CampusConnect CRM – Student Club & Event Management System(Phase-1)

Phase 1: Problem Understanding & Industry Analysis

Industry: Education (Campus Clubs & Events)
Project Type: Salesforce CRM Implementation (Admin + Developer)
Target Users: Students, Club Leaders, Faculty Coordinators, Admin

---

Problem Statement

Student clubs and events on campus are currently managed using scattered tools like Google Forms, WhatsApp groups, and Excel sheets. This leads to:

- Missed or delayed event updates.
- Duplicate or invalid registrations.
- No centralized way to track student participation.
- Manual effort for faculty coordinators to generate reports.

Goal

Implement a Salesforce CRM that:

- Centralizes student, club, and event data.
- Automates event registration and reminders.
- Tracks attendance and generates participation history.
- Provides dashboards for real-time engagement analytics.
- Introduces smart, innovative, and gamified experiences for students.

Requirement Gathering

Business Needs

- Centralized platform for managing clubs, students, and events.
- Automated registration and event reminders.
- Streamlined attendance & feedback tracking.

- Dashboards for faculty and coordinators to monitor student engagement.
- Innovative features like mood-based recommendations, NFT badges, and dynamic pricing to make events more engaging.

Functional Requirements

- **Student Object:** Capture student details (Year, Branch, Skills, Mood).
- **Event Object:** Fields for Event Name, Date, Venue, Capacity, Club, Pricing Tier.
- **Event Registration Object:** Links Students to Events.
- **Automation:** Email/SMS reminders before event day.
- **Reports:** Participation trends, event attendance, club performance.
- **Friend-Match Logic:** Suggests events based on friends' participation.
- **NFT Badges:** Auto-generated collectibles for attended events.

Non-Functional Requirements

- Mobile-friendly (via Salesforce Mobile App).
- Role-based access (Students vs Leaders vs Faculty).
- Secure and scalable for 1000+ student records.
- Easy-to-use interface for non-technical users.

Stakeholder Analysis

Stakeholder	Needs / Responsibilities
Students	Register for events, receive reminders, view participation history, collect NFT badges, get event suggestions.
Club Leaders	Create/manage events, track registrations, configure pricing tiers, issue NFT badges, view attendance.
Faculty	Oversee multiple clubs, approve budgets, track student engagement, review reports.
Admin	Setup org, users, profiles, roles, permissions, and manage data security.

Business Process Mapping

Current Manual Process

- Club shares Google Form links via WhatsApp.

- Student data manually compiled into Excel.
- Event reminders sent manually by club leaders.
- Attendance tracked via paper sheets.
- Reports created monthly using Excel.

Proposed Salesforce Process

1. Student registers via Salesforce form → captured as Event Registration Record.
2. Auto-confirmation email/SMS sent to the student.
3. Salesforce Flow sends reminders 1 day before the event.
4. Attendance tracked digitally using custom objects.
5. Participation history + NFT badges shown in Student Profile.
6. Mood + Friend-Match engine suggests events in real time.
7. Dynamic pricing automatically adjusts fees based on registration count.
8. Dashboards auto-refresh for real-time engagement insights.

Industry-Specific Use Cases

- **Mood-Based Event Suggestions:** Students update their mood (Happy, Stressed, Curious) → Salesforce recommends matching events (Yoga for stressed, Hackathon for curious, Fest for excited).
- **Smart Friend-Match:** When registering, system suggests events that friends or classmates are also attending → boosting participation.
- **Dynamic Pricing / Early Bird Discounts:** First 20 signups free, next 30 at discounted price, last batch at full price → auto-managed in Salesforce.
- **Event NFTs / Digital Collectibles:** Students receive unique NFT badges after attending events. Collecting milestones unlocks rewards like "Campus Influencer."
- **Student Engagement Leaderboard:** Rankings based on event participation, mood badges, and NFT collections.
- **Automated Certificates:** Issued digitally after successful attendance.
- **Budget Approval Workflow:** Faculty reviews & approves club event budgets via Salesforce Approval Process.
- **Multi-Club Membership Management:** Students can belong to multiple clubs at once with participation history tracked.

AppExchange Exploration

- **Eventbrite Sync** → Seamless event registration integration.
- **FormAssembly** → Custom student registration forms.
- **SurveyMonkey** → Event feedback surveys.
- **Conga Composer** → Auto-generate participation certificates.
- **Blockchain/NFT API Integration** → Event NFT badge distribution.

Phase 1 Summary

- Problem statement clarified.
- Requirements (business, functional, non-functional) documented.
- Stakeholders identified with responsibilities.
- Current vs Proposed process mapped.
- Real-world education-specific + innovative use cases listed.
- Relevant AppExchange tools explored for faster implementation.

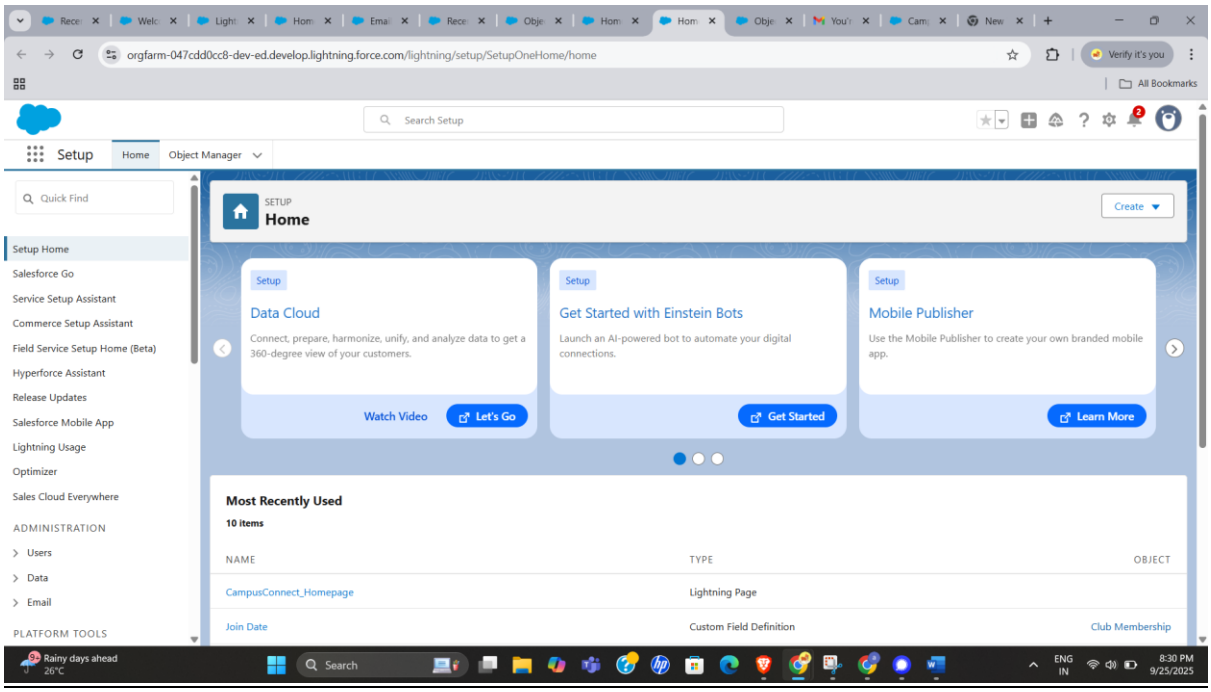
Phase 2 starts from next page

||

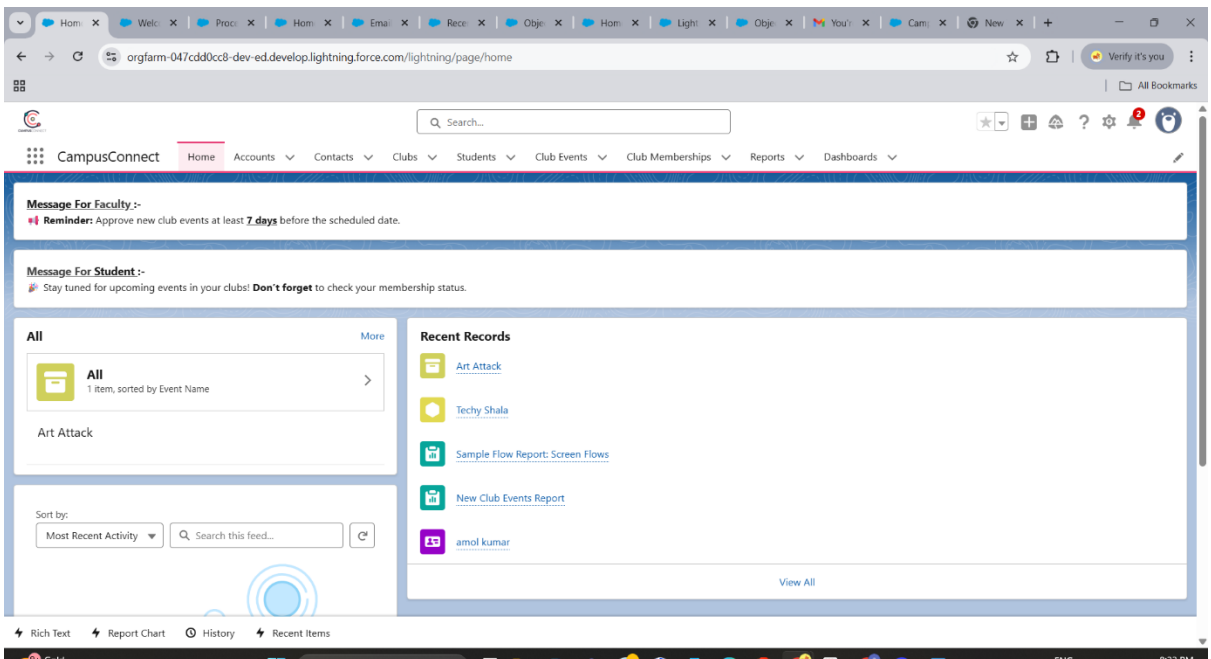
Phase 2: Org Setup & Configuration (Campus CRM + Student Club)

