Library Management Phase 10: Final Presentation & Demo Day

Objective:

Demonstrate the end-to-end functionality of the **Library Management ,** highlight outcomes, and provide documentation for teachers, students, librarians, and library admin.

---

1. Project Pitch

Goal: Showcase how Smart Library CRM simplifies resource tracking, improves student–teacher interaction, and increases library efficiency.

Key Points to Present:

1. Streamlined student/teacher record management.
2. Automated book issue/return tracking & fine calculation.
3. Real-time updates for teachers, students, and librarians.
4. Analytics through reports and dashboards (usage trends, overdue items, popular books).

---

2. Demo Flow (Click-by-Click Navigation)

Step 1 – Login & Access App

- Open Salesforce → Go to **Library Management Lightning App**.

Step 2 – Add/Manage Student or Teacher

- Navigate to **Students** or **Teachers** Tab → **New**.
- Fill in details (Name, DOB, Contact, Class/Department).
- Save → System applies any automation (e.g., auto-assign class or default library card number).

Step 3 – Book Selection / Issue

- Navigate to **Books** Tab → Select Book.
- Create **Borrow Request** by selecting book, issue date, and due date.

Step 4 – Return & Fine Handling

- Navigate to **Borrow Requests** Tab → Mark as **Returned**.

- System auto-calculates Fine Amount (if overdue).
- Librarian records payment of fine if applicable.

Step 5 – Notifications & Feedback

- Confirm book issue/return → Student receives notification (demo simulated without actual emails).
- Navigate to **Feedback** Tab → **New Feedback** to record user feedback on library services.

Step 6 – Dashboard Update

- Navigate to **Dashboards** → **Library Overview Dashboard**.
- Verify that new borrow requests, returns, fines, and feedback appear correctly.

---

3. Handoff Documentation

- **Setup Guide:** Step-by-step instructions to deploy the Library CRM in Salesforce Org.
- **User Manuals:**
    - **Librarians:** How to manage books, issues, returns, and fines.
    - **Teachers/Students:** How their borrow requests, returns, and feedback are managed in the system.

---

4. LinkedIn Showcase

- Tag the project as **Smart Library Management** .
- Highlight Salesforce implementation skills, including:
    - Custom Objects & Relationships (Books, Borrow Requests, Students, Teachers).
    - Apex Triggers (auto-calculate fines, send reminders).
    - Reports & Dashboards (overdue books, most borrowed resources).
    - Data Management & Deployment (imports, backups, permissions).

PHASE 1 Project Title: Library Management

Problem Statement

Managing and tracking books can become cumbersome when done manually or on paper. Small libraries, students, or hobbyists often need a lightweight system to record book details, view them quickly, and make edits without installing heavy software or maintaining a server. The Simple Library Management website addresses this by offering a browser-based CRUD tool using only HTML, CSS, and JavaScript with localStorage.

Requirement Gathering

- Identify what core actions the user needs (add, view, edit, delete books).
- Decide which fields are mandatory (Title, Author, Year, ISBN).
- Determine storage mechanism (localStorage vs. server).
- Collect UI expectations (form + table layout, buttons).
- Plan file structure (index.html, styles.css, script.js).
- Document non-functional needs (works offline, simple design).

Stakeholder Analysis

- Define primary users (students, small libraries, demo users).
- Identify secondary users (developers learning CRUD apps).
- Note stakeholders' technical skills (beginner-friendly, no server).
- Capture feedback on desired features (extra fields, pop-ups).
- Understand constraints (browser-only, small data sets).
- Set expectations for future enhancements (server API, multi-user).

Business Process Mapping

- Map the flow of adding a book (form input → JS → localStorage → table).

- Describe how data persists after refresh (JSON in localStorage).
- Show the edit flow (click edit → prompt/update → save back).
- Map the delete flow (click delete → confirm → remove from storage).
- Define initial data state (empty table).
- Document error handling or confirmations (validation, clearing storage).

Industry-Specific Use Case Analysis

- Compare with small-scale library or personal book lists.
- Explore educational uses (teaching CRUD in classes).
- Consider adaptation for inventory or asset tracking.
- Assess scalability limits (localStorage size, single user).
- Identify privacy and data safety considerations (local only).
- Evaluate user interface needs across devices (desktop, mobile).

AppExchange / Expansion Exploration

- Plan how to extend fields (Genre, Publisher, Ratings).
- Replace prompts with inline edit forms or modals.
- Investigate moving from localStorage to a simple backend or API.
- Consider packaging as a small open-source app or template.
- Experiment with new UI styles (Tailwind, Bootstrap, animations).
- Explore publishing on platforms like Salesforce AppExchange or GitHub Pages as a free tool.

PHASE 2: Org Setup & Configuration

1. Introduction

This phase sets up a Salesforce Developer Org to model a small library for cataloguing, lending, and returns. It builds on Phase 1 by configuring book records, librarian/member roles, and security so the system reflects real library workflows for adding, editing, deleting, and viewing books.

2. Objective

Create a clean Salesforce Developer Org named “Library Connect” with sample users (Head Librarian, Librarian, Member).

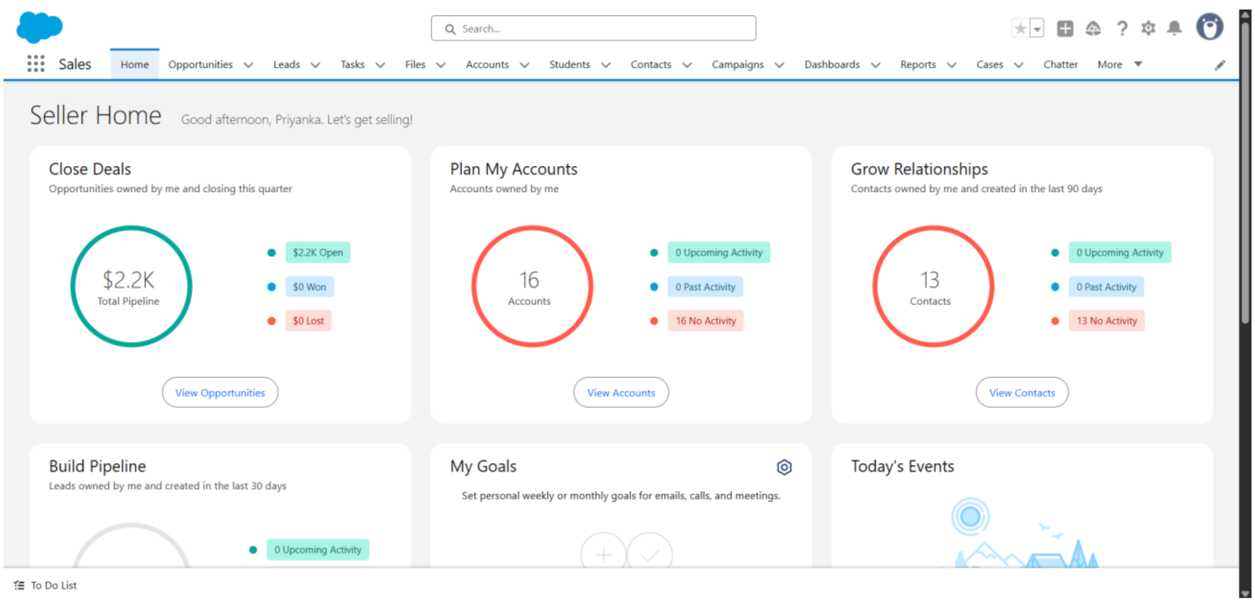
Use OWD and sharing rules to protect book data.

Showcase book entry, editing, and role-based visibility, and lay the groundwork for future fields and external integrations.

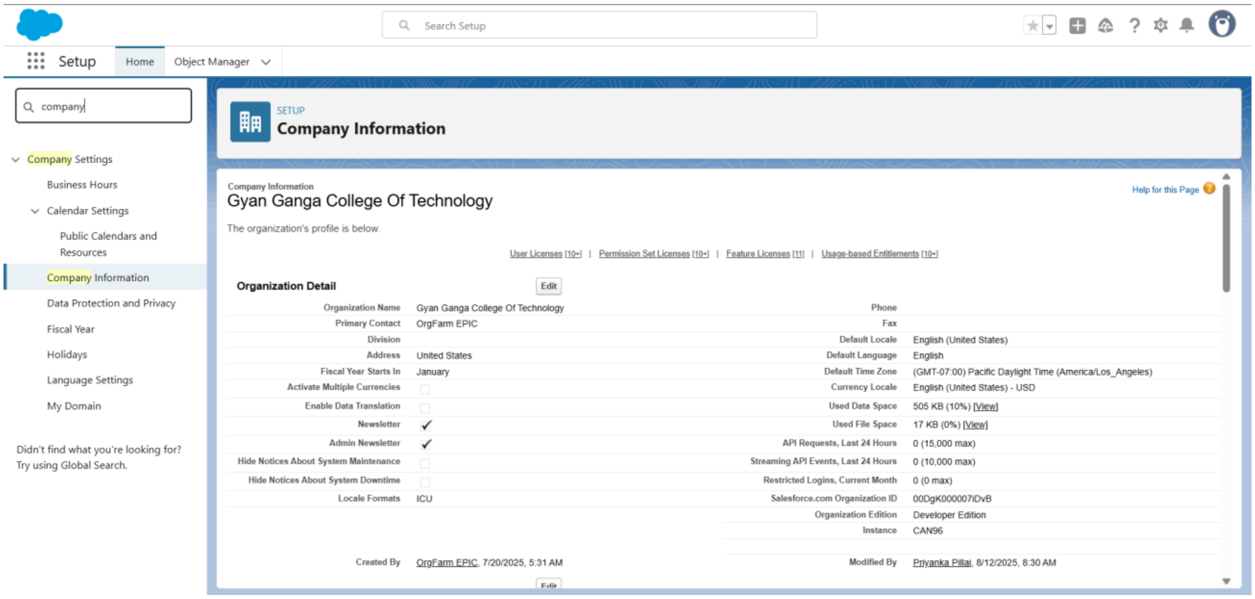
Step 1:- • Salesforce Editions

<u>Sign up & Login</u>

Signed up for a Salesforce Developer Edition and logged into the Lightning Experience. Confirmed access to the Setup area sing the gear icon.



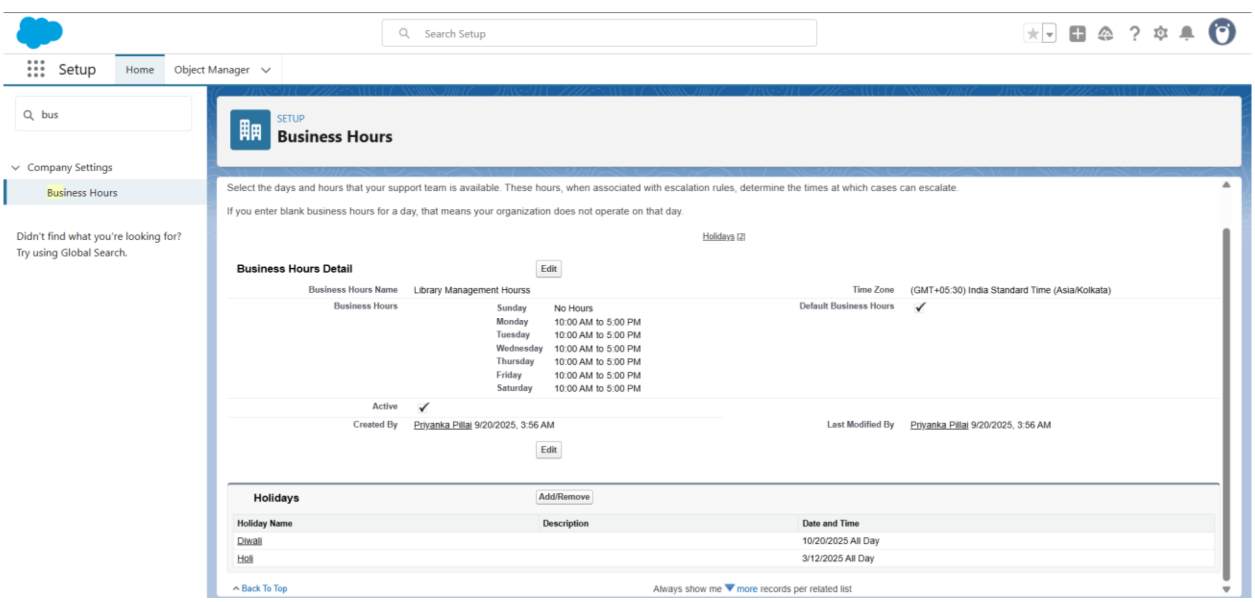
Step 2:- •Company Profile Setup



Step 3:- • Business Hours & Holidays

Created " Library Management Hours" Mon-Sat 10 AM-5 PM.

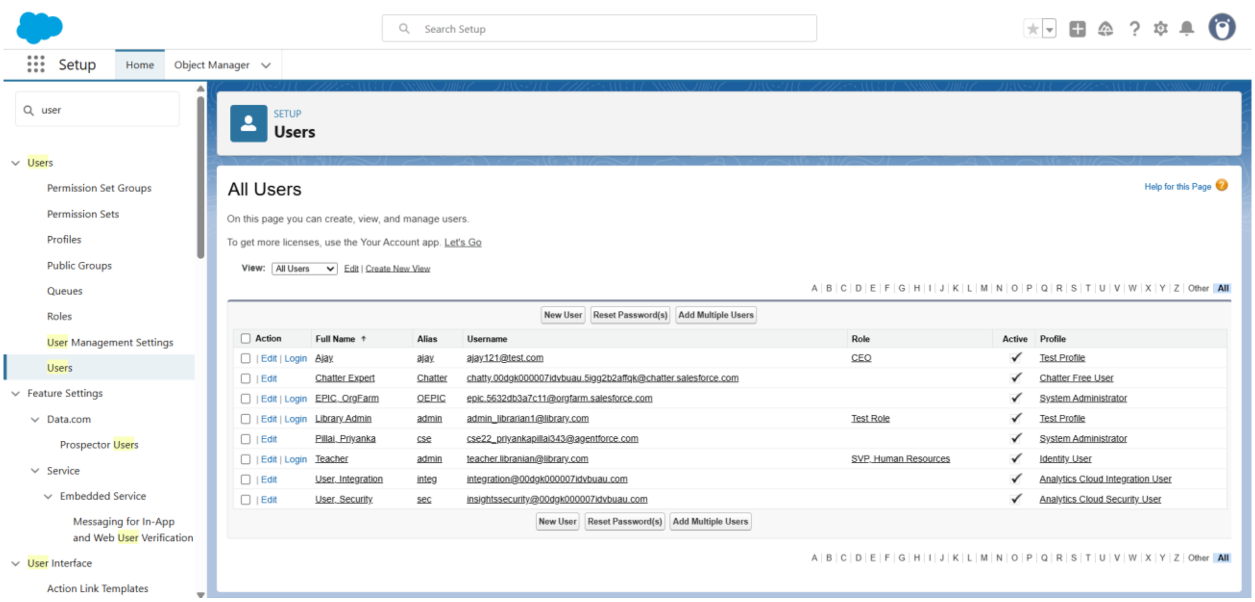
Added relevant Holidays and linked them to Business Hours.



Step 4 — Users

- Created 3 sample users:

Display Name	Username (example)	Profile	Role
Librarian Admin	admin_librarian1@library.com	Test Profile	Test Role
Teacher	Teacher.libranian@library.com	Identity User	SVP,Human Resouces



Step 5 Profiles

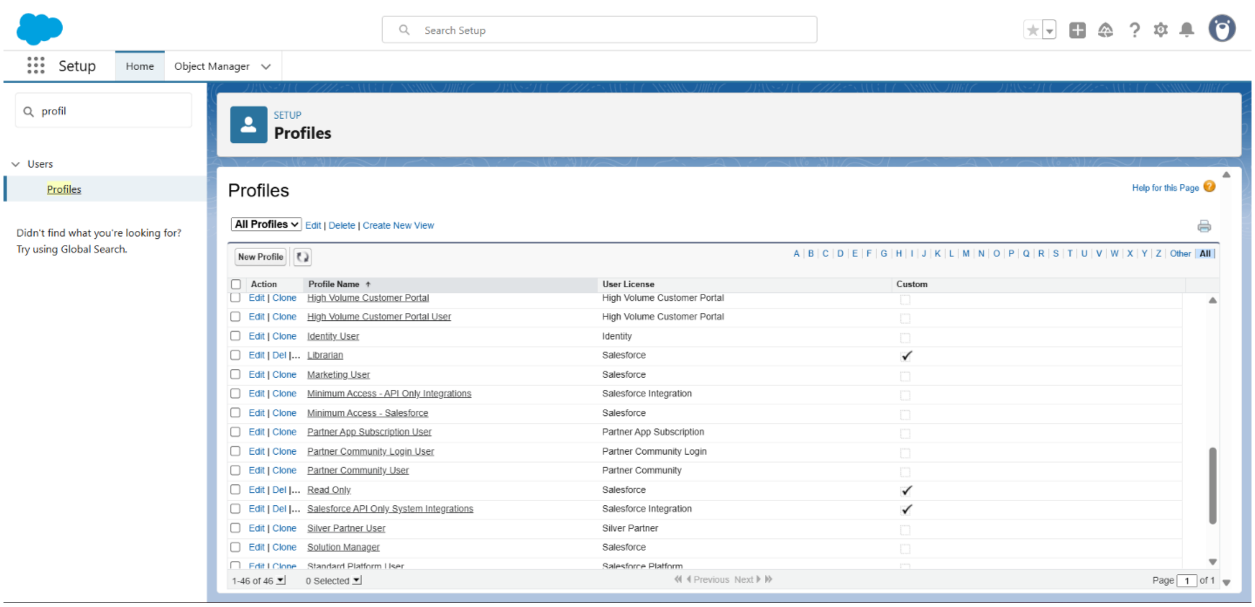
Cloned Standard User → created Librarian profile.

