapping table lists columns from a CSV file ('Name', 'Email__c', 'Phone__c', etc.) and their corresponding Salesforce object field names ('Email__c', 'Phone__c', etc.). The main Data Loader window shows a list of student records with actions like Insert, Update, Upsert, Delete, and Undelete. The top navigation bar indicates the session is for 'Salesforce_Project_P'.

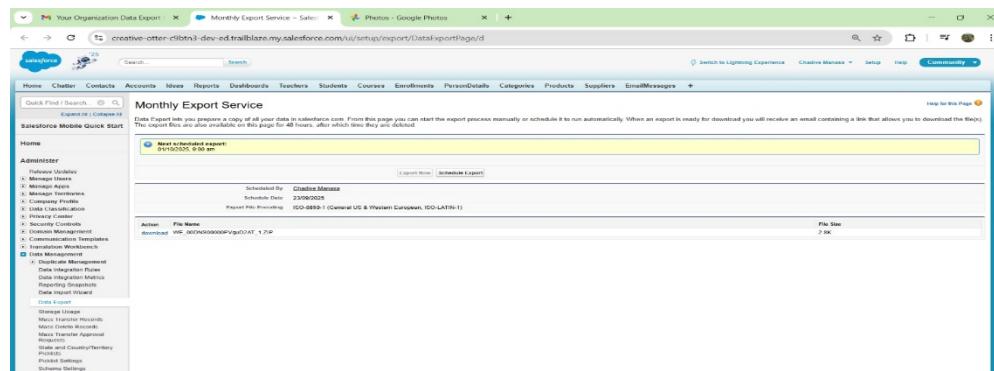
3. Duplicate Rules

- 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.

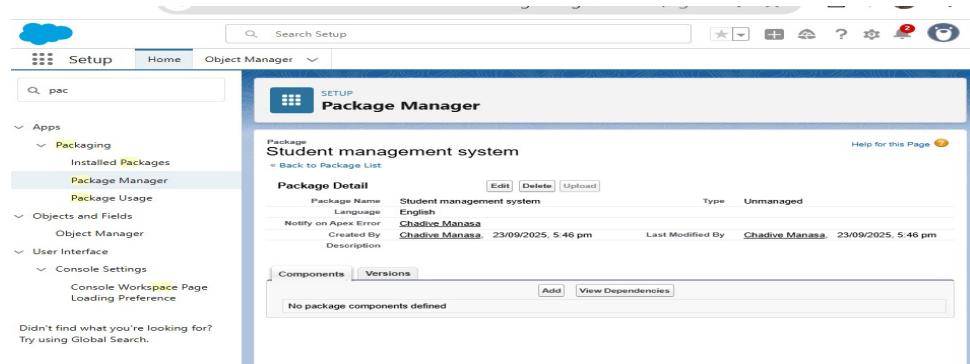


5. Change Sets

- 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.

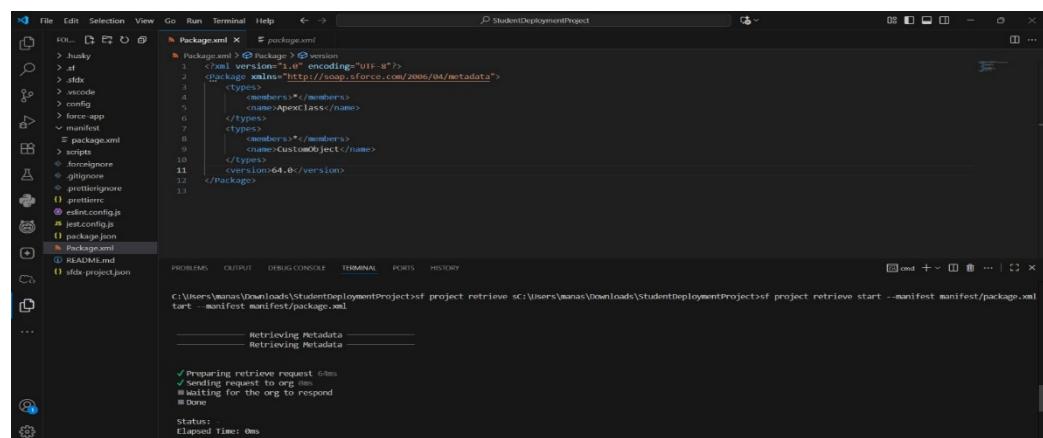


7. ANT Migration Tool

- 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

| Report Name | Description | Folder | Created By | Created On | Subscribed |
|---|---|----------------|-------------------|--------------------|------------|
| New Courses Report | | Art One | Creative Manresa | 23/6/2025, 4:00 pm | |
| New Students Report | | Art One | Creative Manresa | 23/6/2025, 4:01 pm | |
| Sample Prime Report: Student Plans | Which Brown Bear, what's the status of each item? How long do users take to complete the items? | Public Reports | Automated Process | 23/6/2025, 4:09 pm | |
| Sample Report: Orchestration Run Logs | What orchestration run logs were created and what's the current status of their associated orchestration run? | Public Reports | Automated Process | 23/6/2025, 4:09 pm | |
| Sample Report: Orchestration Run Runtimes | What orchestration run runtime logs were created and what's the current status of each run? | Public Reports | Automated Process | 23/6/2025, 4:09 pm | |
| Sample Report: Orchestration Stage Run | What orchestration stage runs have been created and what's the current status of each run? | Public Reports | Automated Process | 23/6/2025, 4:09 pm | |
| Sample Report: Orchestration Step Run | What orchestration step runs have been created and what's the current status of each run? | Public Reports | Automated Process | 23/6/2025, 4:09 pm | |
| Sample Report: Orchestration Work Items | What orchestration work items were created and what's the current status of each work item? | Public Reports | Automated Process | 23/6/2025, 4:09 pm | |