Screenshots:-

Developer Org Setup:-

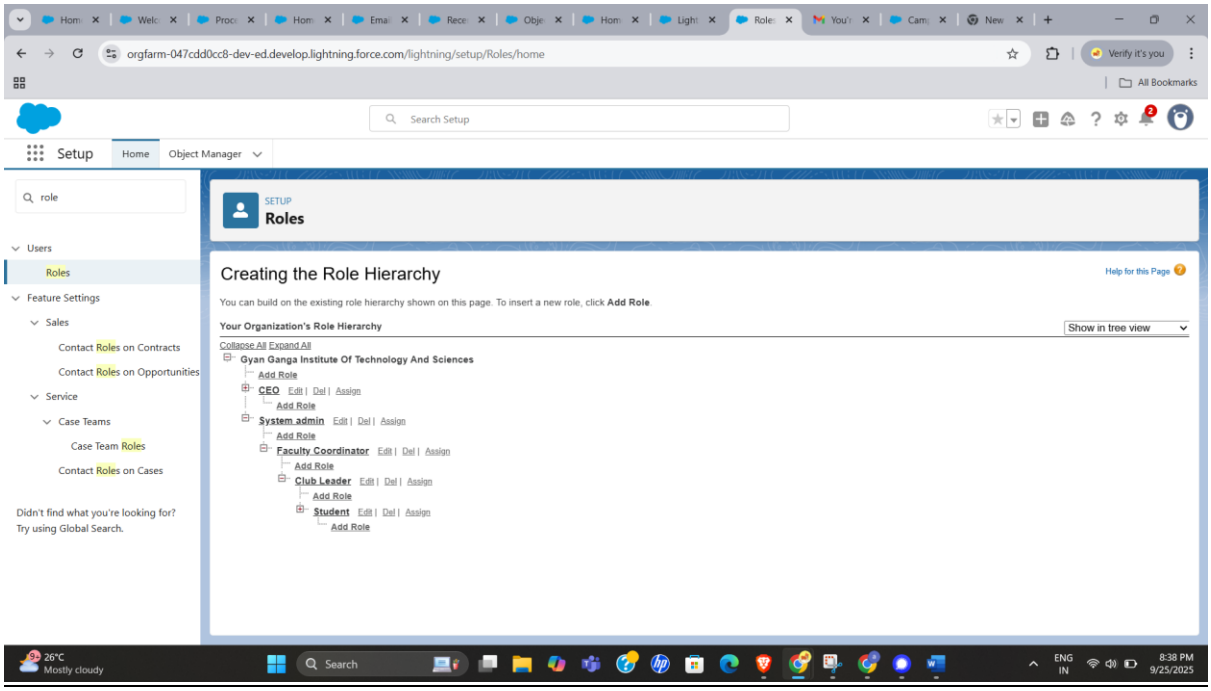


App Is Made By Lightning App Navigation:-



App Is Made By Lightning App Navigation:-

Role Hierarchy:-



Documentation:-

1. Salesforce Editions

- Use a **Developer Edition (Free Dev Org)**.
- Developer Org gives you:
    - 2 licenses (Admin + 1 User).
    - Full access to Objects, Apex, Flows, LWC.
    - Sufficient for Campus CRM prototype.

---

2. Company Profile Setup

- Setup → **Company Information**.
- Example details:
    - Company Name: *CampusConnect CRM*
    - Address: *University Campus, Jabalpur, MP*

- Currency: *INR*
- Default Time Zone: *Asia/Kolkata (IST)*

---

3. Business Hours & Holidays

- Setup → **Business Hours** → New:
  - Name: *Campus Office Hours*
  - Monday–Saturday: 9 AM – 8 PM IST
- Setup → **Holidays** → Add University holidays (Diwali, Republic Day, Annual Fest Week).

Automations like event reminders & mood-based suggestions can reference this.

---

4. Fiscal Year Settings

- Setup → **Fiscal Year**.
- Use **Standard Fiscal Year (April–March)** for academic cycle.
- If needed → Custom Fiscal Year to align with *Semester-based sessions*.

---

5. User Setup & Licenses

- Create 3–4 test users:
  i. **Admin (You)** – System Administrator.
  ii. **Faculty Advisor** – Standard User.
  iii. **Student Leader (Club Head)** – Custom profile with event management access.
  iv. **Student Member** – Basic student access, can register for events.

Helps demo different roles.

---

6. Profiles

- Profiles define what users can do.
- Example Profiles:
  - **System Administrator** → Full access.
  - **Faculty Profile** → Can monitor clubs & approve events.
  - **Club Leader Profile** → Can create/manage student events.
  - **Student Profile** → Can view & register for events only.

---

7. Roles

- Roles define **data visibility hierarchy**.
    - Admin (Top)
        - Faculty Advisor
            - Club Leader
                - Student Member

Role hierarchy ensures leaders can track their members' participation.

---

8. Permission Sets

- Create **extra permissions** beyond profiles.
    - *"Mood Event Dashboard Access"* → For students to see mood-based event suggestions.
    - *"Event Analytics Access"* → For faculty & club leaders to track participation.

---

9. OWD (Organization-Wide Defaults)

- Setup → Sharing Settings.
- Suggested defaults:
    - **Students** → Private.
    - **Events** → Public Read Only (anyone can see events).
    - **Mood Updates** → Private (confidential to student + admin).
    - **Club Data** → Controlled by Parent (Club Leader → Members).

---

10. Sharing Rules

- Add sharing for collaboration:
    - Faculty can see all club events.
    - Club Leaders can see all student registrations for their club.
    - Students can only see **their own registrations**.

---

11. Login Access Policies

- Setup → **Login Access Policies**.
- Allow users to grant temporary access to Admin for troubleshooting.

---

12. Dev Org Setup

- Stick to **Developer Org**.
- Optional: Use **Scratch Orgs (SFDX)** if showcasing DevOps.

---

13. Sandbox Usage

- Not available in free Dev Org.
- Can simulate by creating a **second Dev Org** as "Sandbox" If needed.

---

14. Deployment Basics

- Use **Change Sets** for moving customizations.
- Steps:
    i. Create Outbound Change Set (Dev Org).
    ii. Upload to another Org (Sandbox Demo Org).
    iii. Deploy.