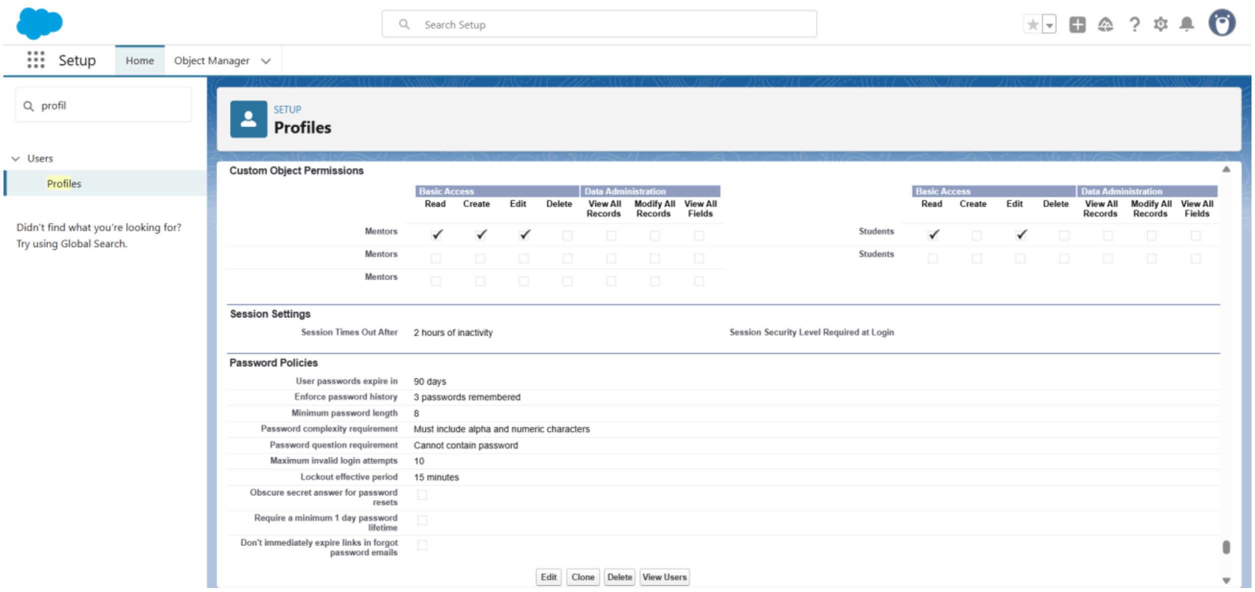
Cloned Standard User → created Teacher profile.

Adjusted Object Permissions:

Mentor = Read/Create/Edit/Delete

Student = Read/Edit/Create

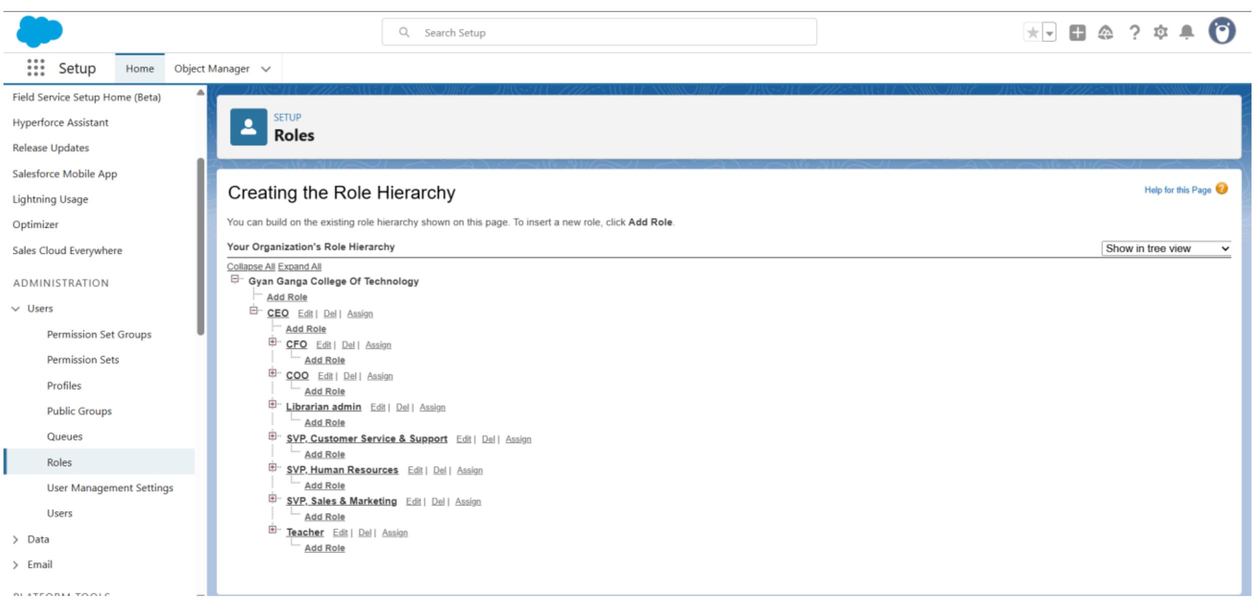




Step 6 — Role Hierarchy

Created simple hierarchy

- **Librarian Admin**
  - **Librarian**
  - **Teacher**

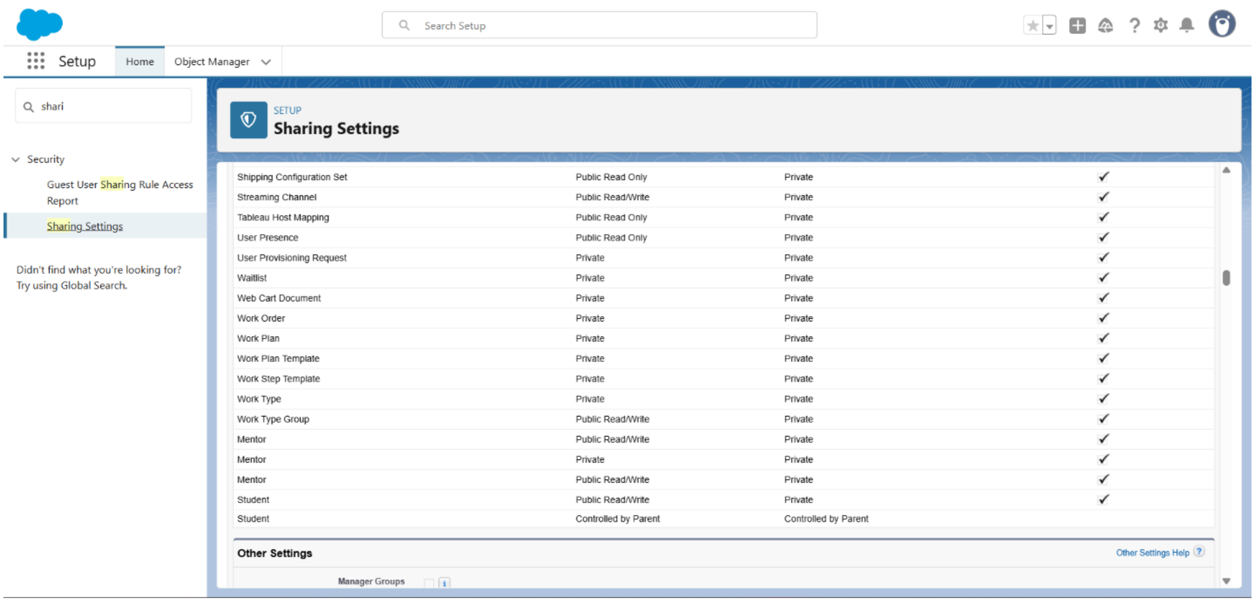


Step 7 Org:- Wide Defaults (OWD)

Setup → Sharing Settings

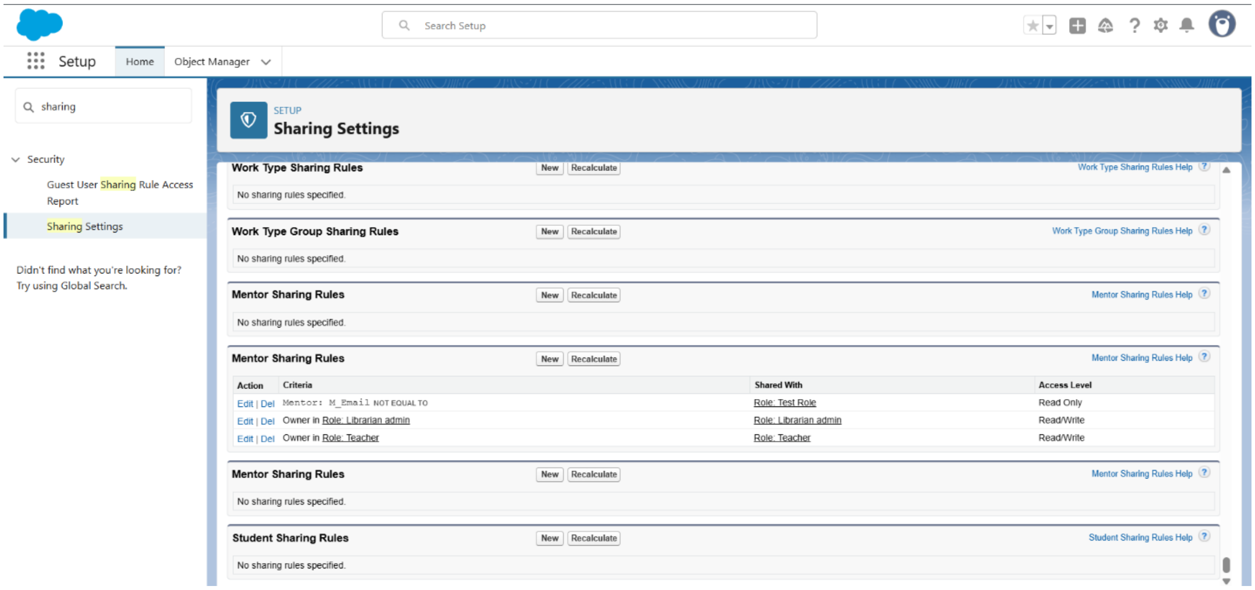
Mentor_c = Private.

Student_c = Controlled by Parent .



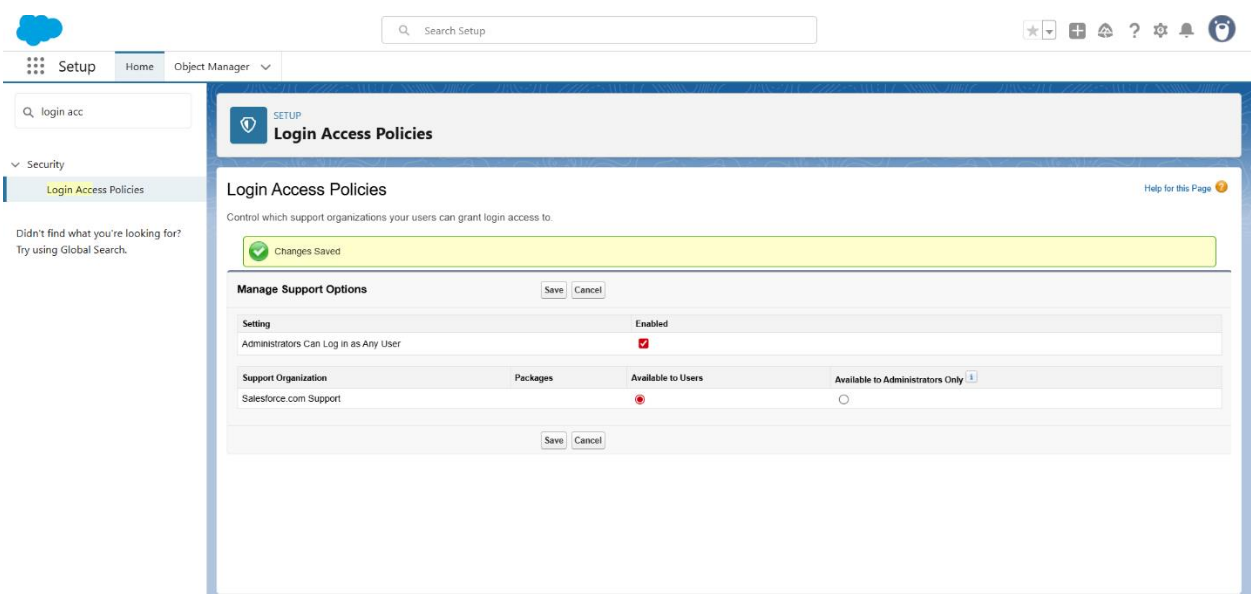
Step 8 : Sharing Rules

- **Created Public Group “Librarian” containing all Library users.**
- **Created Owner-based sharing rule:**
  - o **Owned by: Teacher admin**
  - o **Share with: Librarian admin**
  - o **Access Level: Read/Write.**



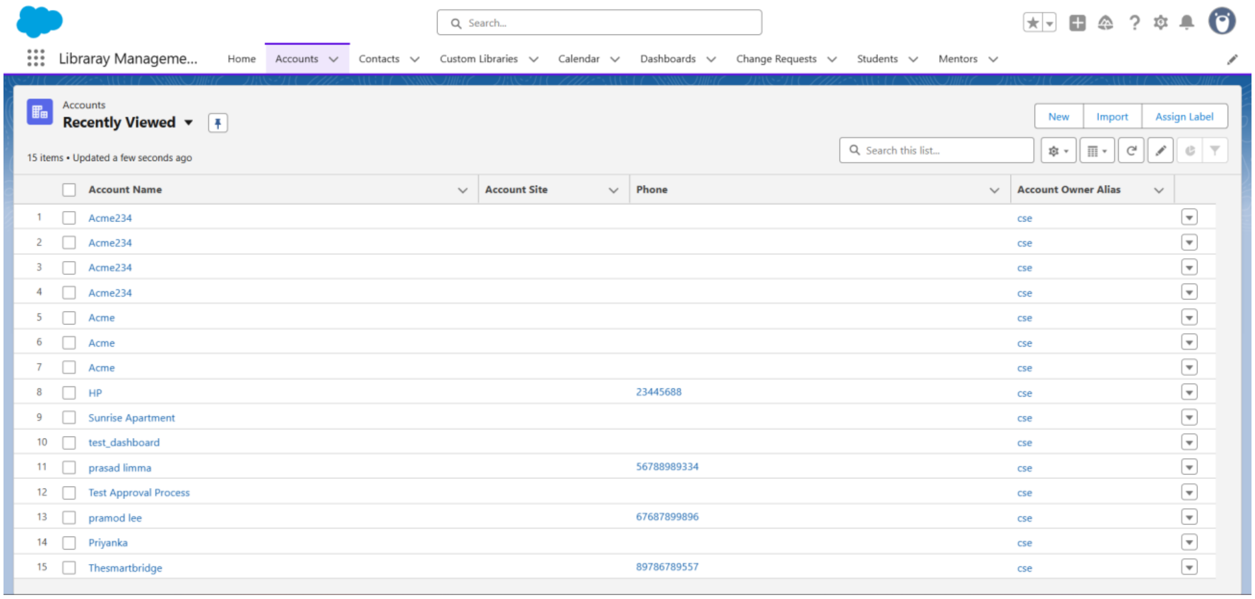
Step 9:- Login Access Policy

- **Enabled Administrators Can Log in as Any User.**

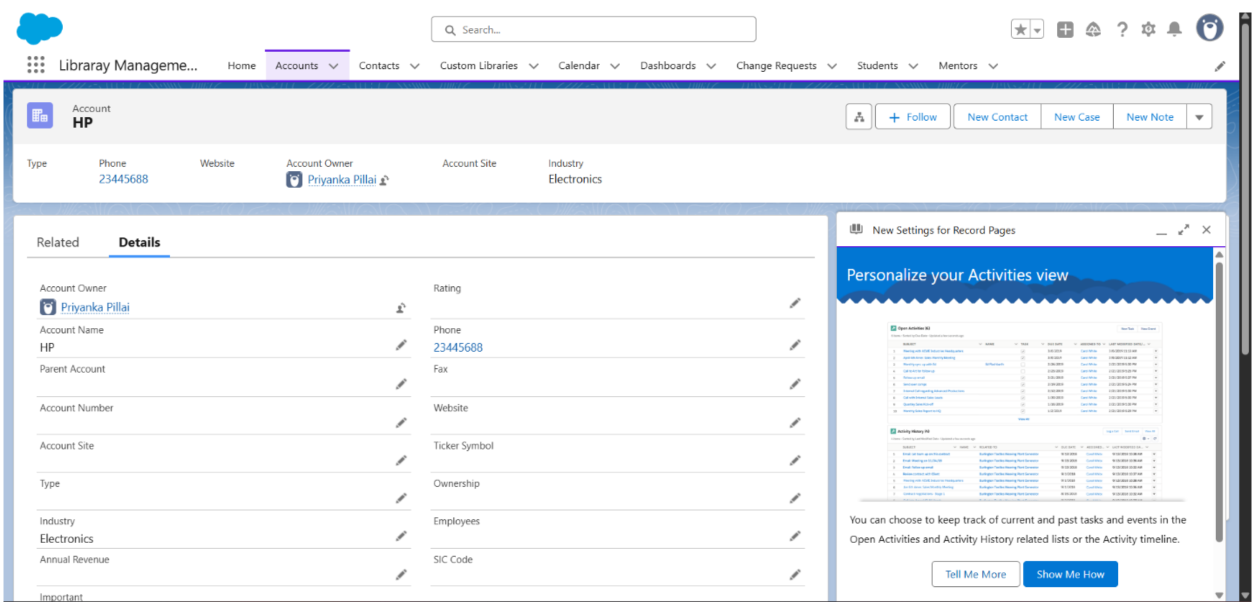


Step 10 — Library Management Lightning App

- **Setup → App Manager → New Lightning App "Library Management**



Step 11 : - Testing



Library Management : Phase 3

Introduction

Define and implement the data structure of Library Connect to link Members, Book Loans and Books. Configure record types, page layouts, navigation tabs and security so that each role (Admin, Librarian, Member) can perform its tasks.