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

The screenshot shows the 'Student Attendance Report' configuration page in the Salesforce setup area. The left sidebar is titled 'Q_Report' and includes sections for Feature Settings, Analytics, Reports & Dashboards, Security, and User Sharing Rule Access Report. The main content area is titled 'Student Attendance Report' and contains two main sections: 'Details' and 'Object Relationships'. The 'Details' section shows the report's display label ('Student Attendance Report'), API name ('Student_Attendance_Report'), description ('Add a meaningful description'), created by ('Chaitali Manasa, 9/23/25, 6:26 PM'), store in category ('Other'), deployment status ('Deployed'), and modified by ('Chaitali Manasa, 9/23/25, 6:26 PM'). The 'Object Relationships' section shows a Venn diagram where 'Students (A)' overlap with 'Courses (B)', indicating a many-to-many relationship. Below these sections are tables for 'Fields' and 'Source Object'.

The screenshot shows the 'Attendance Report Type' configuration page in the Salesforce setup area. The left sidebar is titled 'Q_Report' and includes sections for Analytics, Reports & Dashboards, Security, and User Sharing Rule Access Report. The main content area is titled 'Attendance Report Type' and contains two main sections: 'Details' and 'Object Relationships'. The 'Details' section shows the report's display label ('Attendance_Report_Type'), API name ('Attendance_Report_Type'), description ('Attendance_Report_Type'), created by ('Chaitali Manasa, 9/23/25, 6:26 PM'), store in category ('Other'), deployment status ('Deployed'), and modified by ('Chaitali Manasa, 9/23/25, 6:26 PM'). The 'Object Relationships' section shows a Venn diagram where 'Students (A)' overlap with 'Attendances (B)', indicating a many-to-many relationship. Below these sections are tables for 'Fields' and 'Source Object'.

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

The screenshot shows the 'Student Management Overview' dashboard in the Salesforce interface. The top navigation bar includes links for Home, Students, Courses, Enrollments, Reports, Dashboards, and Home. The main dashboard area features several reports and charts. On the left, there is a 'New Students Report' table showing student IDs (STU-0001 through STU-0007) and a link to 'View Report (New Students Report)'. Below it is a 'New Courses Report' table showing course names (Mathematics) and a link to 'View Report (New Creation Report)'. The right side of the dashboard is a large, empty grid table with columns for Student ID, Course Name, Grade, and Enrollment Status.

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.

The screenshot shows a Salesforce dashboard titled "Dynamic Student Dashboard". At the top, there's a search bar and a navigation bar with links for Home, Students, Courses, Enrollments, Reports, Dashboards, and Home. Below the header, it says "As of 23-Sep-2025, 6:37 pm Viewing as Charlie Manana". The main content area is titled "New Students Report" and contains a list of student IDs: STU-0001, STU-0002, STU-0003, STU-0004, STU-0005, STU-0006, and STU-0007. At the bottom left is a link "View Report (New Students Report)" and at the bottom right is the timestamp "As of 23-Sep-2025, 6:37 pm".

5. Sharing Settings

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

The screenshot shows the "Sharing Settings" page in the Salesforce Setup interface. The left sidebar lists categories like Security, Sharing Settings, and various object types. The main content area displays a grid of sharing rules for different objects. For most objects, the sharing rule is "Private". For "Public Read Only" objects, the sharing rule is "Public Read Only". Under "Controlled by Parent", objects like "Course" and "Grade" have "Controlled by Parent" as the sharing rule. Other objects like "Attendance" and "Work Plan" also have specific sharing rules defined.

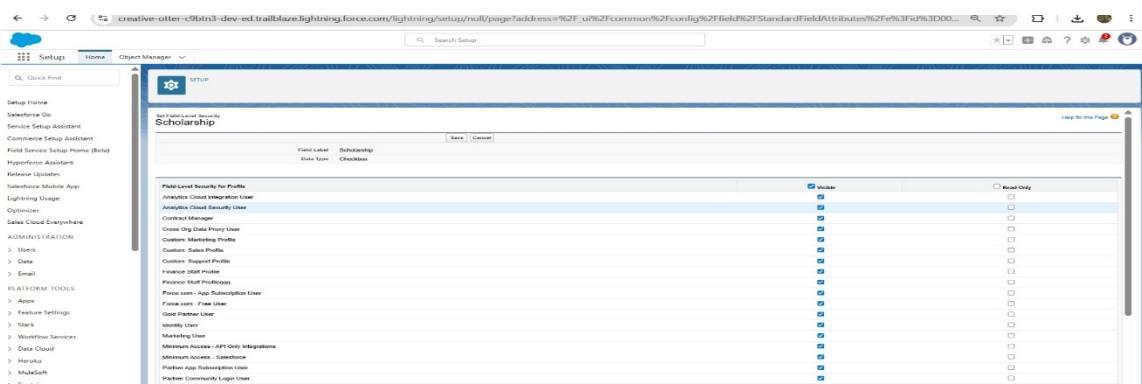
This screenshot shows a continuation of the "Sharing Settings" page from the previous image. It includes several more sections of sharing rules:

- Work Plan Template Sharing Rules:** No sharing rule specified.
- Work Step Template Sharing Rules:** No sharing rule specified.
- Attendance Sharing Rules:** Action: Delete; Owner: User or All Internal Users; Shared With: Role_Teacher; Access Level: ReadWrite.
- Course Sharing Rules:** Action: Delete; Owner: User or All Internal Users; Shared With: Role_Teacher; Access Level: ReadWrite.
- EmailMessage Sharing Rules:** No sharing rule specified.
- Fee Sharing Rules:** Action: Delete; Owner: User or All Internal Users; Shared With: Role_LookUp; Access Level: ReadWrite.
- Grades Sharing Rules:** Action: Create; Owner: User or All Internal Users; Shared With: Role_Teacher; Access Level: ReadWrite.
- Student Sharing Rules:** Action: Delete; Owner: User or All Internal Users; Shared With: All Internal Users; Access Level: Read Only.

- **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)

The screenshot shows the 'Trusted IP Range Edit' page under 'Network Access'. The page title is 'Trusted IP Range Edit'. It contains a form with fields for 'Start IP Address' (set to 192.168.0.1) and 'End IP Address' (set to 192.168.0.255). There is also a 'Description' field which is empty. At the bottom are 'Save' and 'Cancel' buttons.

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

The screenshot shows the 'View Setup Audit Trail' page. It lists 20 audit events for the last 30 months. The first event is: 'Added 10 IP address from 192.168.0.1 to 192.168.0.255'. Other events include: 'Changed welcome email template from "Welcome Student" to "Welcome Student" (Email)', 'Changed security level from "Standard" to "High" (Security)', 'Changed profile "Student User" - Associated security for Student Scholarship was changed from 0 to 2', 'Changed profile "Student User" - Associated security for Student Scholarship was changed from 2 to 0', 'Changed profile "Parent Community User" - Associated security for Student Scholarship was changed from 0 to 2', 'Changed profile "Parent Community User" - Associated security for Student Scholarship was changed from 2 to 0', 'Deleted contact "Jeff Sheldakoff" from Students', 'Created "Jeff Sheldakoff" from Students', 'Created "Jeff Sheldakoff" from Students', 'Created Student Owner Sharing Rule New Sharing Rule', 'Deleted Owner Role "Student" record creation: New Sharing Rule', 'For the 018-00000004040-duplicate rule Student_Duplicate__c, changed "Opportunities On Create" from "None" to "Allow New Reportable"', 'For the 018-00000004040-duplicate rule Student_Duplicate__c, changed "Opportunities On Create" from "Allow New Reportable" to "None"', 'For the 018-00000004040-duplicate rule Student_Duplicate__c, changed "Opportunities On Create" from "Allow New Reportable" to "None" (Reportable)', 'For the 018-00000004040-duplicate rule Student_Duplicate__c, checked "Check for Existing Record" from "Allow" to "Block"', 'For the 018-00000004040-duplicate rule Student_Duplicate__c, checked "Check for Existing Record" from "Allow" to "Block"', 'Student mailing rule: Student_MailingRule_Check mailing rule, activated by Creative Mantra', 'For mailing rule Student_MailingRule_Check mailing rule, added matching criteria where matching method is Exact, the field is Last_Name and match blank fields is "Does Not Match If Null"', 'For mailing rule Student_MailingRule_Check mailing rule, added matching criteria where matching method is Exact, the field is First_Name and match blank fields is "Does Not Match If Null"', 'For mailing rule Student_MailingRule_Check mailing rule, added matching criteria where matching method is Exact, the field is Middle_Name and match blank fields is "Does Not Match If Null"', 'For mailing rule Student_MailingRule_Check mailing rule, added matching criteria where matching method is Exact, the field is Last_Name and match blank fields is "Does Not Match If Null"', 'For mailing rule Student_MailingRule_Check mailing rule, added matching criteria where matching method is Exact, the field is First_Name and match blank fields is "Does Not Match If Null"', 'For mailing rule Student_MailingRule_Check mailing rule, added matching criteria where matching method is Exact, the field is Middle_Name and match blank fields is "Does Not Match If Null"'. At the bottom is a note: 'Downloadable audit trail for the last 30 months (Excel .csv file)'.

The screenshot shows the 'Student Field History' setup page. It lists fields for tracking: First Name, Last Name, Phone, Scholarship, and Student ID. At the bottom are 'Save' and 'Cancel' buttons.