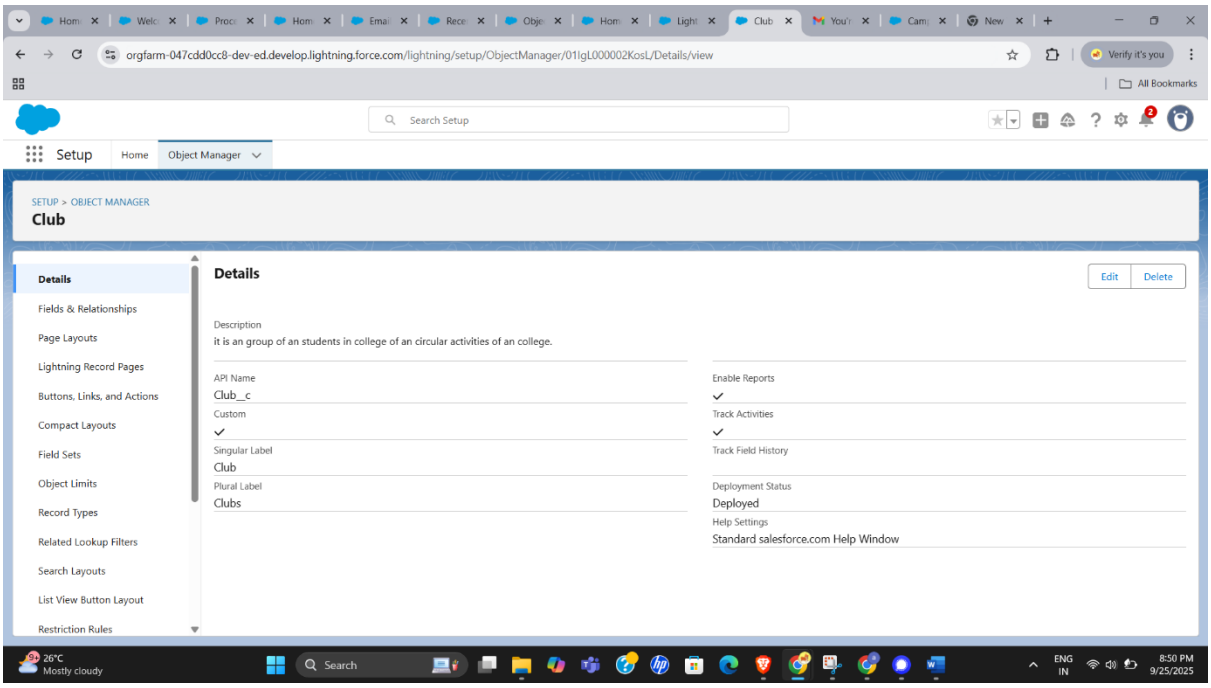
Phase 3 starts from next page

||

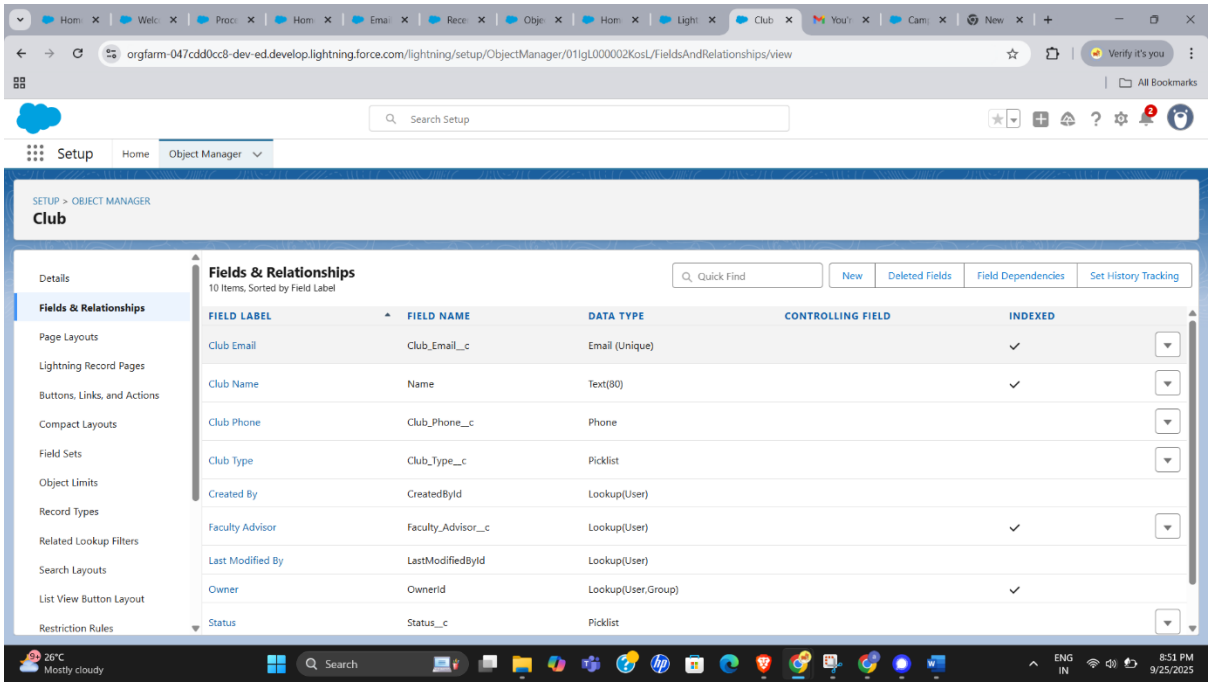
Phase 3: Data Modeling & Relationships(Campus CRM + Student Club)