1. Standard & Custom Objects

- ❖ Contact
- Used to store student information such as Date of Birth, Roll Number/Student ID, Contact Details.

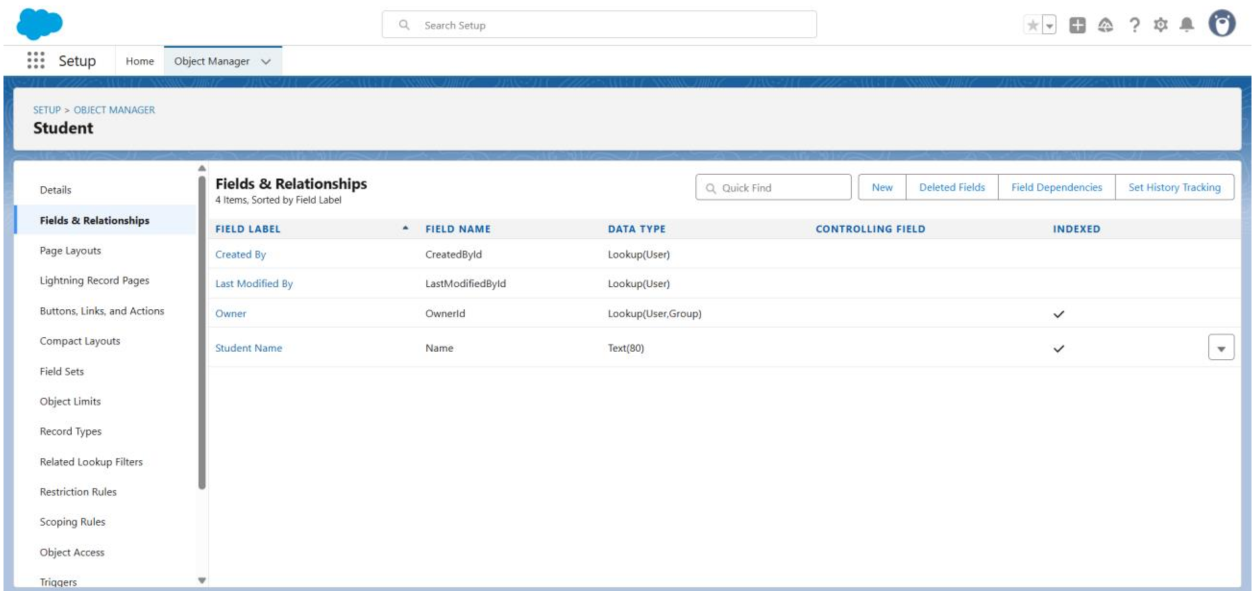
- ❖ Teacher
- Captures teacher information and tracks their classes/sessions. Includes fields for Subjects Taught, Qualifications, Availability/Timings, and lookup to Student (Contact) for assigned students or mentoring relationships.

- ❖ Student
- Tracks student academic cases like Performance, Issues, or Counseling. Includes fields for Description, Plan/Action, Status, and lookups to Student (Contact) and Teacher (User).

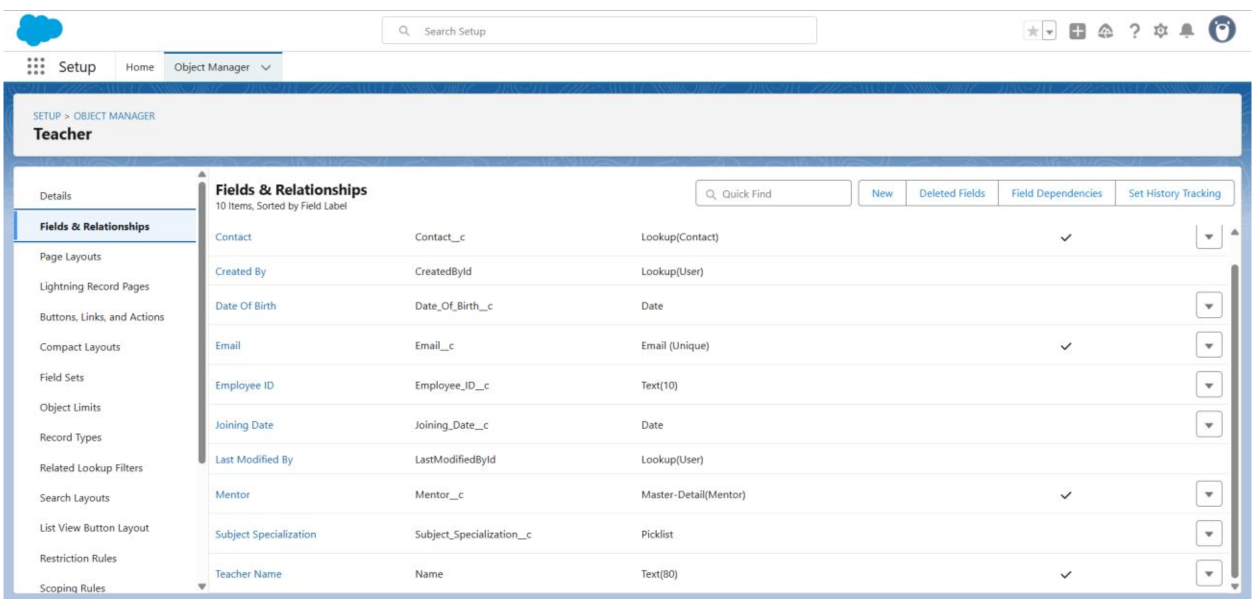
2. Fields

Student – Stores key student details (name, DOB, ID, contact, class, status) and links them to book loans, courses, and mentors to track borrowing and enrollment.

Teacher – Stores main teacher details (name, DOB, ID, subject, contact, status) and links them to students, classes, and book loans to show who teaches or approves activities.



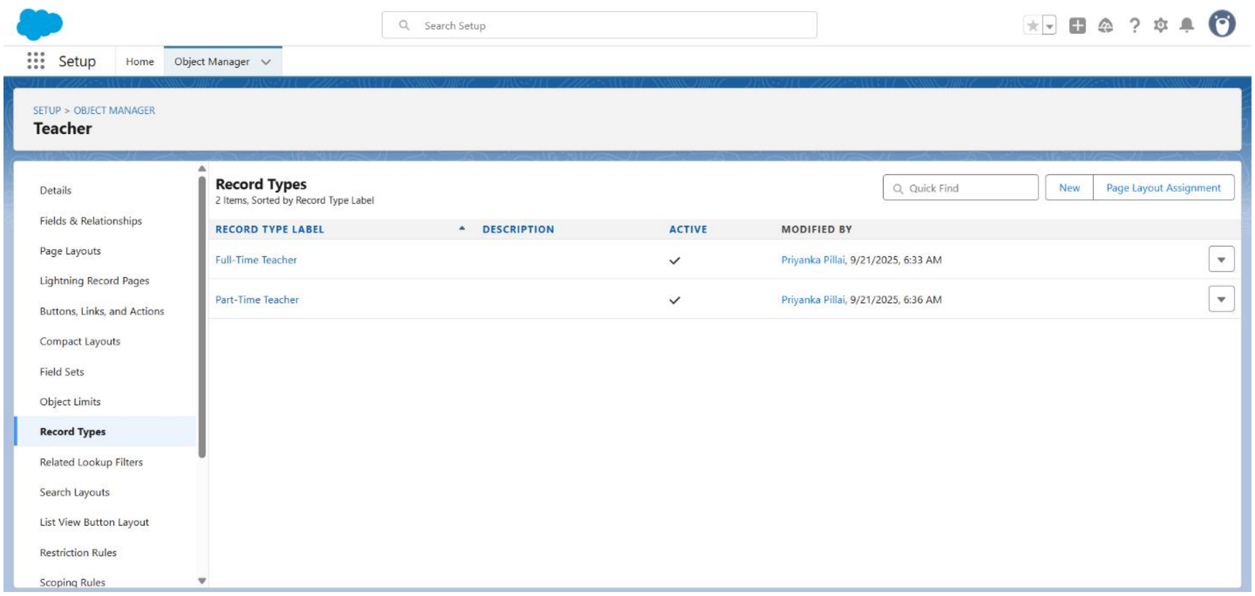
Student



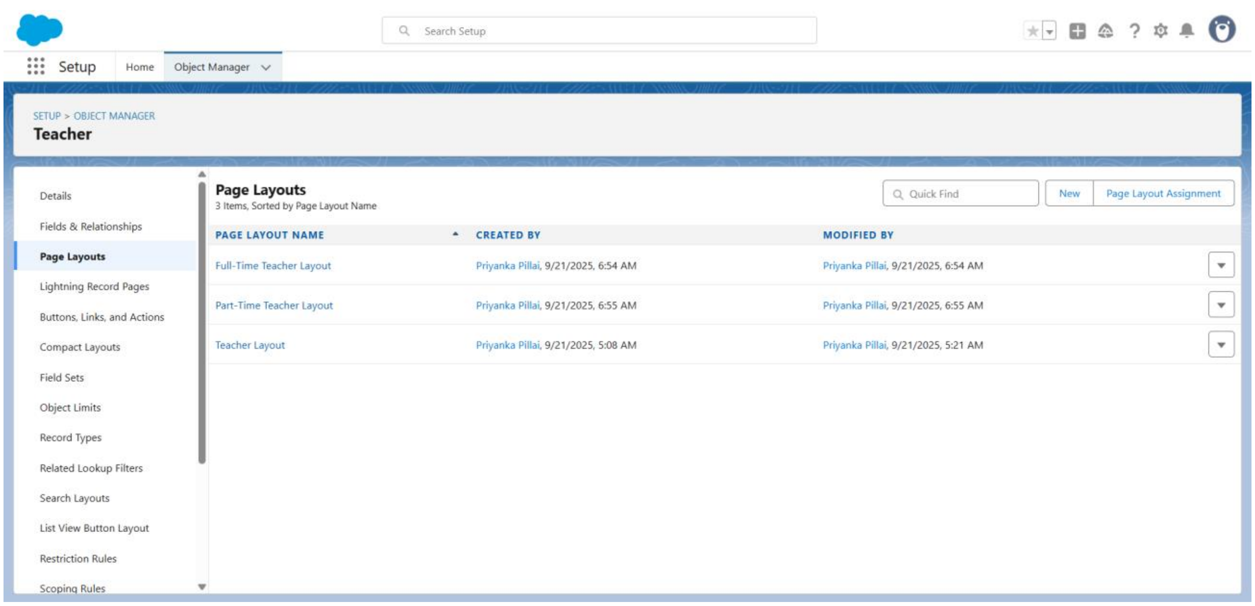
Teacher

3. Record Types

- **Teacher record types: Full-Time Teacher and Part-Time Teacher**
- **Separate page layouts exist for each record type to show relevant fields.**