Screenshots:-

Club Object:-

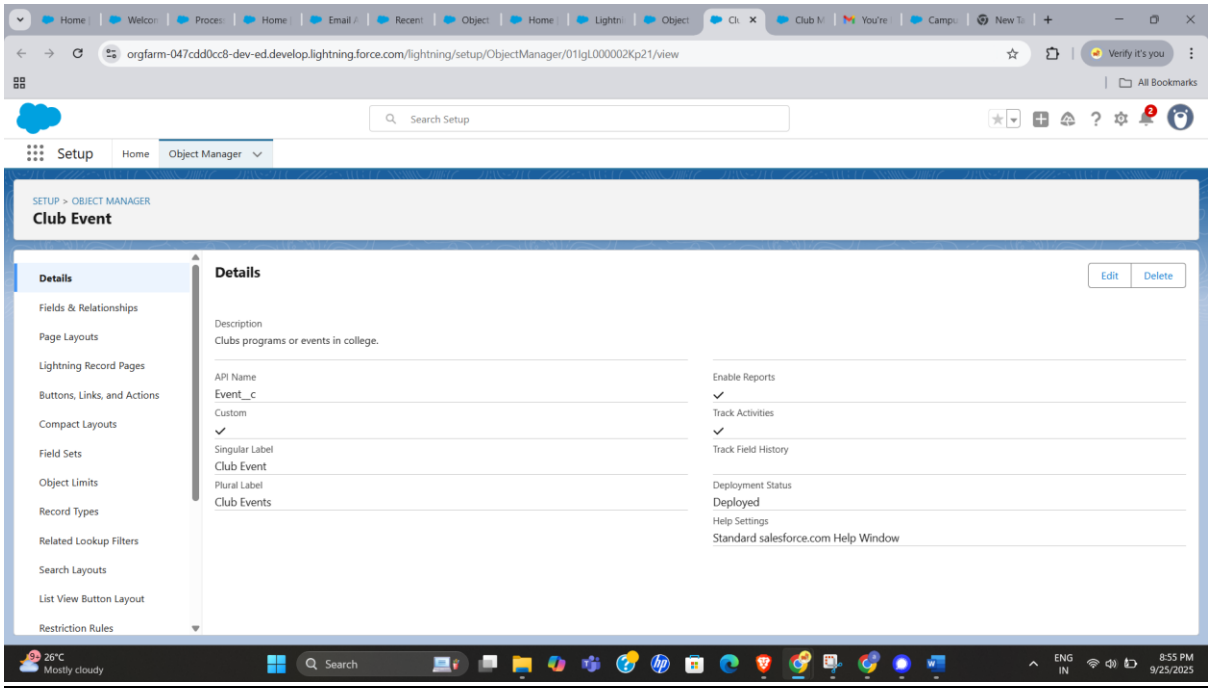


Fields:-

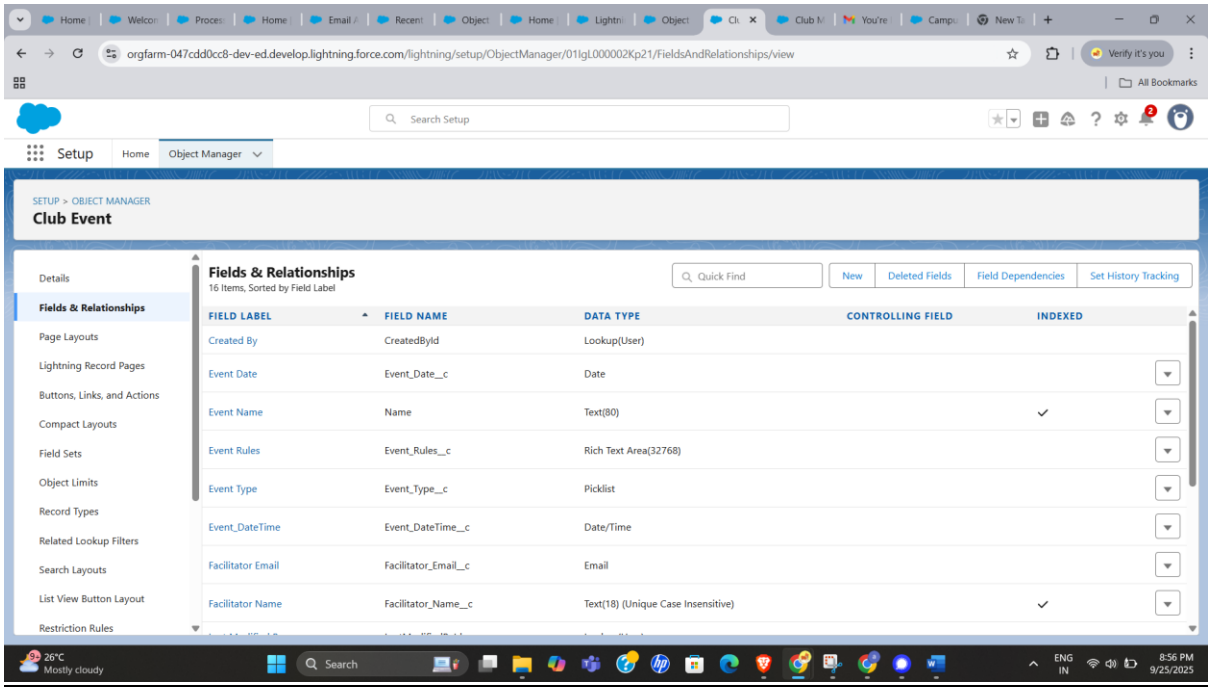


Club Event(custom object):-

Object:-

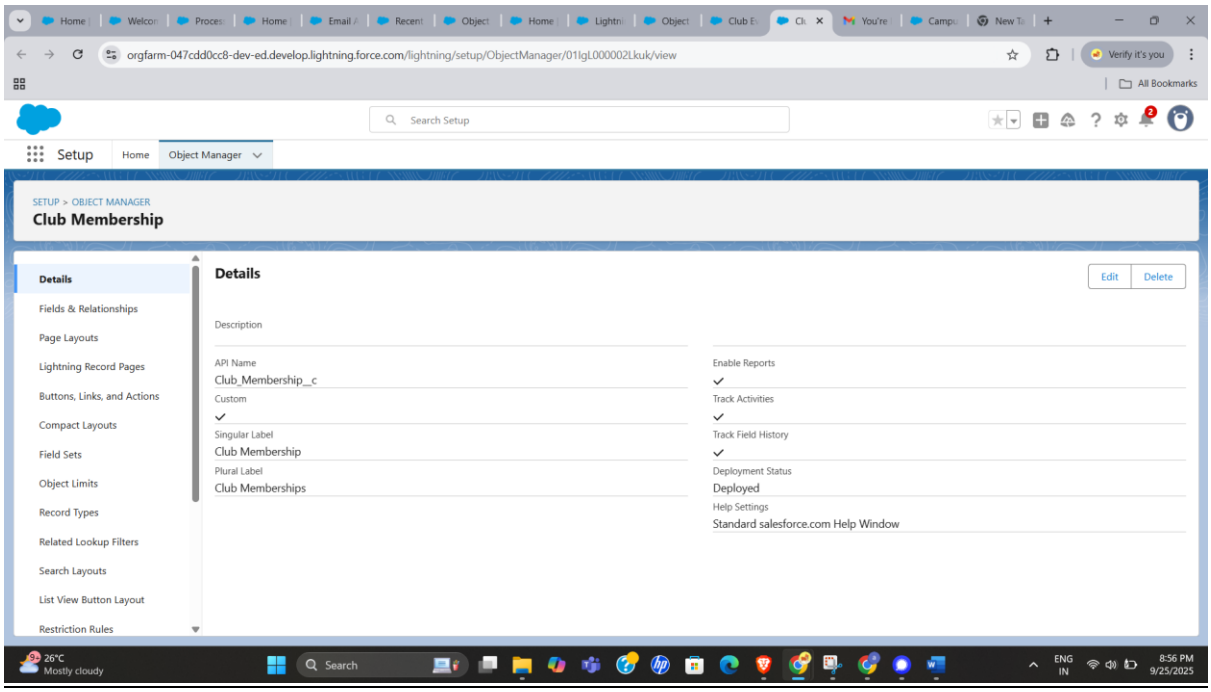


Fields & Relationships:-

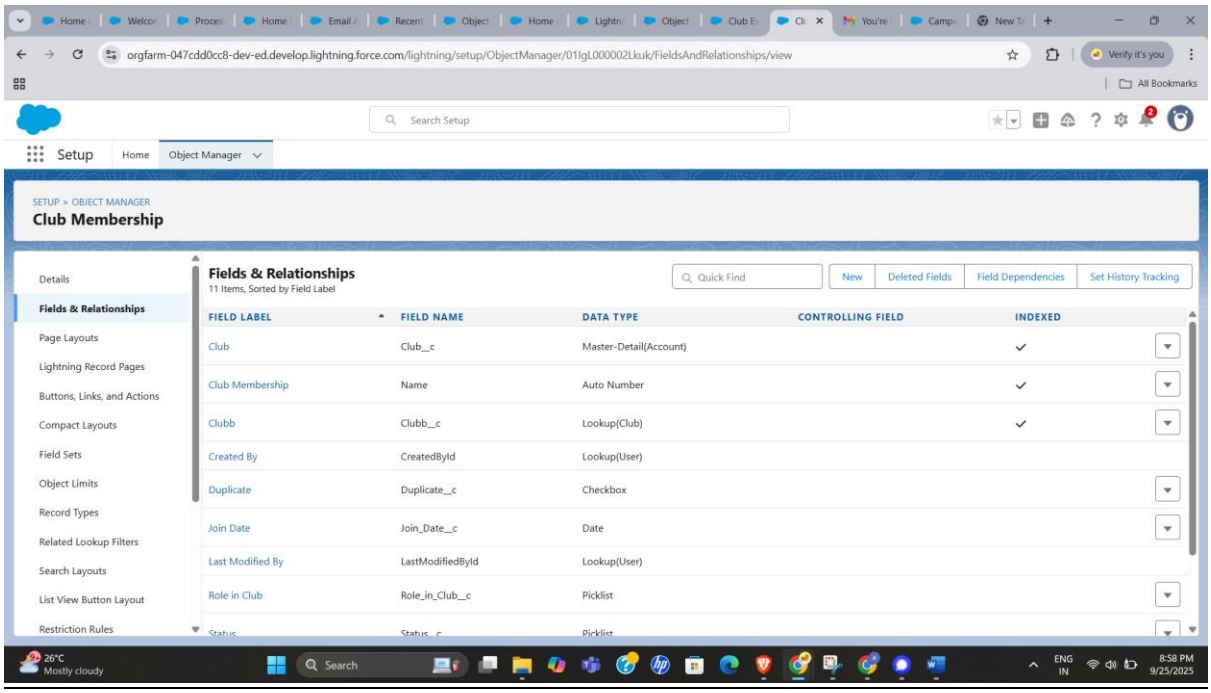


Club Membership (custom object):-

Object:-

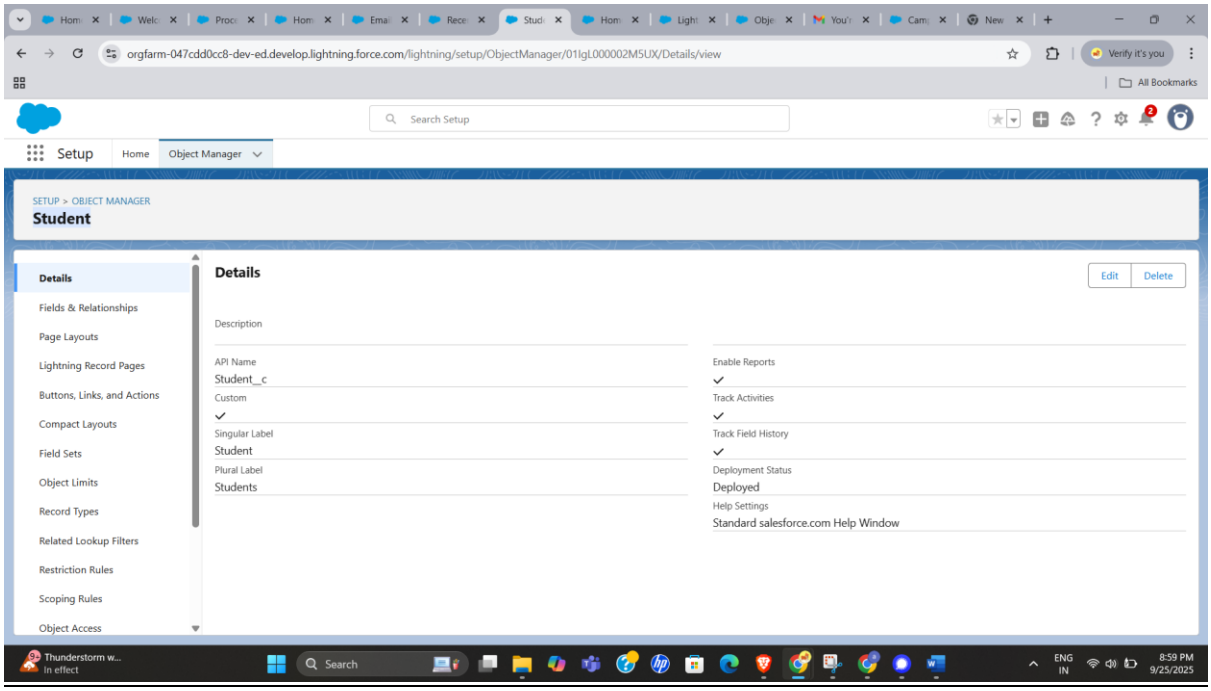


Fields & Relationships :-

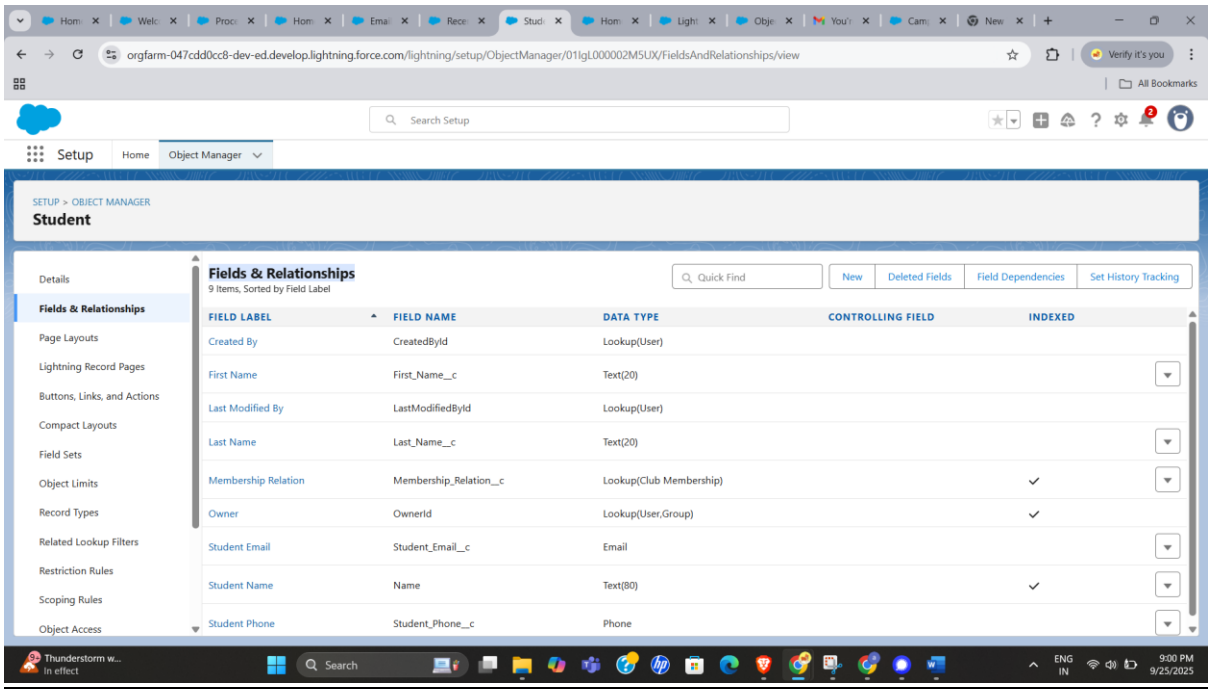


Student (custom object):-

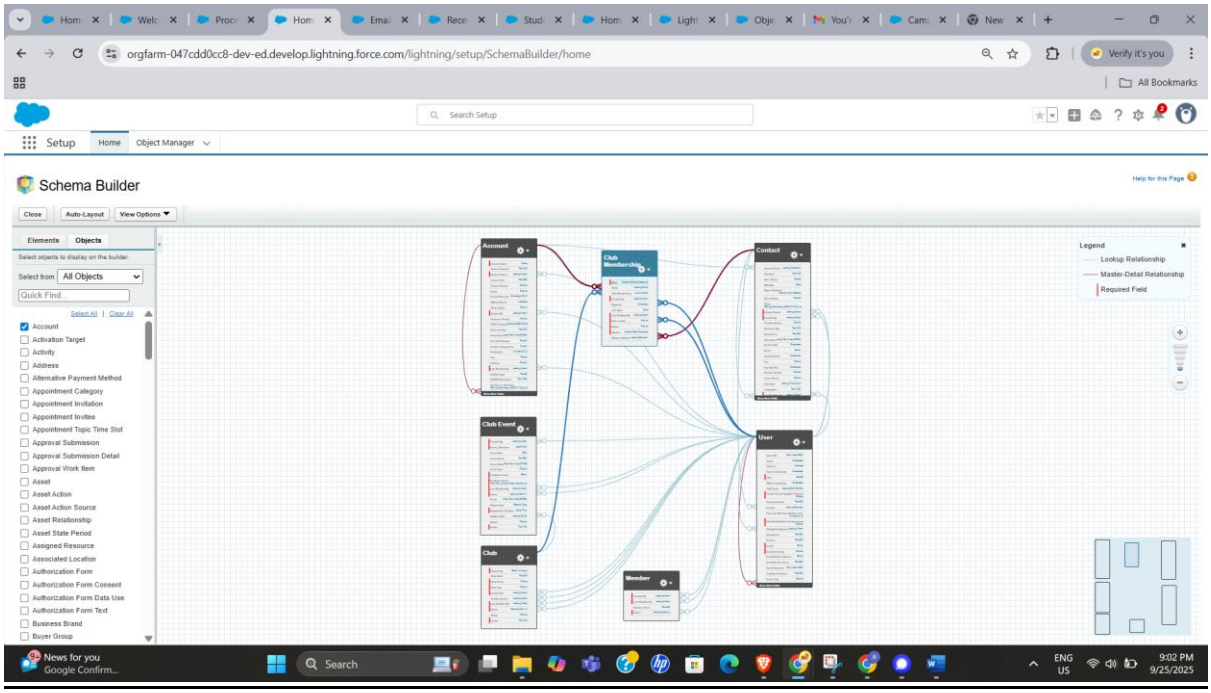
Object:-



Fields & Relationships:-



Schema Builder:-



Documentation:-

Standard & Custom Objects

- **Standard Objects Utilized:**
    - **Club :**Used to represent Clubs.
    - **Student:** Used to represent Students, linked to Clubs through memberships.
- **Custom Objects Created:**
    - **ClubMembership__c:** Junction object to link Students (Contact) and Clubs (Account).
    - **ClubEvent__c:** Custom object to represent Events hosted by Clubs.
- **Notes:** External Objects were not implemented in Developer Edition but are explained below for extension.

Fields

- **Student :**Name__c, Email__c, Phone__c,Address__c
- **Club :**Club_Type__c, Faculty_Advisor__c, Club_Email__c, Description__c
- **ClubMembership__c:** Role_in_Club__c, Joining_Date__c, Membership_Status__c, Notes__c
- **ClubEvent__c:** Event_DateTime__c, Registration_Deadline__c, Venue__c, Event_Type__c, Status__c,

---

Record Types

- **ClubEvent__c:**
    - Workshop
    - Seminar
    - Competition
    - Cultural Event
- **Notes:** Record types were created to allow separate page layouts and picklist variations for different scenarios.

Page Layouts

- **Club :**
    - Fields grouped as Club Information, Faculty Advisor, and Contact Info.
    - Related lists: Events, Memberships.
- **Student :**
    - Sections: Personal Details, Academic Info, Memberships, Event Participation.

    - Related list: Club Memberships.
- **ClubEvent__c:**
    - Event Details, Venue, Capacity, Status.
    - Related lists: Volunteers, Registered Students.
- **Volunteer__c:** Volunteer details with linked Student and Event.
- **ClubMembership__c:** Joining Date, Role, Membership Status, linked Student and Club.

Compact Layouts

- **Club:** Displayed Club Name, Club Type, Faculty Advisor, Active Status.
- **Student :**Displayed Name, Branch, Year, Skills.
- **ClubEvent__c:** Displayed Event Name, Date, Venue, Status.
- **ClubMembership__c:** Displayed Student Name, Role, Membership Status.

Schema Builder

- Used to visualize object relationships.
- Showed **Student** linked to **Club** through **ClubMembership__c (junction object)**.
- **ClubEvent__c** linked to **Club** , and **Volunteer__c** connected with both **Event and Student**.
- This schema diagram was saved and included in project documentation for clarity.

Lookup vs Master-Detail vs Hierarchical Relationships

- **Master-Detail Relationships Implemented:**
    - ClubMembership__c → Club
    - ClubMembership__c → Student
- **Lookup Relationships Implemented:**
    - ClubEvent__c → Club
- **Hierarchical Relationship:** Not applicable (only works on User object). A **hierarchical relationship** is a ranked structure with levels of authority.

Junction Objects

- **ClubMembership__c:** Primary junction object linking Students and Clubs.
- Enabled many-to-many relationship:

- One Student can join multiple Clubs.
- One Club can have multiple Students.

External Objects

- Not implemented in Developer Edition.
- Explained as a possible extension:
  - External Object “ExternalEvent__x” could be integrated using Salesforce Connect to sync events hosted by partner institutions.
  - Read-only access to external data while keeping CampusConnect records clean.

Key Outcomes of Phase 3

- Robust **data model** created to represent Clubs, Students, Memberships, Events, and Volunteers.
- **Many-to-many relationships** managed through ClubMembership junction object.
- Different **page layouts and record types** enhanced user experience for students, club leaders, and faculty.
- Compact layouts and schema visualization simplified navigation and reporting.
- Extended explanation included for External Objects and Hierarchical relationships, even if not implemented.