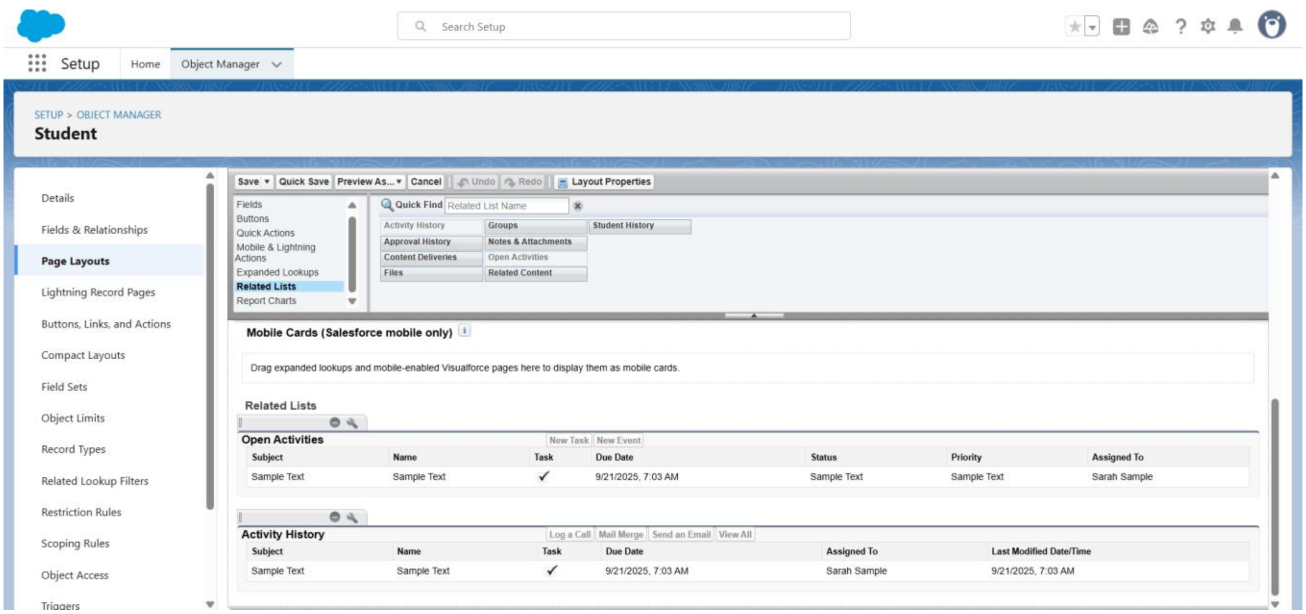


Teacher Record Types

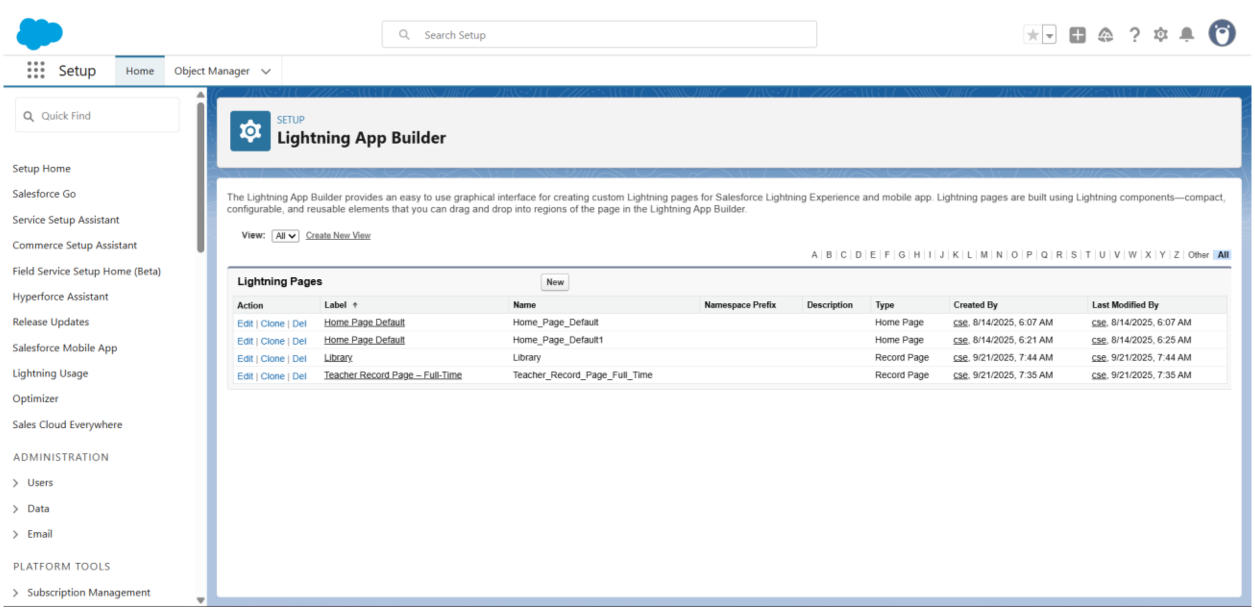


Teacher page layout

4.Page Layouts & Related Lists

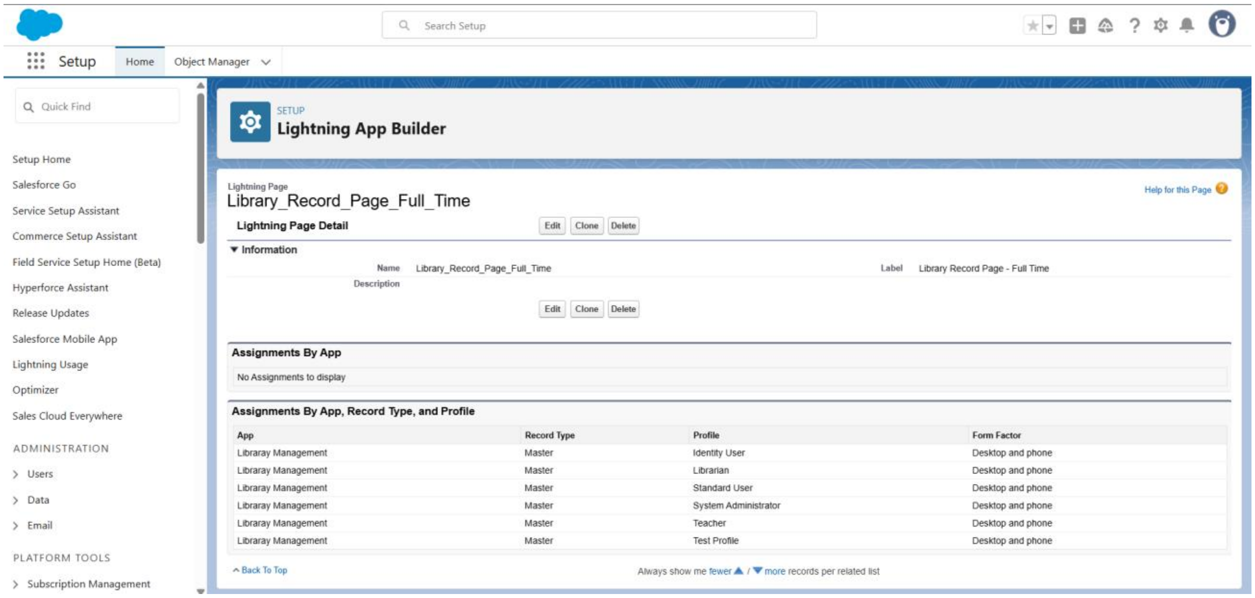


5. Lightning Record Pages & Compact Layouts



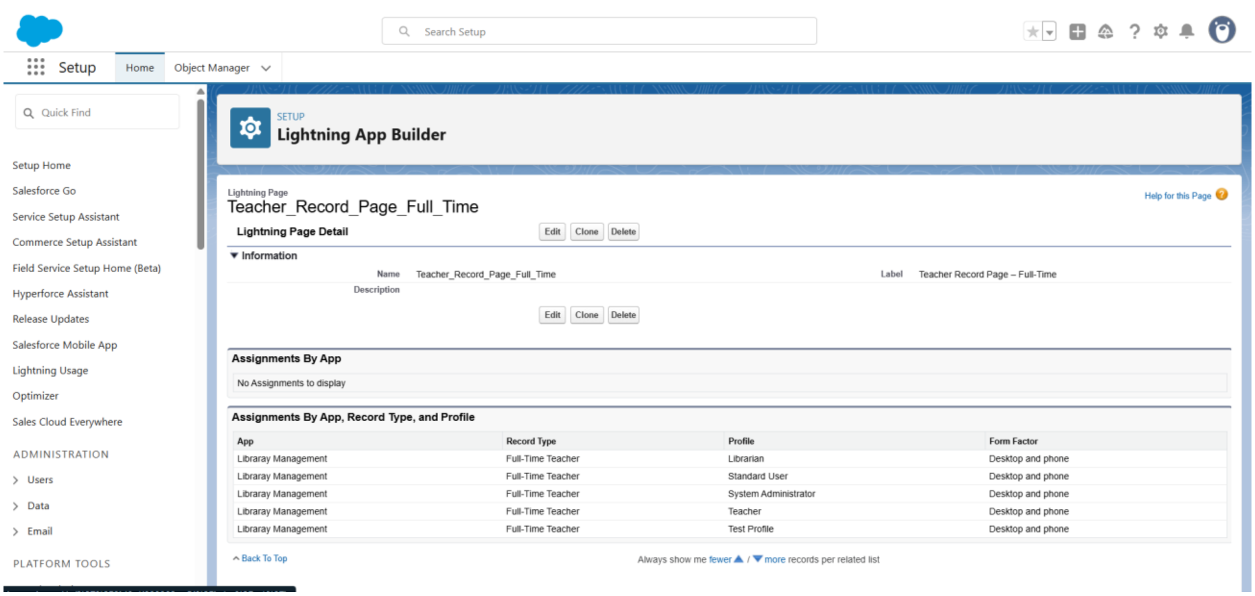
➢ **Library**

- **Custom Lightning Record Pages created for Library to give Librarians and Students tailored views.**
- **Compact Layouts show key Library fields (Book Title, Author, Availability, Genre) for quick glance.**



➢ **Teacher**

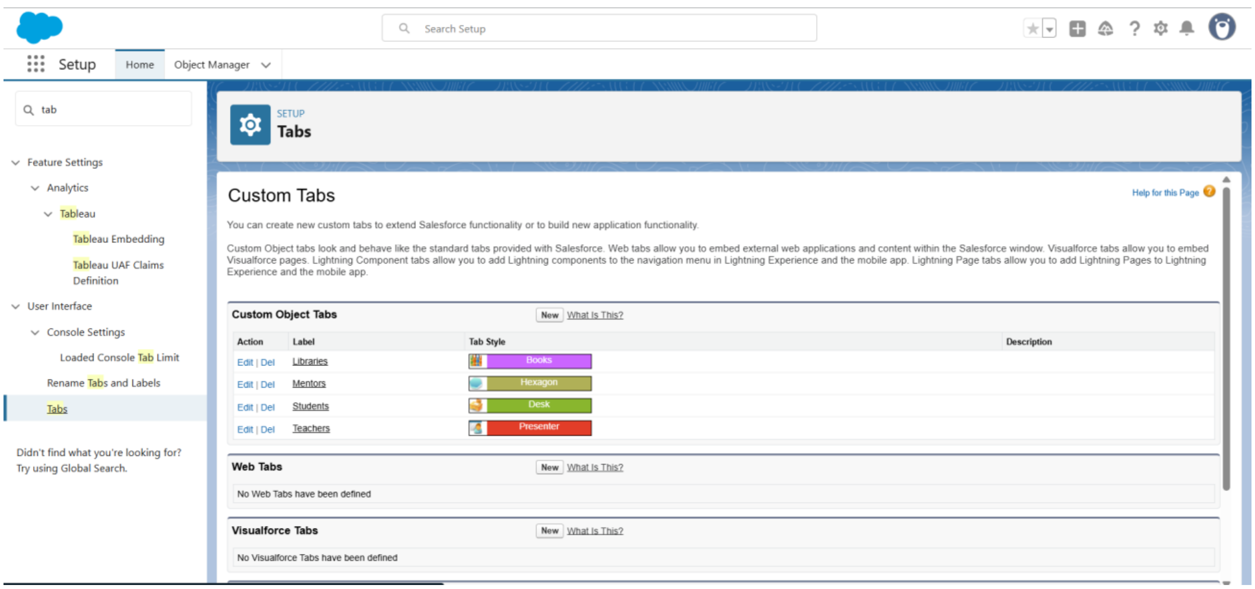
- **Custom Lightning Record Pages created for Teacher to give Admins and Staff tailored views.**
- **Compact Layouts show key Teacher fields (Name, Subject, Class, Contact) for quick glance.**



6. Tabs & App Navigation

- **Teacher**
- **Tabs & Navigation: Teachers, Students, Classes, Attendance, Reports. Custom tab for Teachers.**

- **Library**
- **Tabs & Navigation: Library, Books, Students, Issued Books, Reports. Custom tab for Library.**



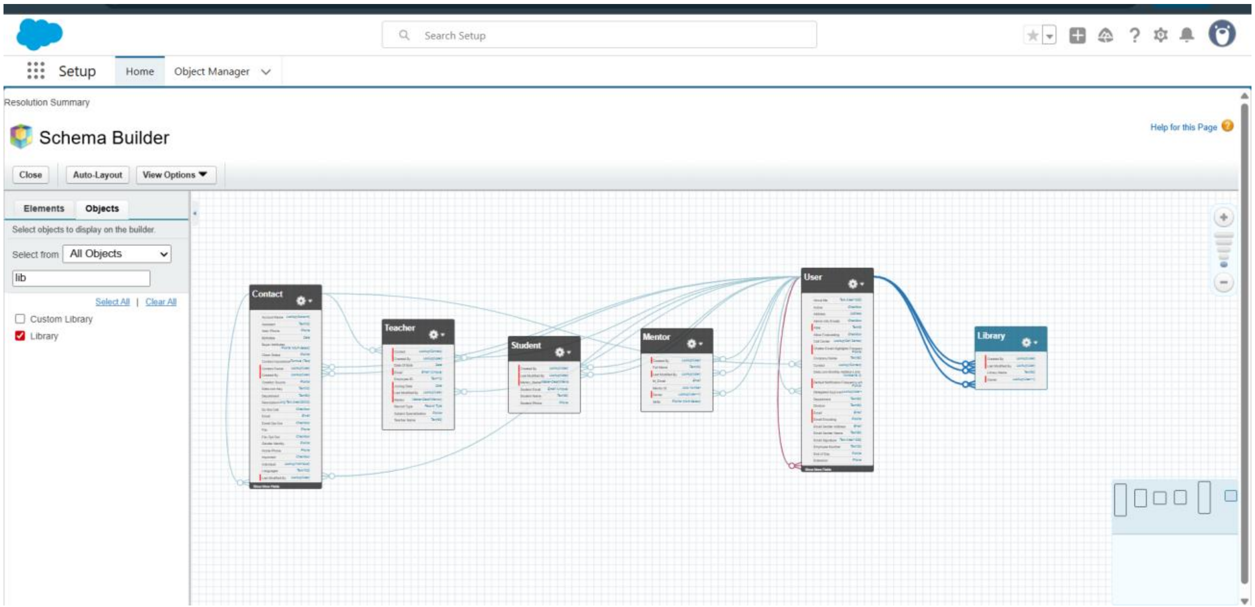
7. Schema Builder (ERD)

Teacher

Schema Builder (ERD): Displays Teacher, Student, Class, and User objects with lookup relationships. Teacher is central, linked to Classes and Students.

Library

Schema Builder (ERD): Displays Library, Book, Student, and User objects with lookup relationships. Library is central, linked to Books and Students.



8. Security & Field-Level Visibility :

- **Teacher profile has Read/Create/Edit on Student and Class records.**
- **Librarian profile has Read/Create/Edit on Library and Book records, but read-only on Student records.**
- **Sensitive fields like Salary or Private Notes hidden from Librarian profile.**
- **Mentor profile can view assigned Students and Classes but cannot edit Library records.**

9. Sample Data & Testing :

- **Sample Students and Teachers created with details like Name, Class, and Contact info.**
- **Library records (Books) created by Librarian, linked to Students and Teachers with record type "Issued" or "Available."**
- **Mentor records created and linked to Students for guidance tracking.**
- **Teacher login view shows Classes and Students; Student record shows related lists (Books issued, Mentor info).**

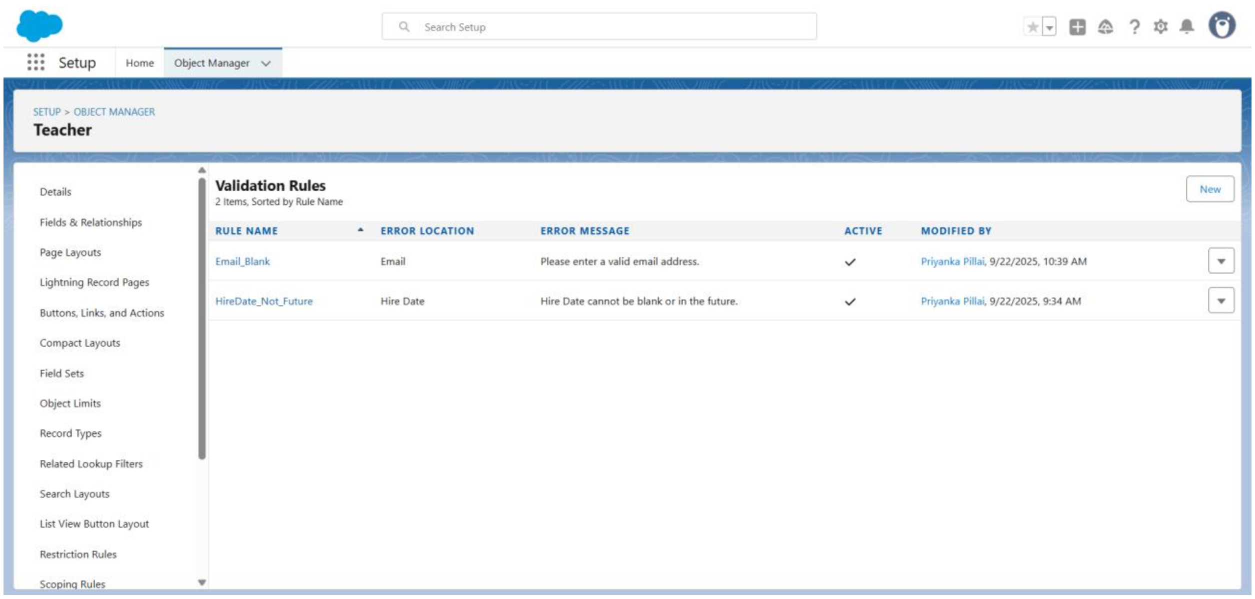
10. Current System Behaviour :

- **Library records link both Students and Teachers for issuing/tracking books.**
- **Mentors hold guidance/notes information linked to the same Students.**
- **Record types like Issued Book and Available Book display appropriate layouts.**
- **Student records show all related Classes, Library records, and Mentor assignments.**
- **Teachers can view Students and Classes assigned to them.**
- **Librarians can create Library records and view limited Student info.**
- **The Schema Builder ERD reflects the current relationships between Teacher, Library, Mentor, Student, and User.**

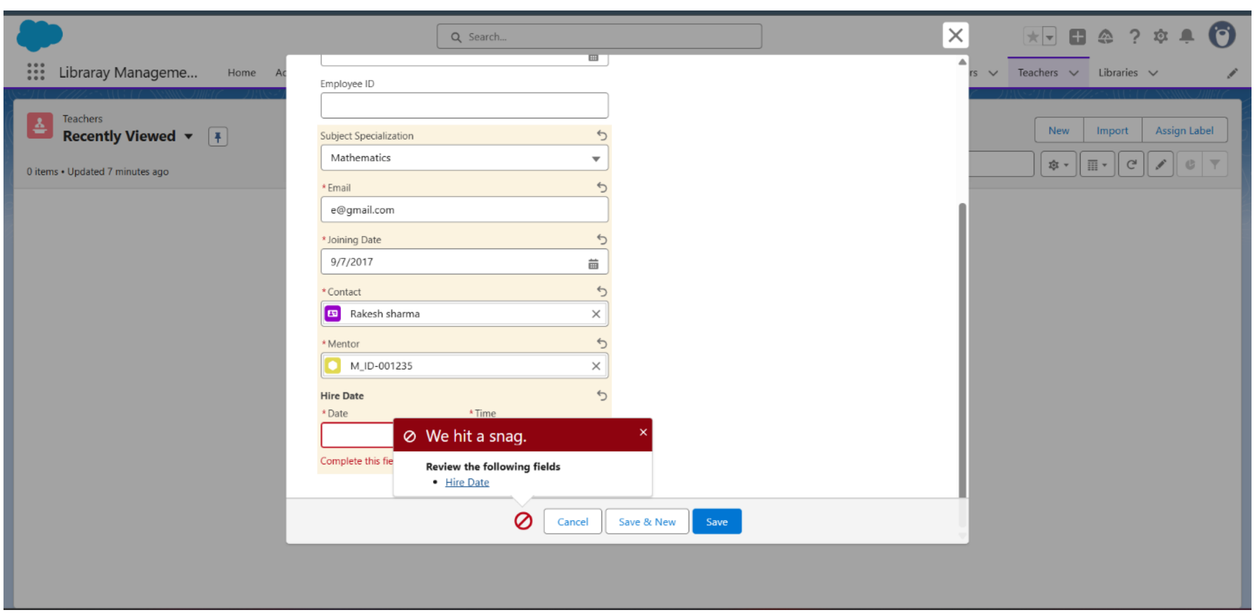
Library Management:- Phase 4: Process Automation (Admin)

1. Validation Rules

- **A validation rule created on Teacher object HireDate_Not_Future Date/ Time Hire Date cannot be blank or in the future.**
- **A validation rule created on Teacher object Email_Blank Please enter a valid email address.**



Validation Rules

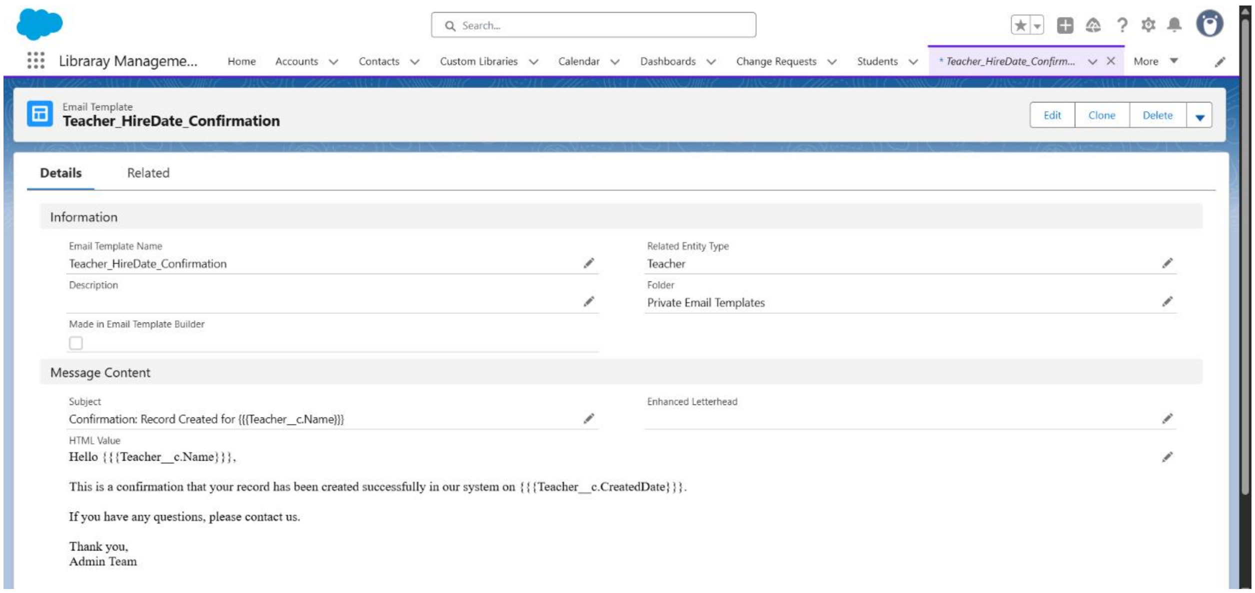


Validation Rules Testing

2. Email Templates for Automation

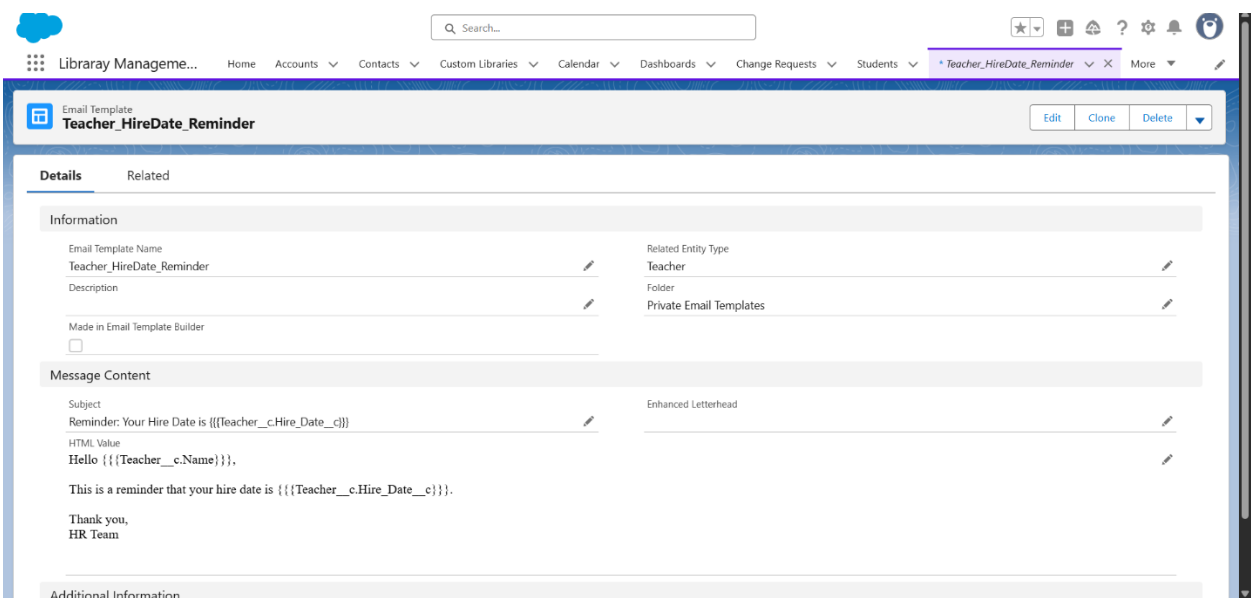
Two classic email templates were created and stored under Private Email Templates:

Teacher_HireDate_Confirmation : – sends confirmation to the teacher when a date is created or confirmed.



Teacher_HireDate_Confirmation

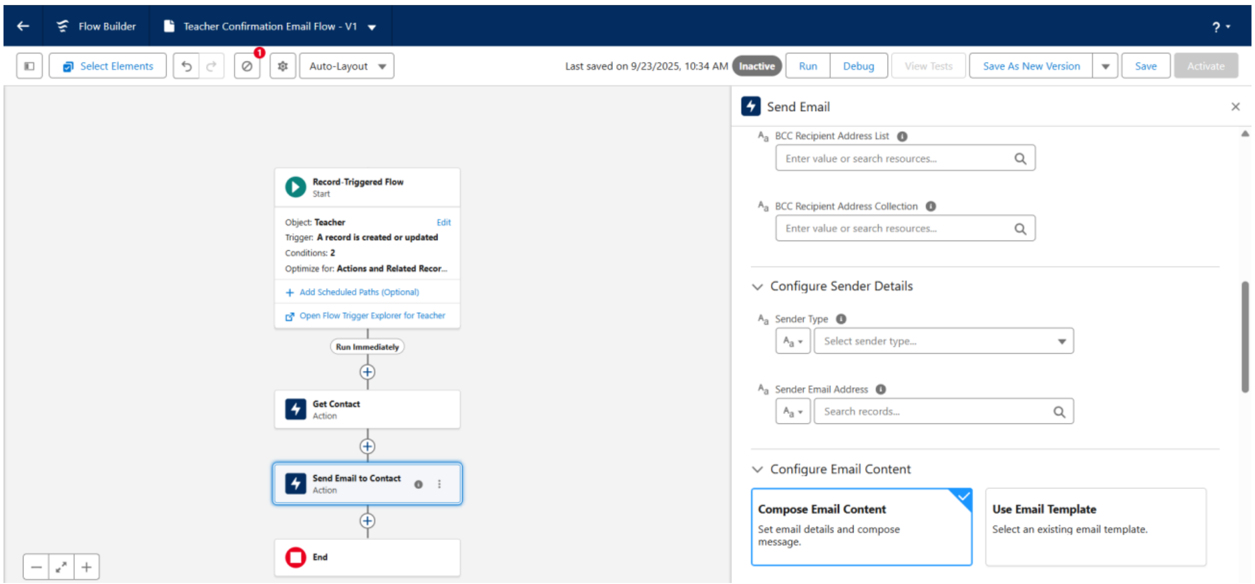
Teacher_HireDate_Reminder :-



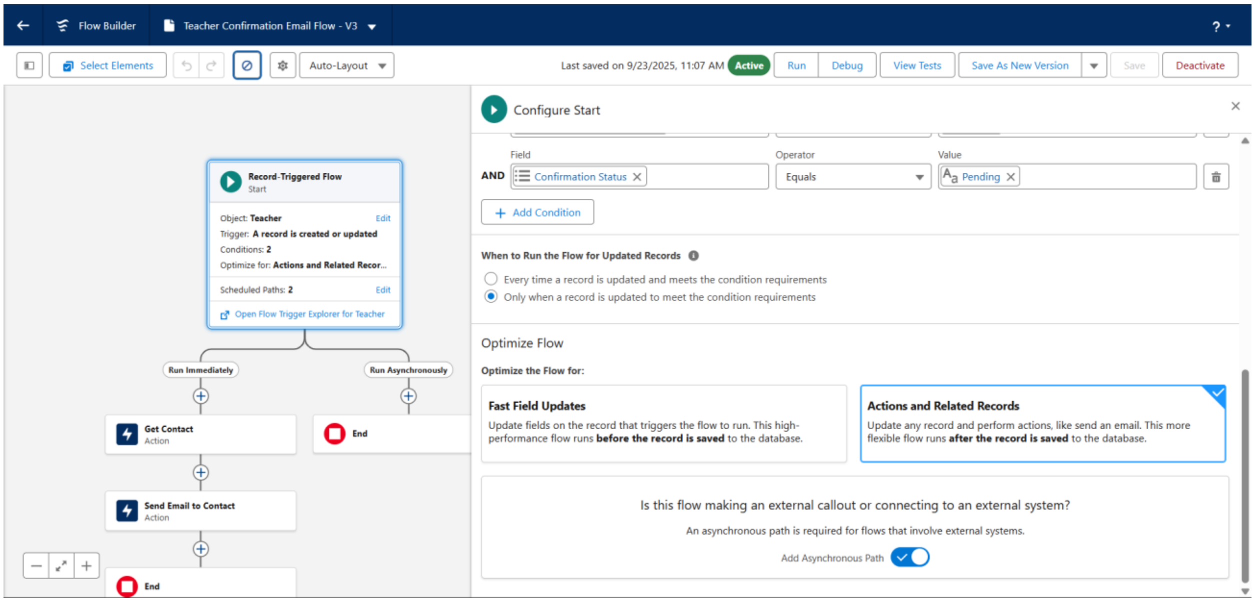
Teacher_HireDate_Reminder

3. Flow for TeacherConfirmation :-

- A record-triggered flow “Teacher_Confirmation_Flow” built on Teacher object:
- Trigger: When record is created or when Status changes to “Confirmed.”
- Action:
- Update “Confirmation Sent” checkbox on the Teacher.



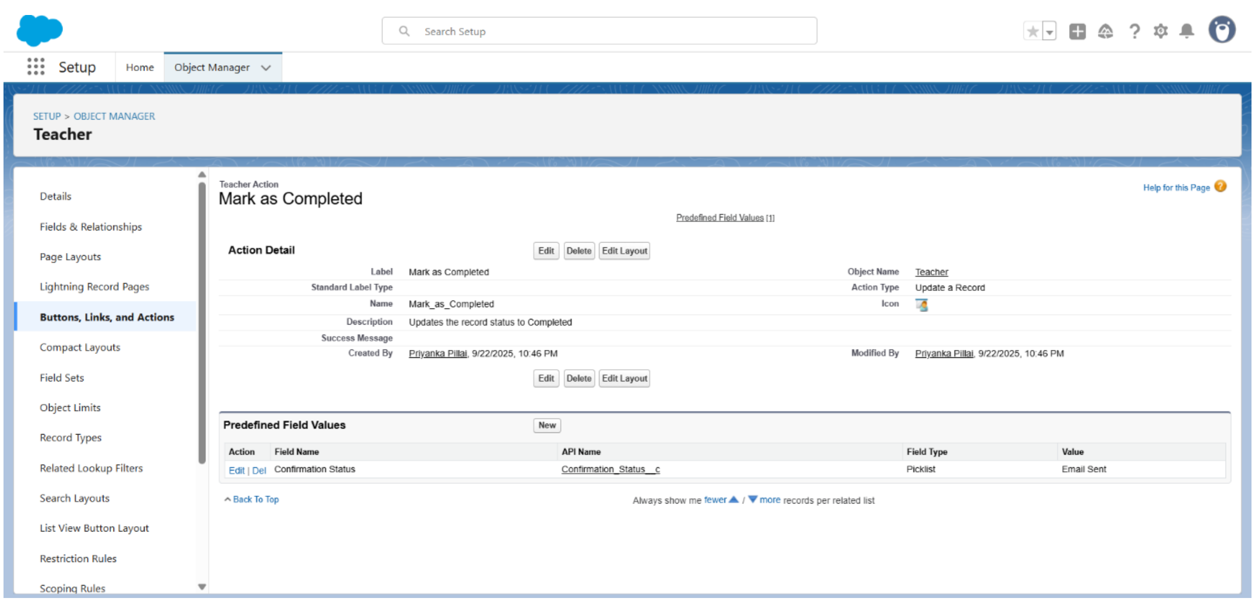
4. **Reminder Notification & Email (Scheduled Path)**



6. Quick Action – “Mark as Completed”

A Quick Action added Teacher object:

- **Action Type: Update a Record.**
- **Predefined Field Values: Status c = “Completed.”**



Mark as Completed

8. Testing & Results

Sample Appointments were created to test the automation:

Library Management :- Phase 5: Apex Programming (Developer)

- Classes & Objects:

- Created an Apex utility class **TeacherService:**
- Inner Comparator: **TeacherHireDateComparator** – sorts teachers by **Hire_Date__c.**

- Static Method:

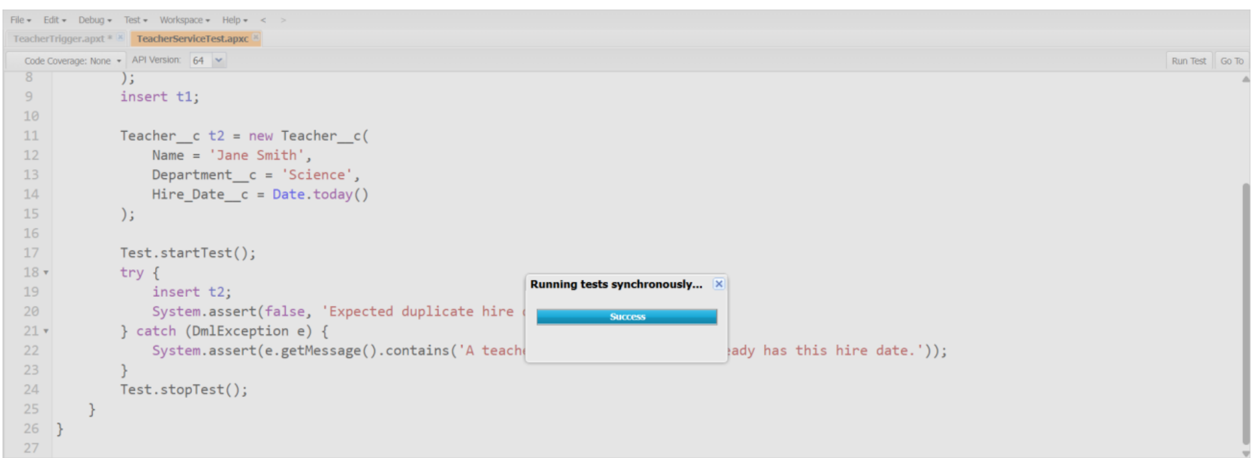
- **preventDuplicateHireDates(List<Teacher_c> incoming, Map<Id, Teacher_c> oldMap)** – checks new/updated teacher records against existing teacher records to ensure no conflicting hire dates for the same department (or other criteria) and uses **addError()** to block duplicates.

File → New → Apex Class.

Name it TeacherServiceTest → click OK.

Paste test code:

Save.



2.Apex Trigger

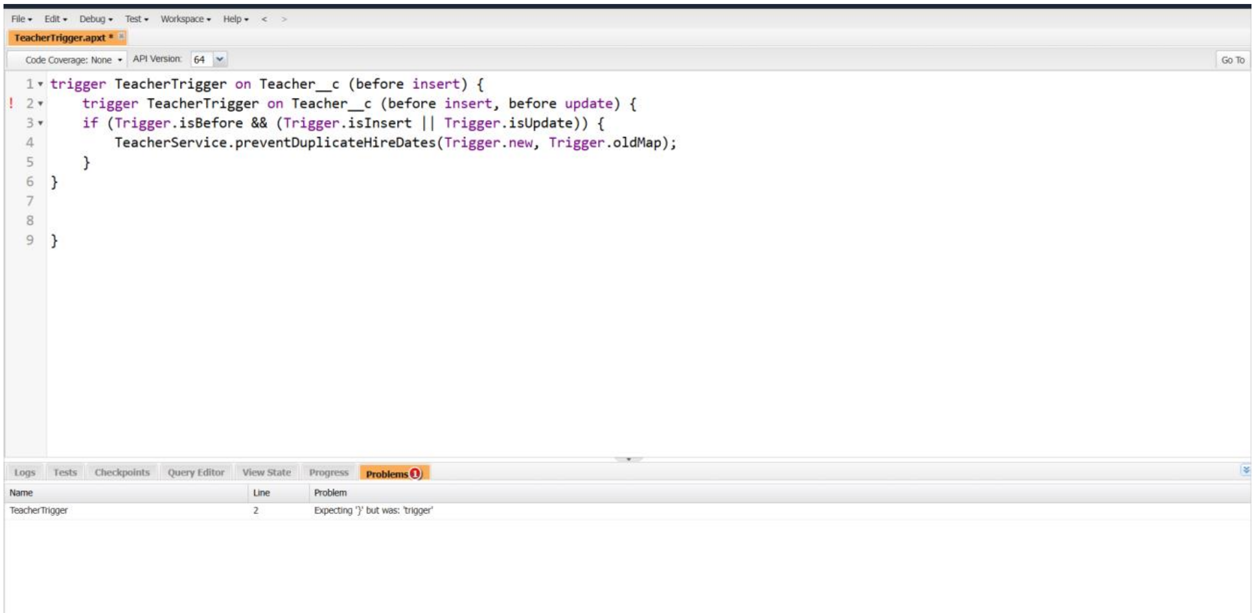
File → New → Apex Trigger.

Enter:

- Name: TeacherTrigger
- sObject: Teacher__c

Click Submit.

Save.



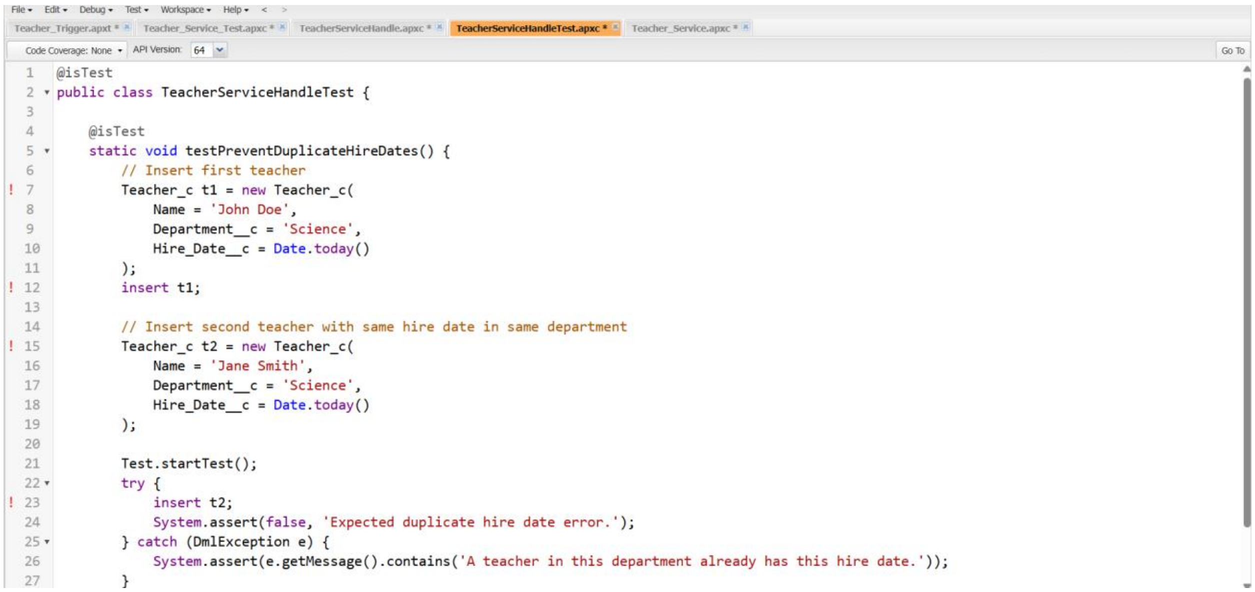
3.Trigger Design Pattern:

Business logic is kept in the TeacherService class; the trigger on Teacher__c simply calls the class method.