Phase 4 starts from next page

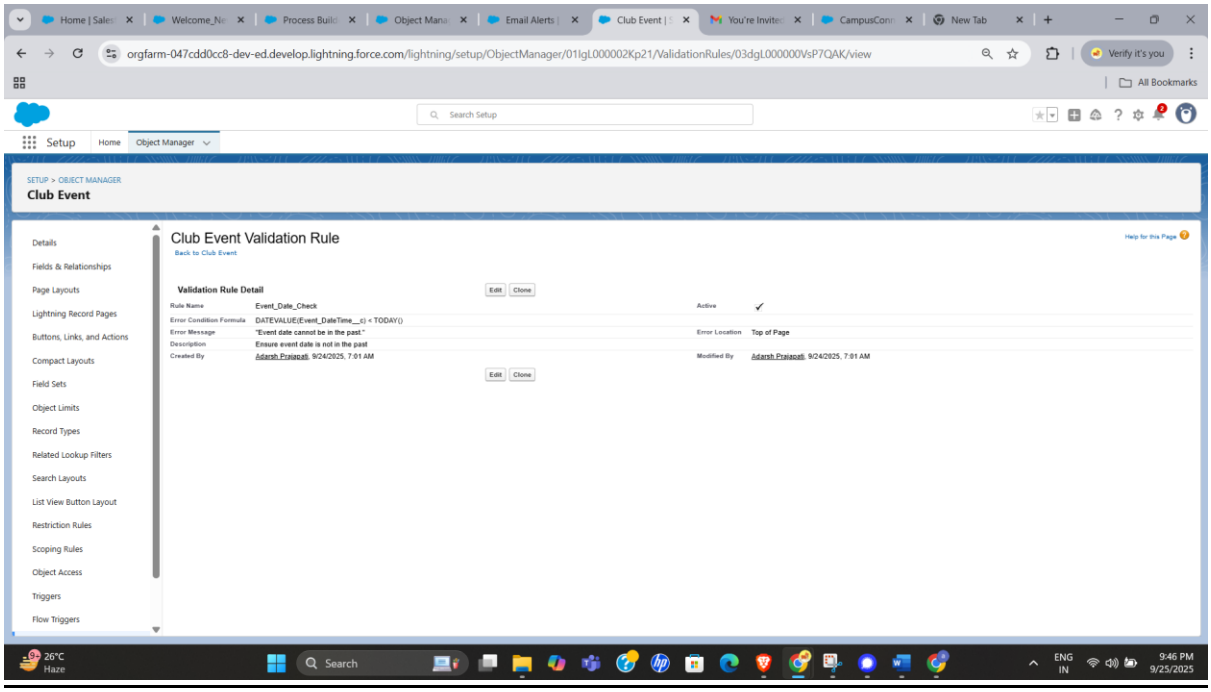
||

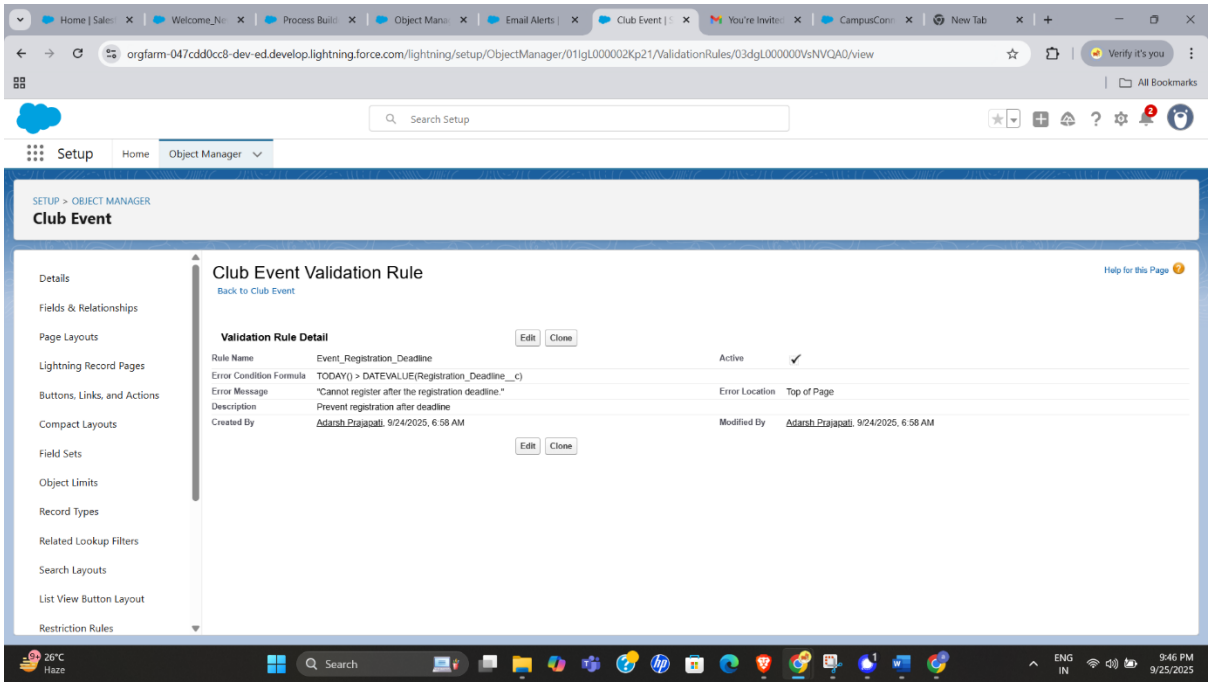
Phase 4: Process Automation (Campus CRM + Student Club)