4. SOQL & Collections

Inside preventOverlap:

- SOQL queries existing Teacher for relevant new Teacher.




```
@isTest
public class TeacherServiceHandleTest {

    @isTest
    static void testPreventDuplicateHireDates() {
        // Insert first teacher
        Teacher_c t1 = new Teacher_c(
            Name = 'John Doe',
            Department__c = 'Science',
            Hire_Date__c = Date.today()
        );
        insert t1;

        // Insert second teacher with same hire date in same department
        Teacher_c t2 = new Teacher_c(
            Name = 'Jane Smith',
            Department__c = 'Science',
            Hire_Date__c = Date.today()
        );

        Test.startTest();
        try {
            insert t2;
            System.assert(false, 'Expected duplicate hire date error.');
        } catch (DmlException e) {
            System.assert(e.getMessage().contains('A teacher in this department already has this hire date.'));
        }
```

5. Control Statements

for loops, if conditions and addError() to prevent overlaps.

6. Test Classes (Teacher Example)

A test class **TeacherTriggerTest** was created to:

- Insert a valid teacher (no duplicate hire date in the same department) and verify it saves successfully.
- Attempt to insert a teacher with a duplicate hire date in the same department and verify a **DmlException** with “already has this hire date” in the message is thrown.

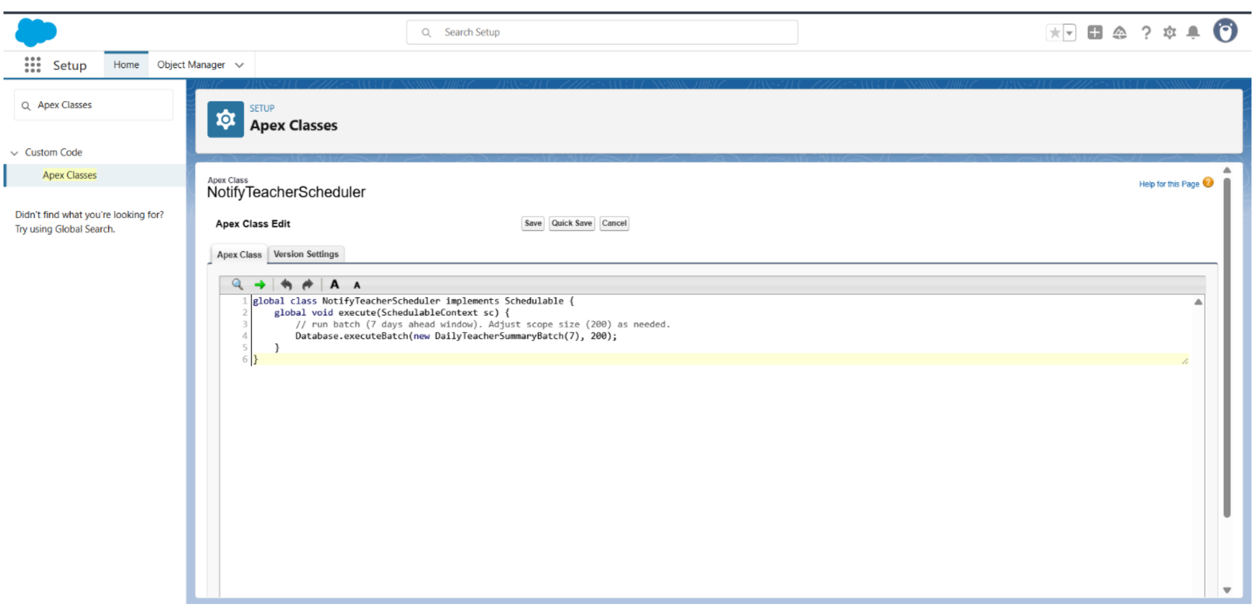
This test indirectly executes the logic in

TeacherServiceHandle.preventDuplicateHireDates and gives code coverage for both the trigger (**TeacherTrigger**) and the service class (**TeacherServiceHandle**).

7. **Batch Apex and Scheduled Jobs**

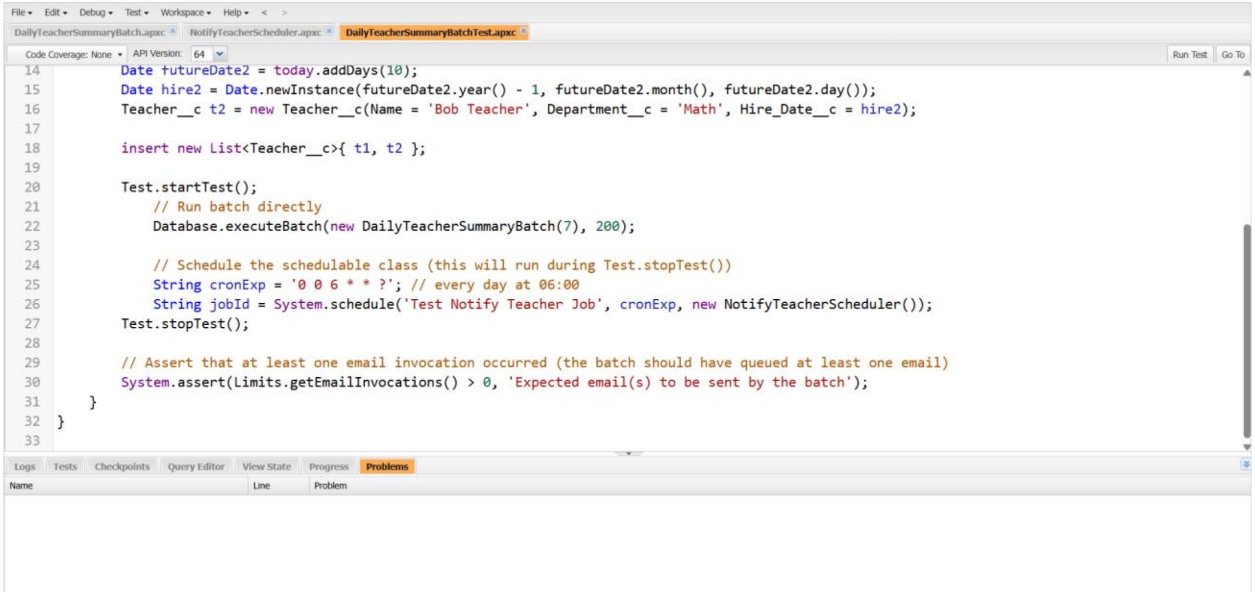
A Batch Apex class TeacherSummaryBatch was created to:

- Query recently hired teachers by department.
- Generate and log summary reports (or prepare reminders for admin review).
- Run daily at 6:00 AM using a Scheduled Apex job.



8.Exception Handling

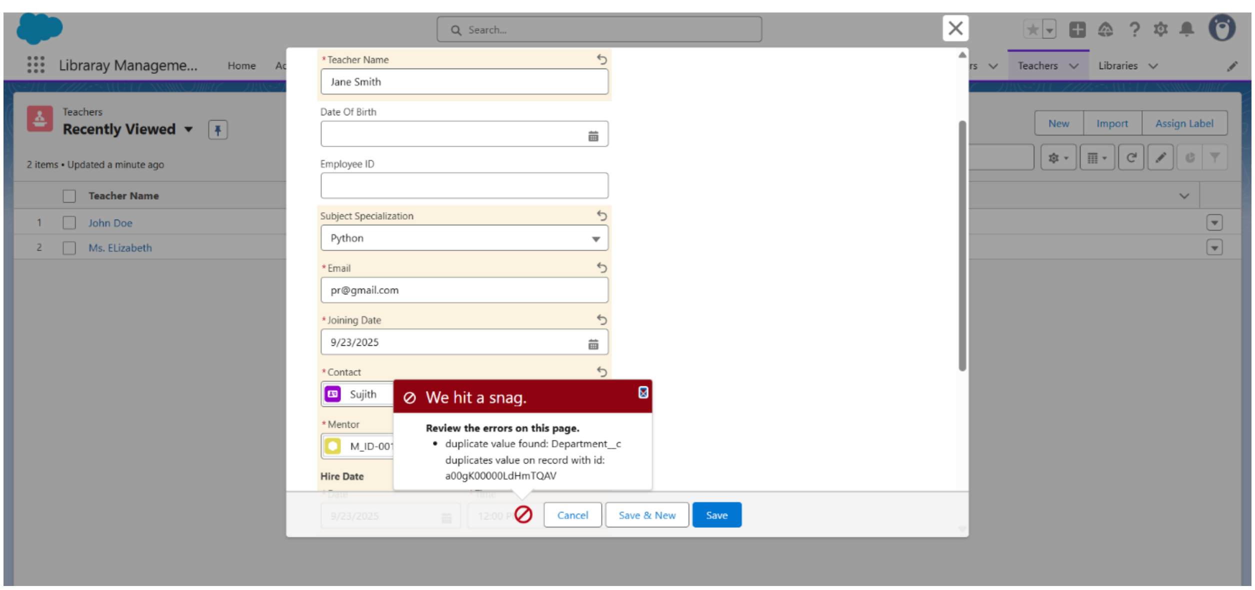
Used **addError()** on SObjects to stop DML with a user-friendly message.



9.Skipped / Not Implemented

Queueable Apex, Future Methods, Asynchronous Processing beyond the daily batch job were not implemented at this phase.

10. **Testing & Results**



Library Management : Phase 6: User Interface Development

Introduction

This phase focuses on building an intuitive Lightning experience for different library roles (Teacher, Student, Librarian/Admin).

Using Lightning App Builder, Record Pages, Tabs, Home Page Layouts, Utility Bar, plus a custom LWC and Apex integration, a seamless UI was delivered to every user type.

1. Lightning App Builder

• Customized Pages for Each Role: Different Lightning Record Pages were designed for Book/Resource – Teacher View and Book/Resource – Student View (and optionally Librarian/Admin View).

Role-Based Components: Each page displays only the components relevant to that specific role, e.g.

– Teacher: can upload resources, create assignments, view students' borrowing history.

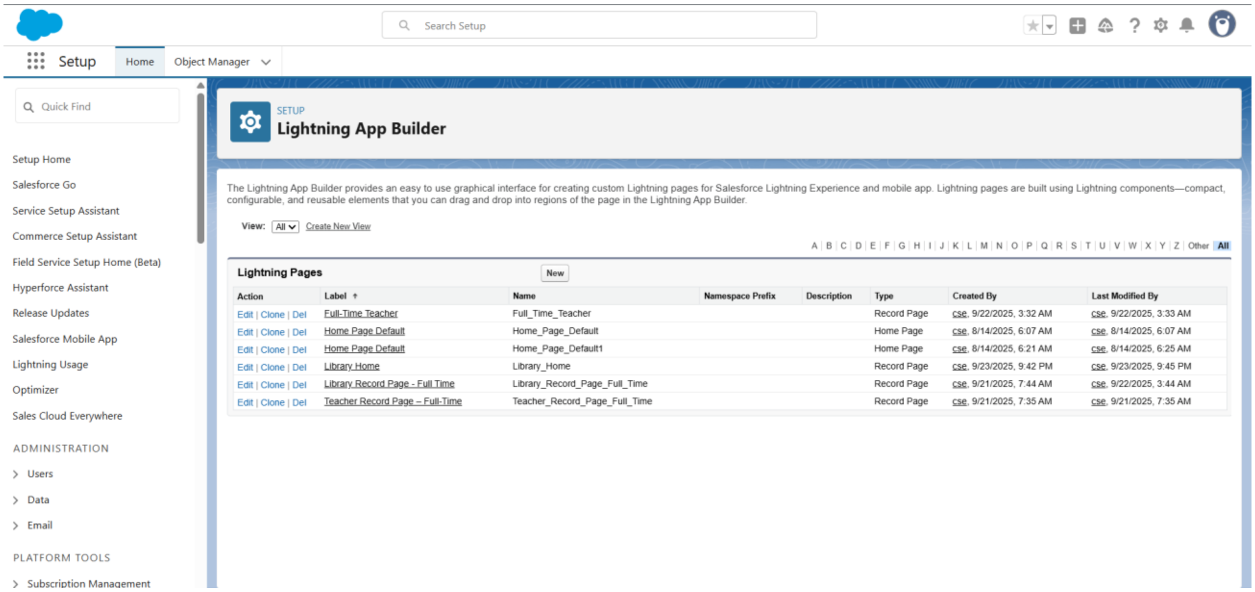
– Student: can search books, borrow/return, view personal borrowing history.

– Librarian/Admin: can add/edit/delete books, manage stock, approve requests.

• Tabs & Home Layouts: Tabs and Home Page Layouts were set up separately for Teacher, Student, and Librarian/Admin so each profile lands on the correct workspace.

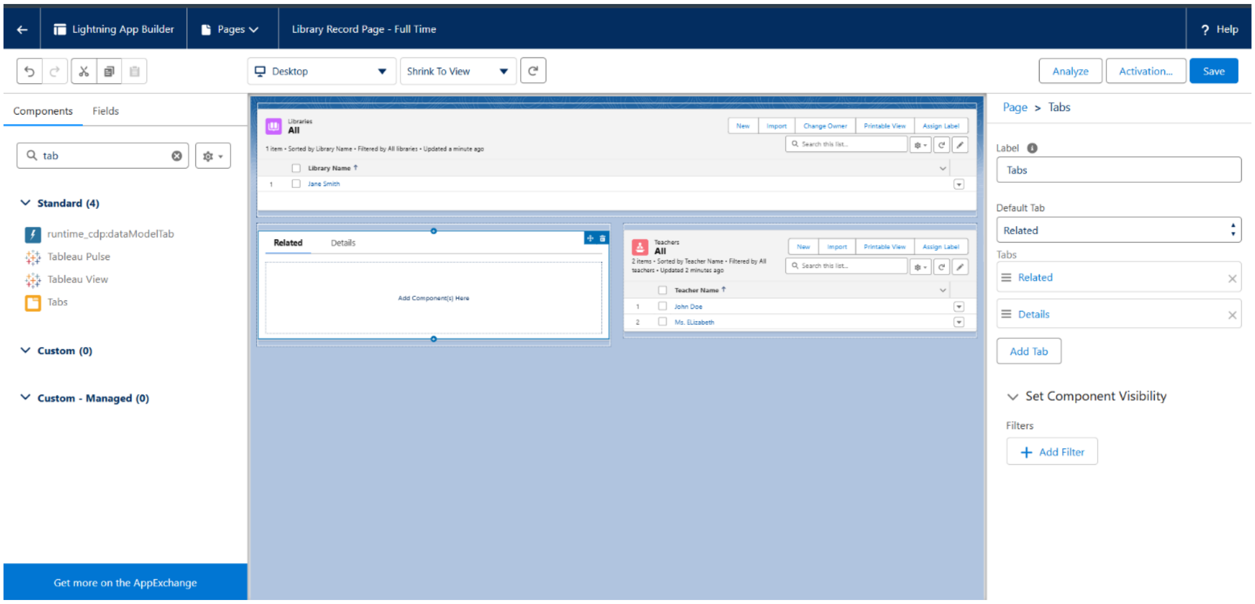
Utility Bar: Added quick-access actions and shortcuts in the Utility Bar for commonly used features by each role, e.g.

Librarian/Admin: "Add New Book," "View Overdue Items," "Send Notifications."

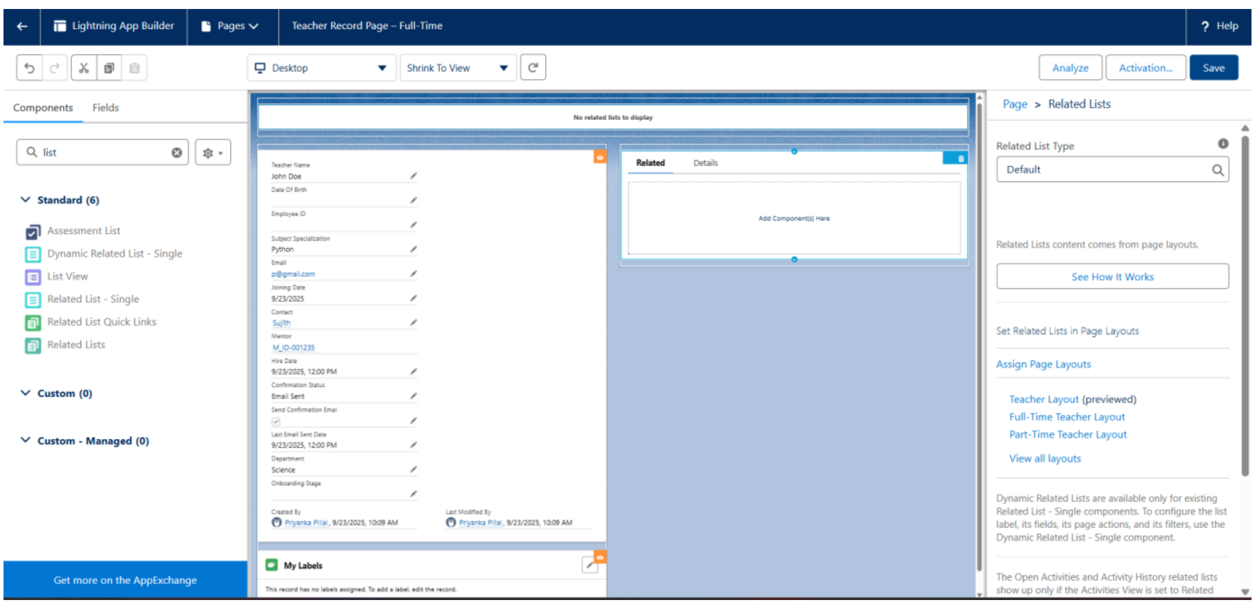


2. Record Pages (Teacher & Student Views)

- Teachers need quick access to class resources and student borrowing history. Students need an easy way to browse and request library materials. Two record pages were created with appropriate components:
- Library Resource – Teacher View: shows resource summary, related student borrow/return history, and an upcoming assignments/resources LWC.
- Library Resource – Student View: optimized for quick browsing, book request/renewal, and viewing personal borrowing history.
- Activated per profile/record type so each user sees the correct page automatically.