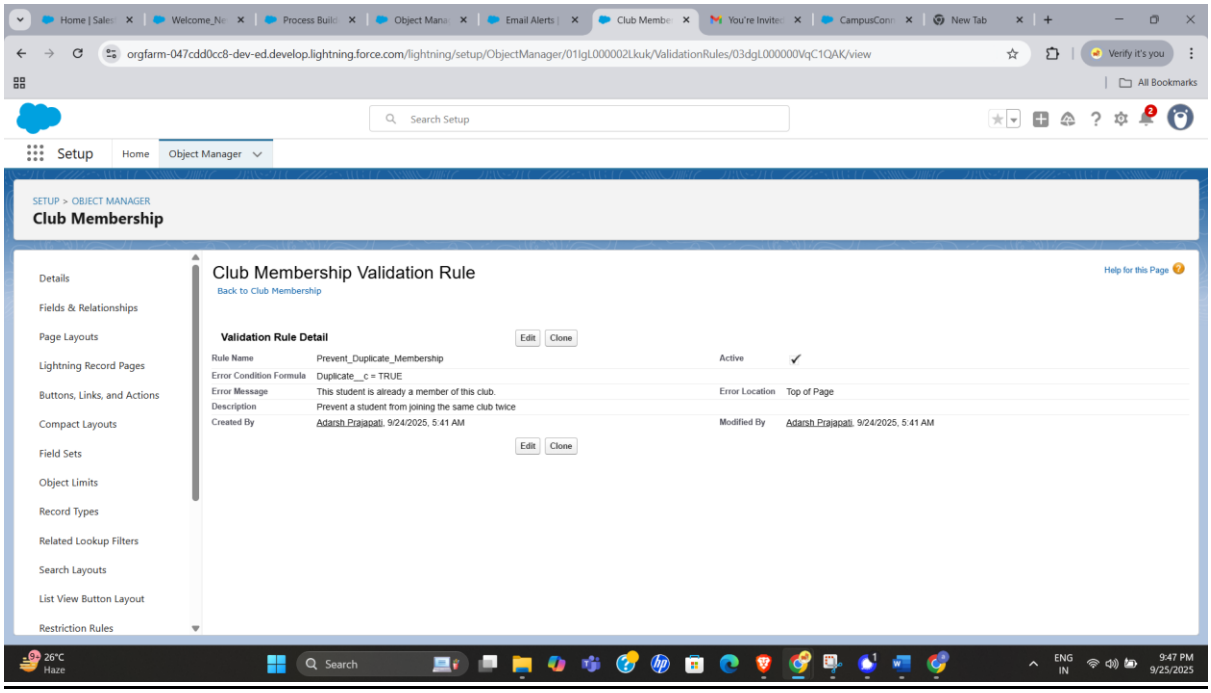
Screenshots:-

Validation Rules:-

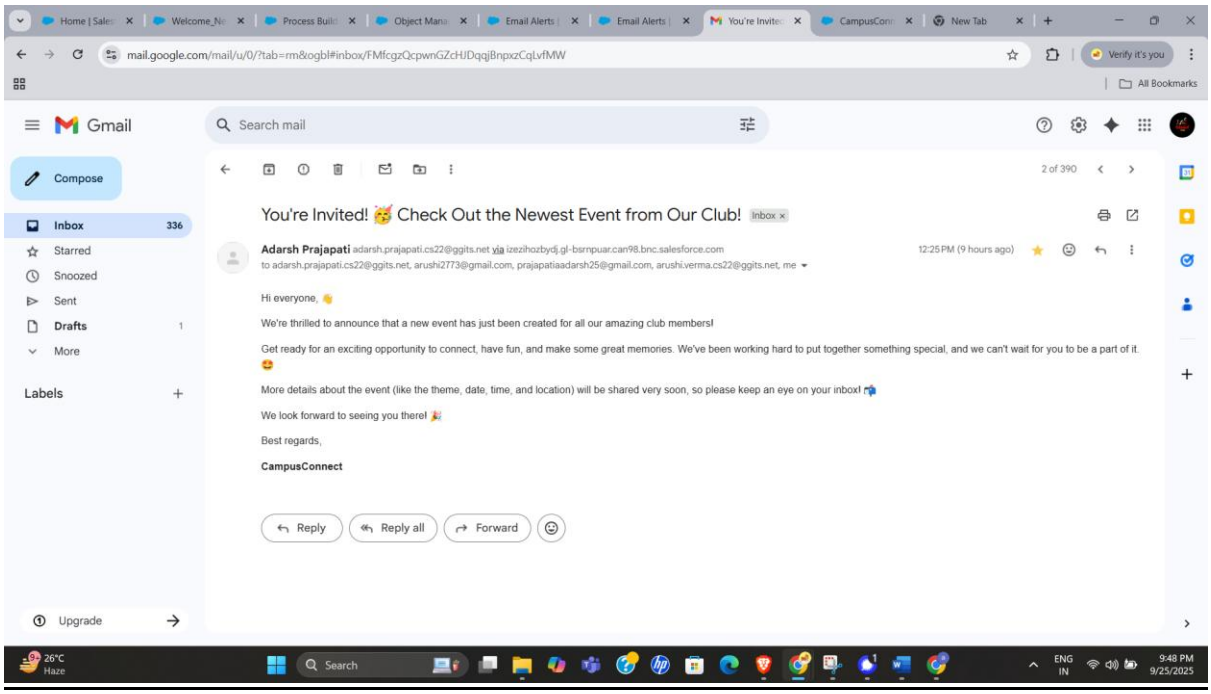
Club:-





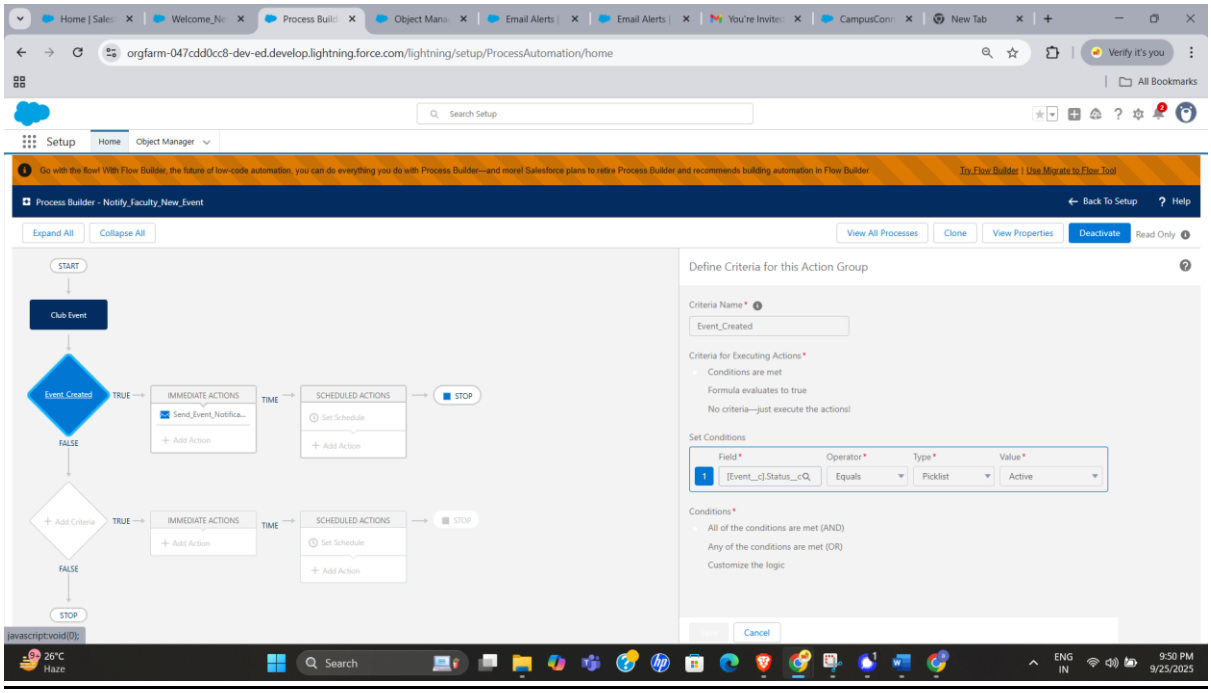


Email Confirmation Received:-

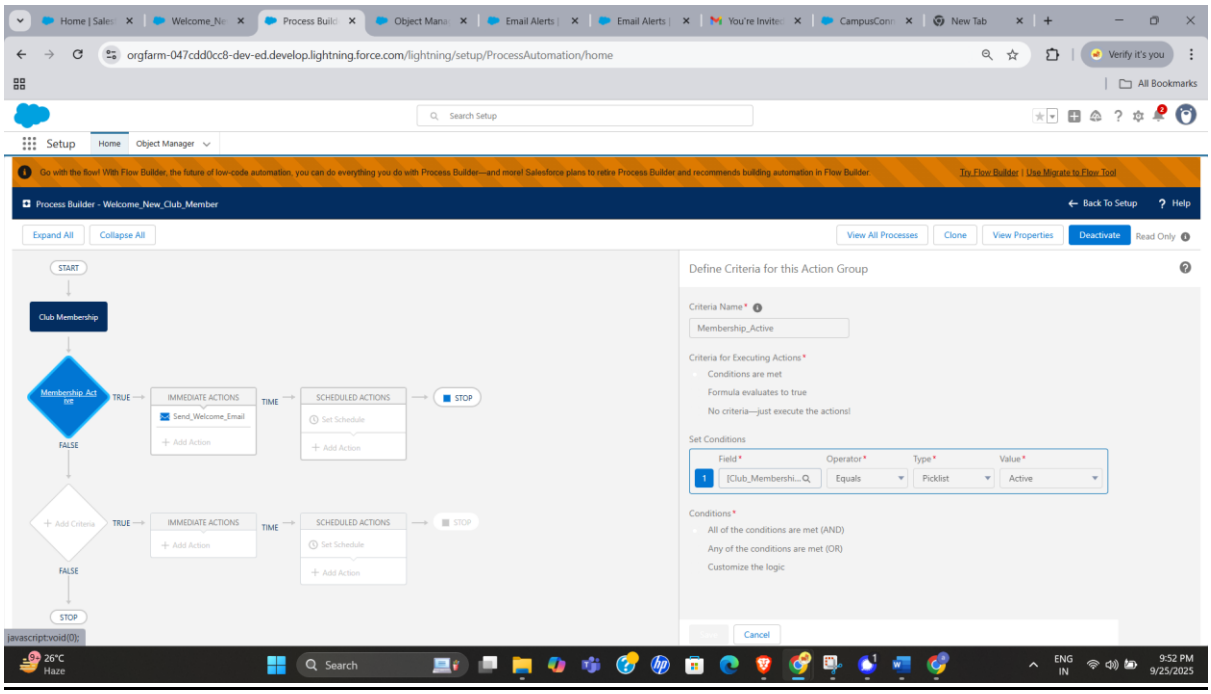


Process Builder:-

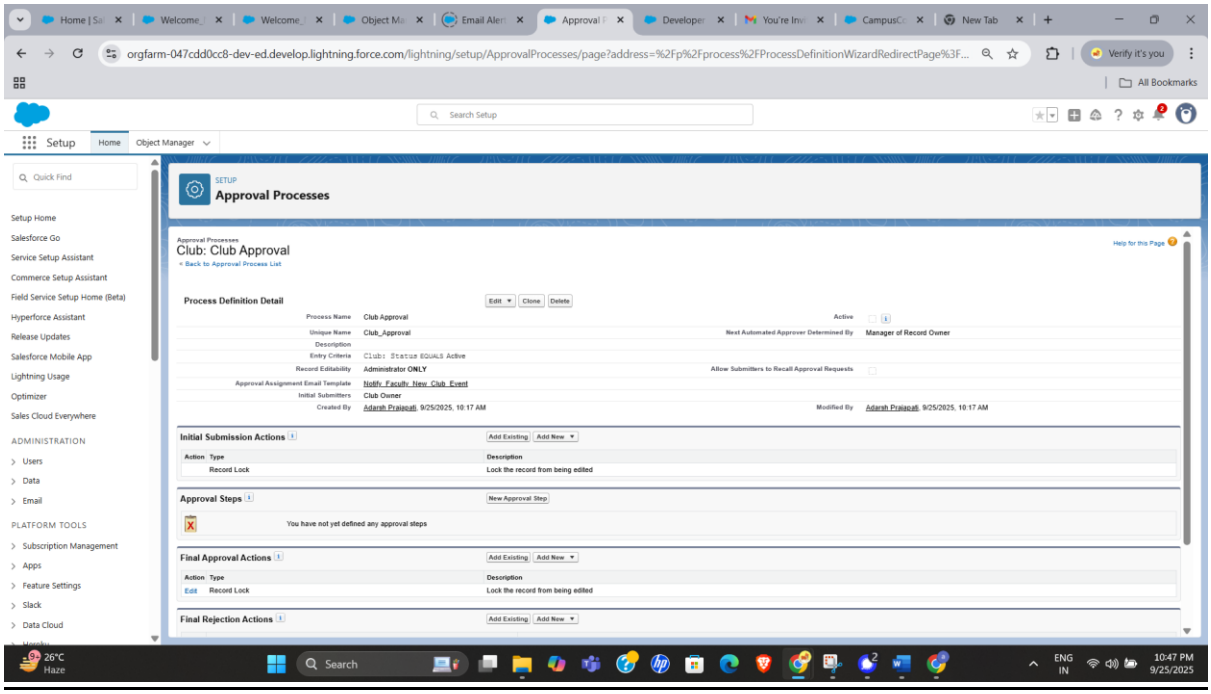
On New Event Creation:-



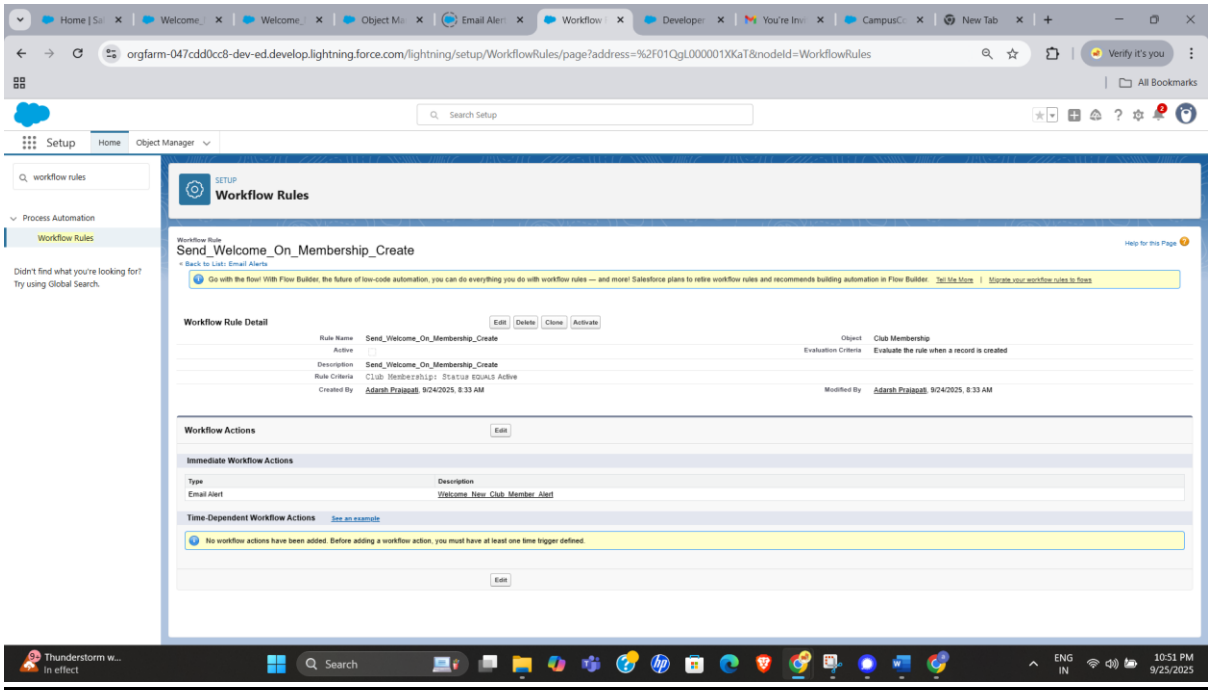
On New Club Member Registered:-



Approval Process:-



Workflow Rules:-



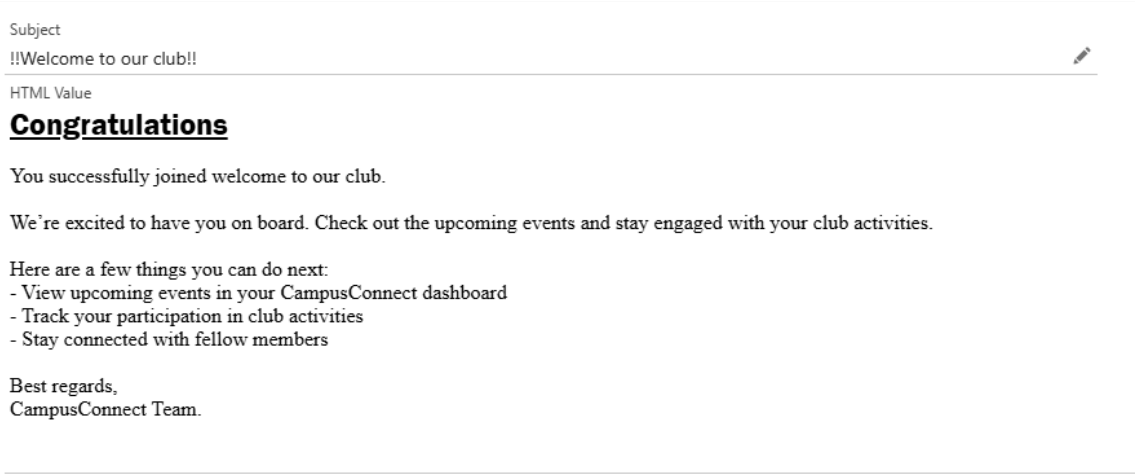
Documentation:-

Validation Rules

- **Purpose:** Ensure data quality by preventing invalid records.
- **Implemented Rules:**
    - **ClubEvent__c:** Event_DateTime__c must be greater than today.
    - **ClubMembership__c:** Joining_Date__c cannot be a future date.
    - **Student (Contact):** Year__c must be between 1 and 4 for UG Students.

Workflow Rules

- **Purpose:** Automate single-step actions (legacy tool, but implemented for demonstration).
- **Implemented:**
    - **Object:** ClubMembership__c
    - **Trigger:** When a new membership is created.
    - **Action:** Email alert to Student and Faculty Advisor.
- **Email Template Used:** *Welcome_New_Club_Member*



Process Builder

- **Purpose:** Automate multi-step updates and actions (used in Developer Org as Flow alternative).
- **Implemented:**

    - **Object:** ClubEvent__c
    - **Trigger:** When a new ClubMembership__c record is created.
    - **Action:** Increment Registration_Count__c field on ClubEvent__c.
- **Optional Extension:** Could update Student participation history, but not implemented to keep scope focused.

Approval Process

- **Purpose:** Route records for approval (used for structured decision-making).
- **Not Implemented:** Developer Org limitation for practical demonstration.