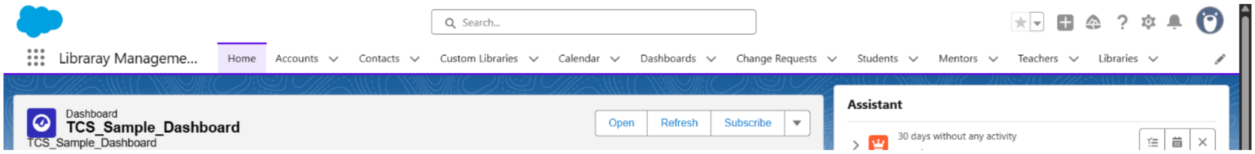
Library(Record Page)



Teacher(Record Page)

3. Tabs

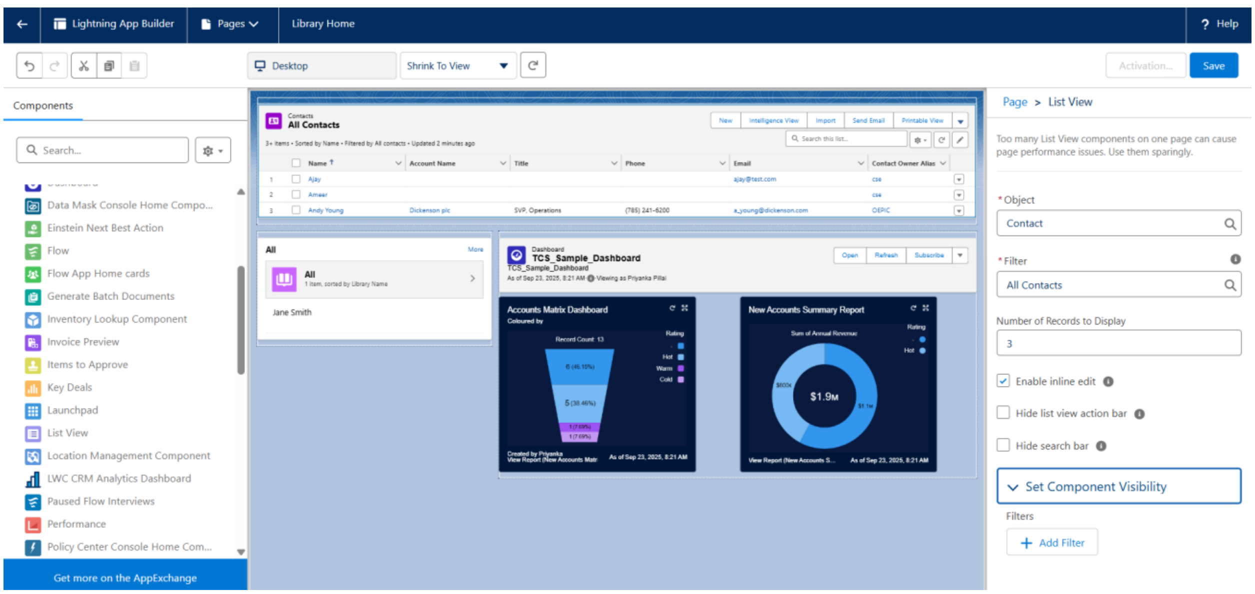
All major library objects (**Books/Resources, Borrow Requests, Students/Teachers, Classes**) were exposed as tabs in the **Library Management Lightning App** so Teachers, Students, and Librarians can navigate quickly to the areas they need.



Tabs in App

4. Home Page Layouts

A custom **Library Management Home Page** was built that displays dashboard components such as **Upcoming Book Returns, Recent Borrow Requests, Popular Books**, and **Overdue Items Report Chart** to give Teachers, Students, and Librarians a real-time view of library activity.



Library Home Page Layout

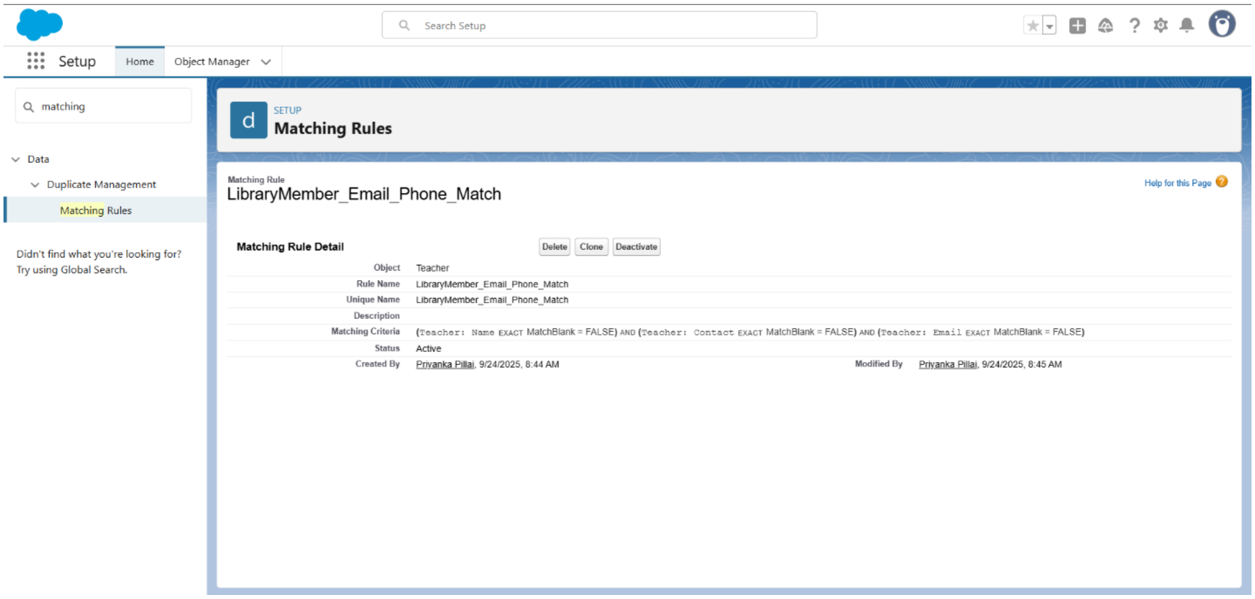
Phase 8: Library Management Data Management & Deployment

1. **Create the Matching Rule**

- **Setup → Quick Find → Matching Rules → New Rule**
- **Object:** Teacher
- **Rule Name:** LibraryMember_Email_Phone_Match
- **Add Matching Criteria:**
    - Field = **Email** → Matching Method = **Exact**

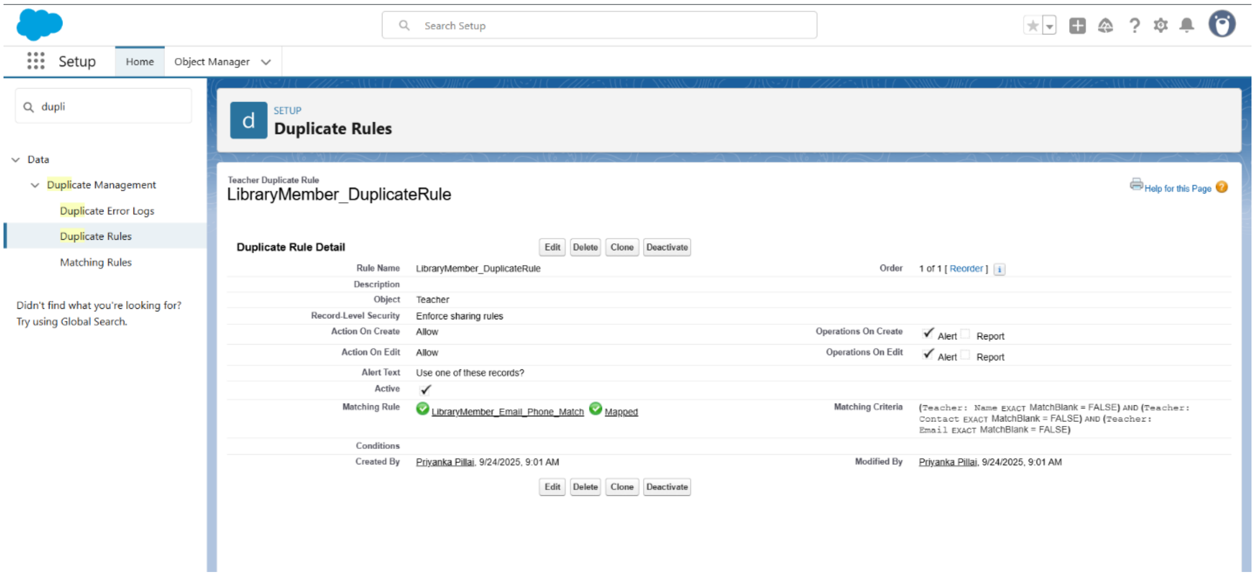
    ○ Click **Add Row** → Field = **Phone** → Matching Method = **Exact**

- Click **Save**
- Click **Activate**



2) Create the Duplicate Rule

- **Setup → Quick Find → Duplicate Rules → New Rule**
- **Object: Teacher**
- **Rule Label: LibraryMember_DuplicateRule**
- **Under Matching Rules: click Add and select the LibraryMember_Email_Phone_Match matching rule you activated in the previous step.**
- **Action on Create: choose Alert (recommended while testing so users are warned but not blocked).**
- **Action on Edit: choose Alert.**



Step 2: Data Backup – Library Management

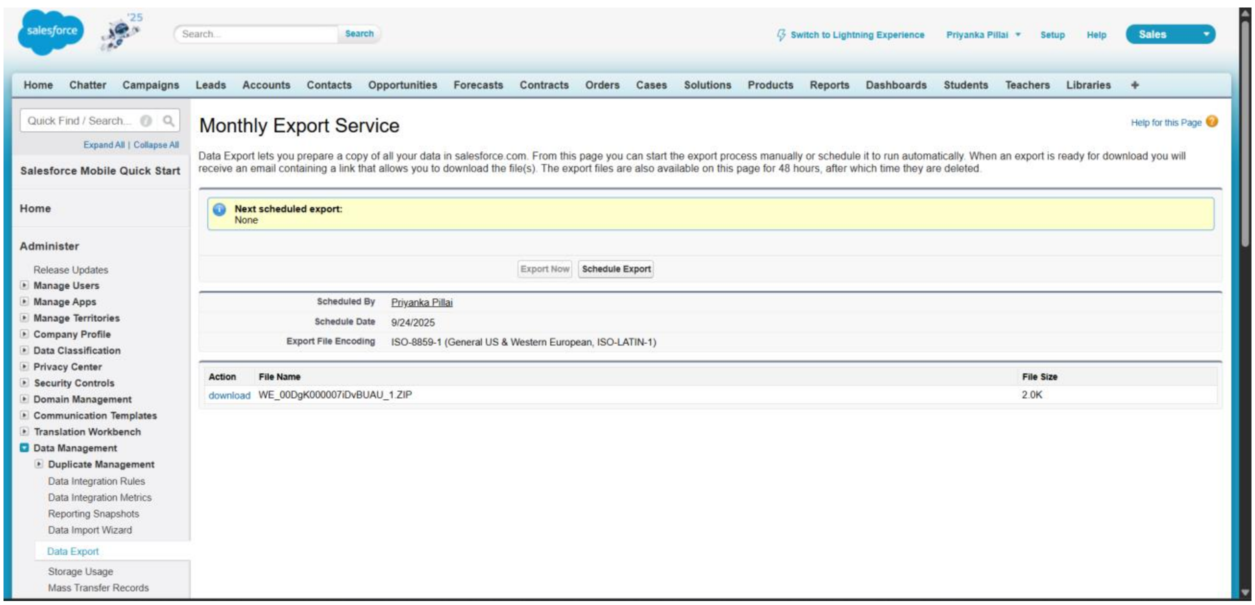
Goal: Regularly back up all key Library Management data (Teachers, Students, Books, Borrow Requests, etc.) for safety and compliance.

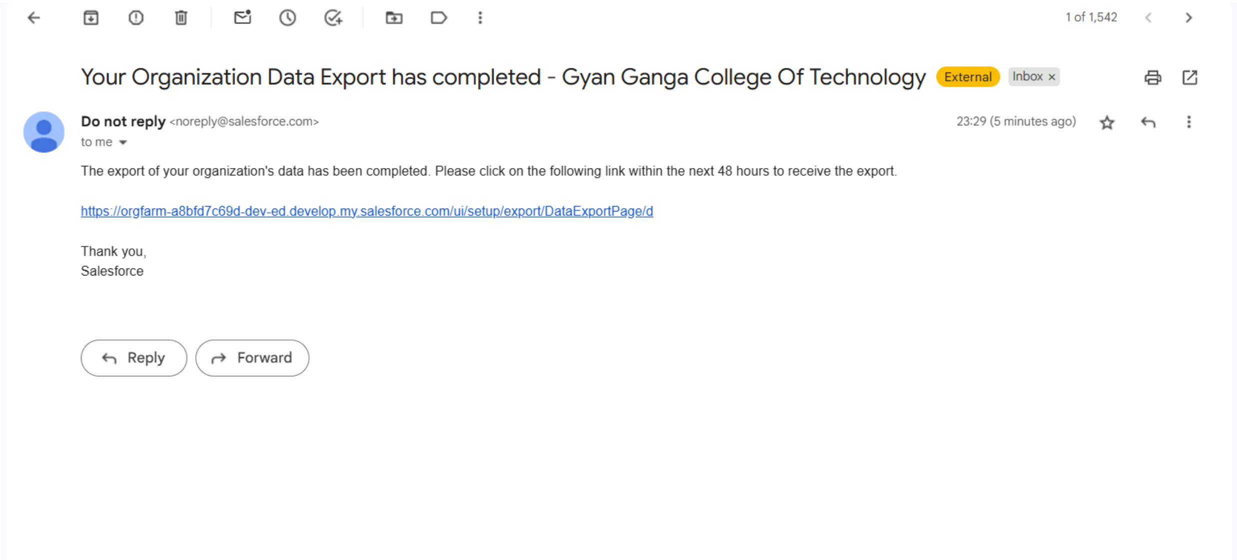
Steps

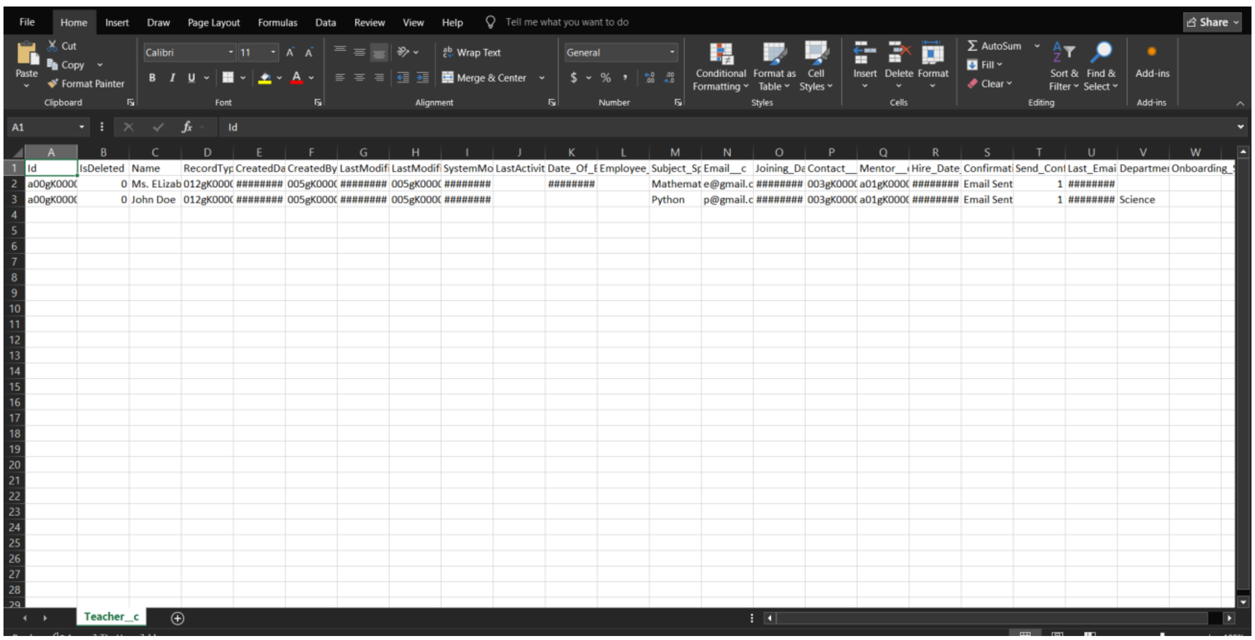
- **Go to Setup.**
- In **Quick Find**, type **Data Export** → click **Data Export**.
- **Choose one:**
  - **Export Now** → run a one-time backup.
  - **Schedule Export** → set weekly or monthly backups.
- **Select the objects you want:**
  - Library Members (Teachers/Students)
  - Books/Resources
  - Borrow Requests / Returns
  - Classes / Courses

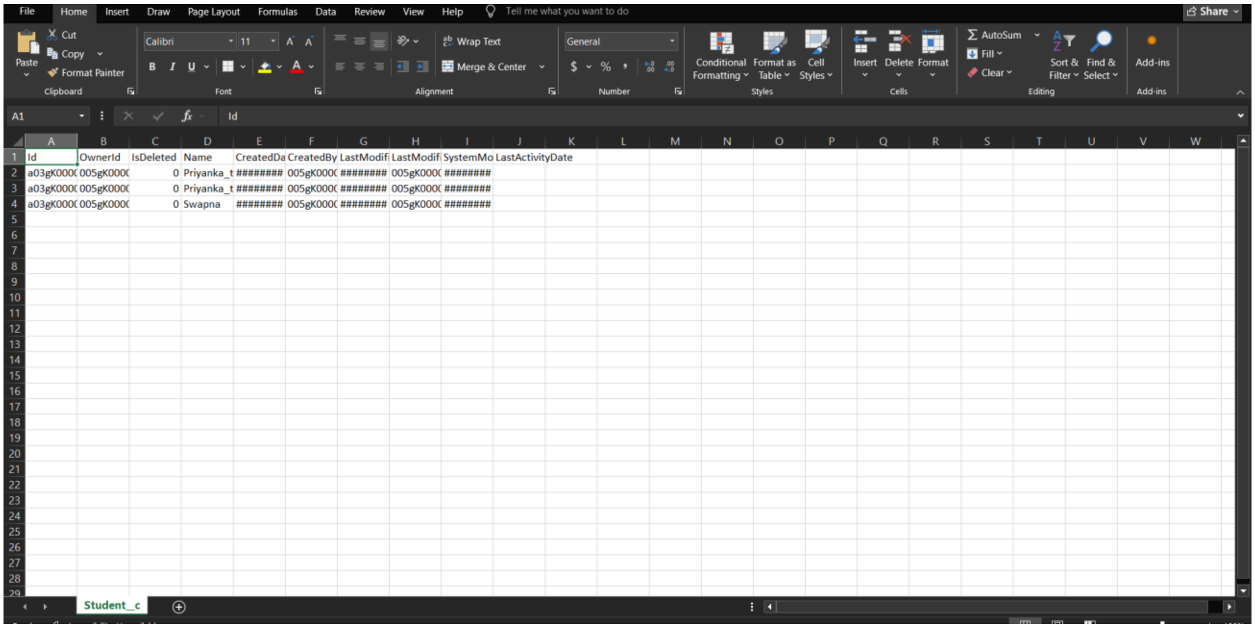
- Librarians (Users)
- Any other standard objects you use (e.g., Users, Attachments).

- Click **Start Export** (for immediate) or **Save** (for scheduled).
- **Wait** — Salesforce will email you when the backup is ready.
- **Download** the .zip file from the export page → extract CSV files.
- **Store the backup securely** (encrypted drive, secure server, or trusted cloud storage).




1 of 1,542
Your Organization Data Export has completed - Gyan Ganga College Of Technology
External
Inbox
Do not reply <noreply@salesforce.com>
to me
23:29 (5 minutes ago)
The export of your organization's data has been completed. Please click on the following link within the next 48 hours to receive the export.
https://orgfarm-a8bfd7c69d-dev-ed.develop.my.salesforce.com/ui/setup/export/DataExportPage/d
Thank you,
Salesforce
Reply
Forward


File
Home
Insert
Draw
Page Layout
Formulas
Data
Review
View
Help
Tell me what you want to do
Share
A1
Id
Id | IsDeleted | Name | RecordTyp | CreatedDa | CreatedBy | LastModifi | LastModifi | SystemMo | LastActivit | Date_Of_E | Employee_ | Subject_Sp | Email__c | Joining_Da | Contact__ | Mentor__ | Hire_Date_ | Confirmati | Send_Con | Last_Emai | Departmer | Onboarding_
a00gK000(| 0 | Ms. ELizab | 012gK000(| ######## | 005gK000(| ######## | 005gK000(| ######## | | ######## | | Mathemat | e@gmail.c | ######## | 003gK000(| a01gK000(| ######## | Email Sent | 1 | ######## | |
a00gK000(| 0 | John Doe | 012gK000(| ######## | 005gK000(| ######## | 005gK000(| ######## | | | | Python | p@gmail.c | ######## | 003gK000(| a01gK000(| ######## | Email Sent | 1 | ######## | Science |
Teacher__c


File Home Insert Draw Page Layout Formulas Data Review View Help
Tell me what you want to do
Share
A1
Id
Id OwnerId IsDeleted Name CreatedDa CreatedBy LastModifi LastModifi SystemMo LastActivityDate
a03gK000(005gK000(0 Priyanka_t ######## 005gK000(######## 005gK000(########
a03gK000(005gK000(0 Priyanka_t ######## 005gK000(######## 005gK000(########
a03gK000(005gK000(0 Swapna ######## 005gK000(######## 005gK000(########
Student__c

Library Management **Phase 9:**

Reporting, Dashboards & Security Review

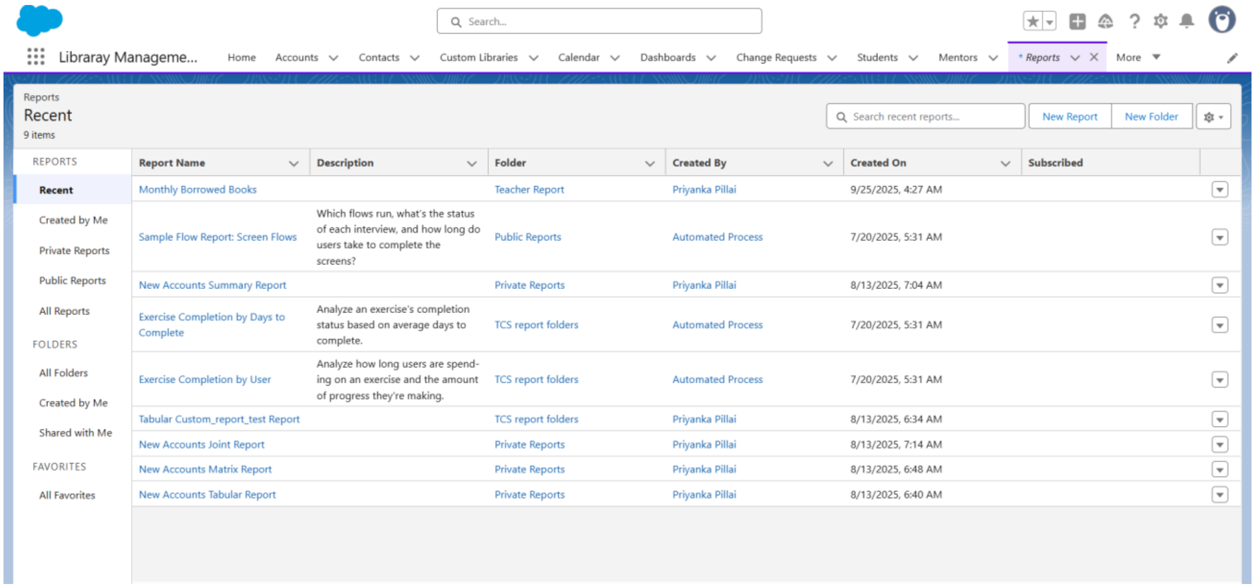
Step 1: Reporting

A **Report** is a list or summary of records in Salesforce that helps you analyze library data.

- Reports let you **filter, sort, group, and summarize** your Library Management data.
- **Example:** You can create a report to see all books borrowed this month or the top books borrowed by Teachers vs. Students.

Steps

- **Go to the App Launcher → search "Reports" → click Reports.**
- **Click "New Report" - Monthly Borrowed Books.**
- **Select the Report Type** - Teacher
- **Click Continue.**
- **Add Filters**
- **Add Columns**
- **Group Data (optional)**
  - *Example:* Group Borrow Requests by Book or by Member (Teacher/Student).
- **Summarize (optional)**
  - Add totals, averages, or counts (like total borrowed books per Teacher).
- **Click Save & Run** → Give the report a name→ Choose a folder to save it in.



Step 2: Dashboards

What it is:
A Dashboard is a visual board that displays charts, numbers, and tables built from your Reports. It helps Teachers, Librarians and Management instantly see performance trends (for example: most borrowed books, top students, pending returns).

Dashboard Components You Can Add

- **Charts** – Bar, Pie, Line charts to show trends.
- **Gauges** – Show progress toward a target (e.g. % of books returned on time).
- **Metrics** – One key number (e.g. total borrowings this month).
- **Tables** – Raw report data in a grid.

Steps (Click by Click)

- **Go to the App Launcher → type "Dashboards" → click Dashboards.**
- **Click "New Dashboard".**
- **Enter Dashboard Name** – e.g. Library Performance Dashboard or Teacher Engagement Dashboard.
- **Choose a Folder** (select the folder you created earlier for Reports/Dashboards).

- **Click “Create”.**
- **Click “+ Component”** (to add your first chart, metric or table).
- **Select a Report** you already built (for example: “Books Borrowed This Month” or “Top Students”).
- **Choose Visualization Type:**
- Bar Chart
- Pie Chart
- Gauge
- Metric
- Table

Configure Data Display:

- Choose which fields, groupings, and filters you want for this chart or metric.

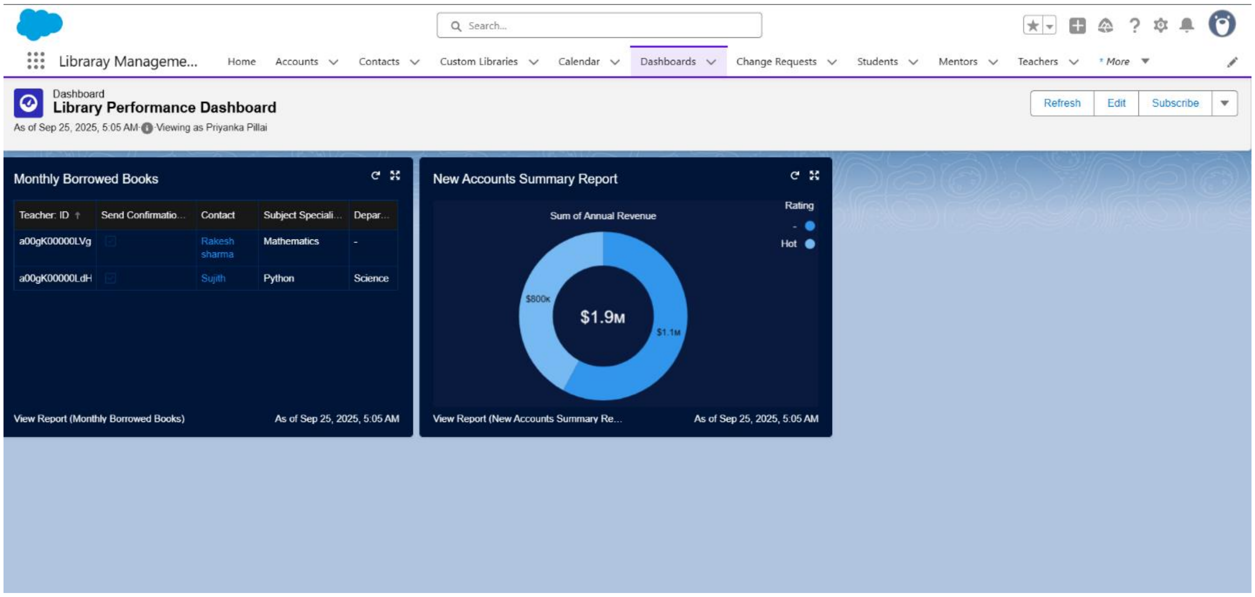
Click “Add”.

Repeat “+ Component” for more charts/metrics if needed.

Click “Save” → then “Done”.

Click “Refresh” anytime to pull in the latest data.

Result:
Your dashboard now shows live numbers and visuals from your Salesforce reports for Teachers, Students and Librarians.



Step 3: Field-Level Security

What it is:

Field-Level Security controls who can see or edit individual fields on a record.

It protects sensitive data like Teacher Contact, Student DOB, Library Fines, Book Costs, etc., so only the right roles can view or update them.

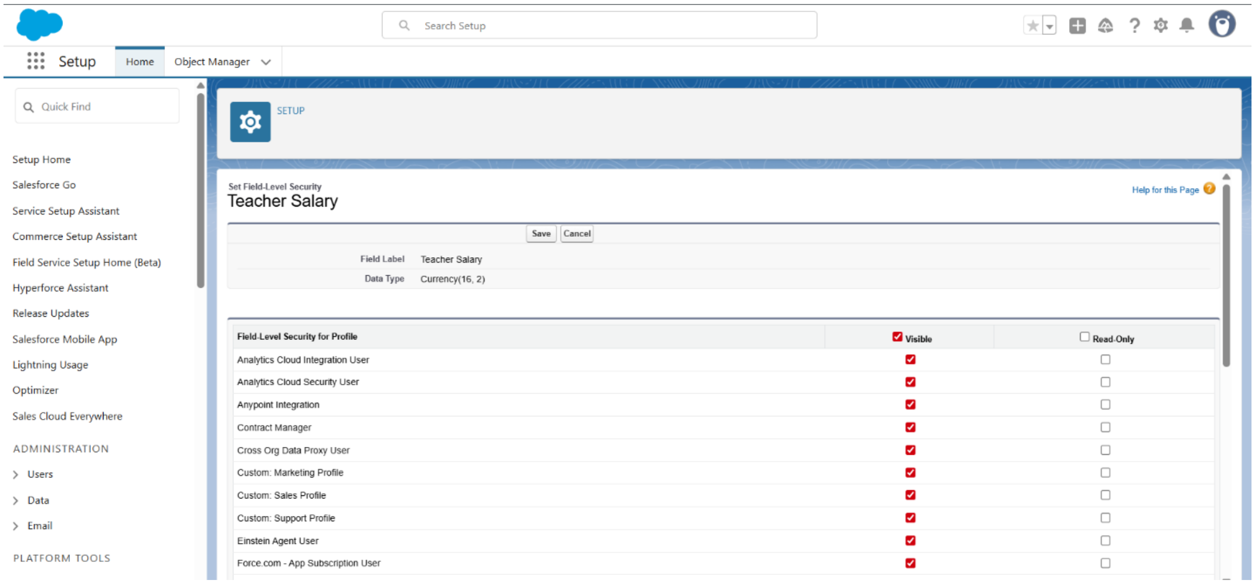
Steps (Click by Click)

- Go to Setup → Object Manager.
- Select your Object (example: Student or Teacher or Book).
- Click “Fields & Relationships”.
- Click the field you want to protect (example: Student DOB).
- Click “Set Field-Level Security”.
- For each Profile:
- Visible → can see the field.
- Read-Only → can see but cannot edit.
- Hidden → cannot see at all.
- (Example: For Student DOB → Visible to Teacher & Librarian profiles, Hidden for Student profile.)
- Click “Save”.

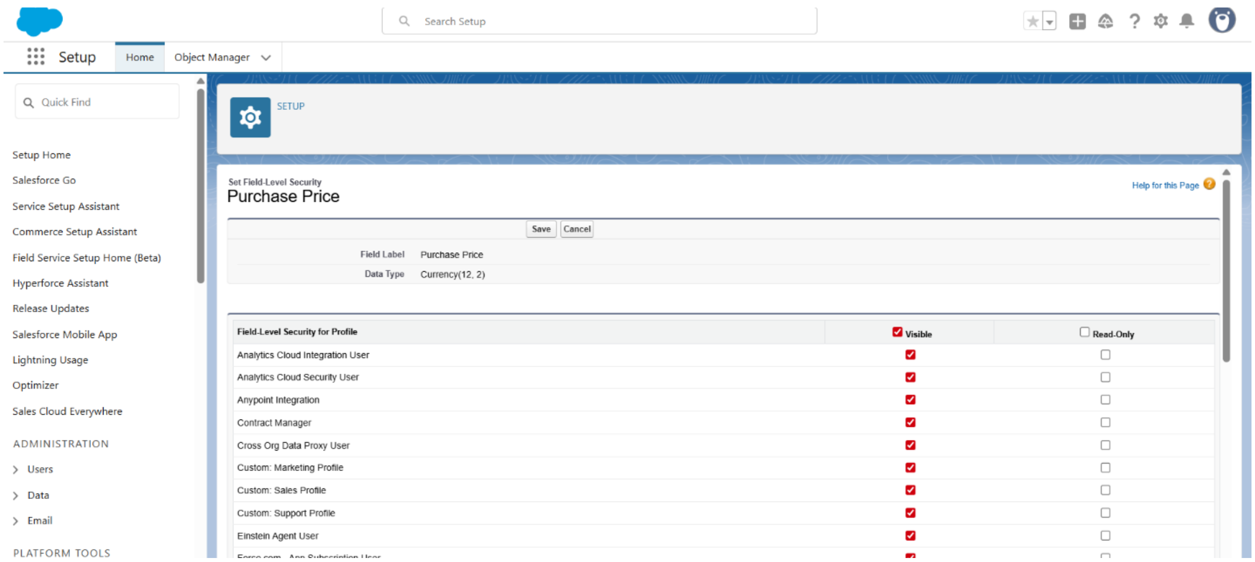
- Repeat steps 4–7 for other sensitive fields like Teacher Salary, Book Purchase Price, Library Fine Amount, etc.

- Result:
- Only authorized roles (Teacher, Librarian, Admin) can view or edit those sensitive fields. Students or other profiles see nothing or read-only according to your settings.

- Setup → Object Manager → Teacher → Fields & Relationships → New.
- Choose Field Type = Currency → Next.
- Field details:
- Field Label: Teacher Salary
- Field Name: Teacher_Salary__c
- Length / Decimal Places: choose e.g. 16,2 (or your org standard)
- Help Text: Annual salary in local currency. Visible to Admin only.
- Description: Stores salary for payroll/audit
- Click Next.
- Set Field-Level Security (during creation)
- On the Set Field-Level Security screen (shown while creating), pick visibility per profile:
- visible for all
- Click Next.
- Add to Page Layouts
- Choose page layouts where the field appears (usually Admin layouts only) → Save.



- Setup → Object Manager → Student → Fields & Relationships → New.
- Field Type = Currency → Next.
- Field details:
- Field Label: Purchase Price
- Field Name: Purchase_Price__c
- Length/Decimals: e.g., 12,2
- Help Text: Cost price of the book — visible to Librarians & Admins.
- Click Next.
- Set Field-Level Security (during creation)
- Choose visibility: all
- Next → Add to relevant Page Layout(s) → Save



- Setup → Object Manager → Student → Fields & Relationships → New.
- Field Type = Currency → Next.
- Field details:
- Field Label: Fine Amount
- Field Name: Fine_Amount__c
- Help Text: Fine charged for overdue returns — visible to Librarians only.
- Click Next.
- Set Field-Level Security
- Visibility choices: for all
- Next → Add to Page Layout(s) → Save.