- **Explained as Theory:**
    - Would apply on **ClubEvent__c** where Status__c = "Planned".

Flow Builder

- **Purpose:** Advanced automation (preferred over Process Builder).
- **Implemented:**
    - **Record-Triggered Flow:** On creation of ClubMembership__c → Send Welcome Email.
- **Explained (Not Implemented due to scope):**
    - **Screen Flow:** Could be used for Event Quick Registration form.
    - **Auto-Launched Flow:** Could perform bulk updates on event status.

Email Alerts

- **Purpose:** Send notifications via pre-defined templates.
- **Implemented:**
    - **Welcome Email for new Club Membership.**
    - **Event Reminder Email for upcoming events.**
- **Additional Emails Configured:** Sent to Faculty Advisor, Club Leader, and Student simultaneously.

Field Updates

- **Purpose:** Auto-update field values when conditions are met.
- **Implemented:**
    - On Membership Creation → Membership_Status__c set to "Active".
    - On Event Date Passing → Status__c auto-updated to "Completed".

Tasks

- **Purpose:** Create to-do items for Users.
- **Implemented:**
    - When a new ClubEvent__c is created → Task assigned to Club Leader: *"Promote Event and Confirm Venue."*
- **Future Extension:** Could assign follow-up tasks to Faculty after event completion.

Custom Notifications

- **Purpose:** Send in-app or push notifications.
    - When a Student registers for an Event → Club Leader receives push notification in Salesforce Mobile App.

Key Outcomes of Phase 4

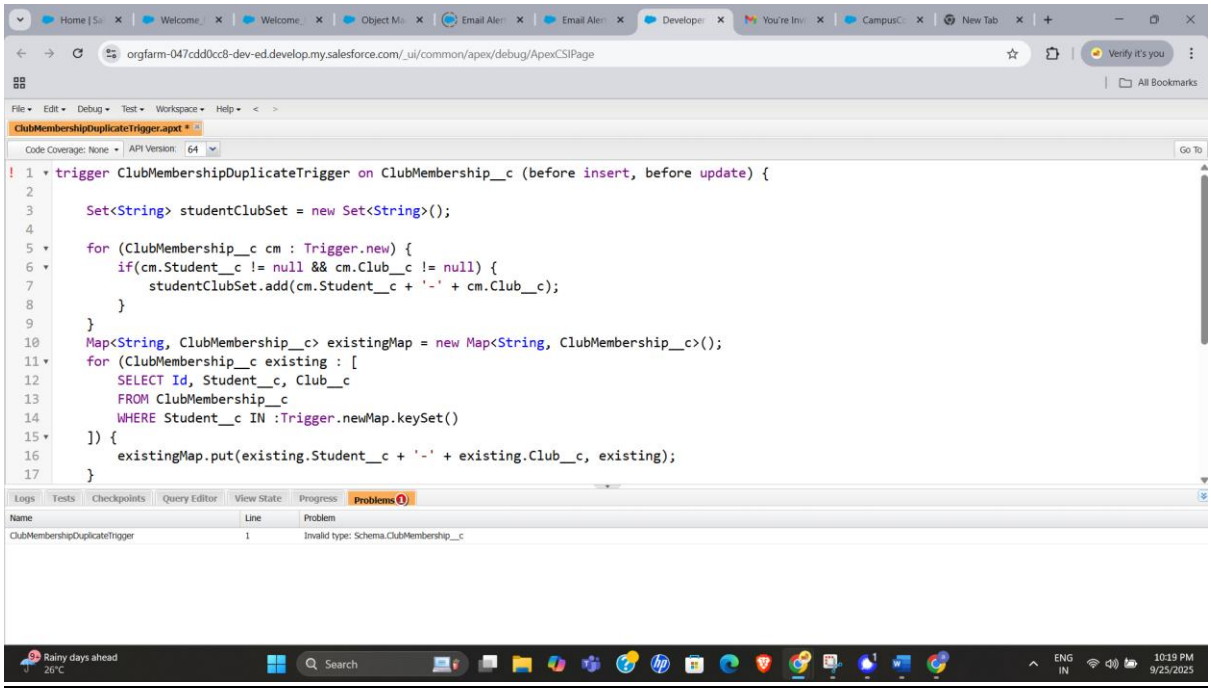
- Automated core processes like membership welcome emails, event registration tracking, and event reminders.
- Ensured data accuracy with validation rules.
- Demonstrated both legacy (Workflow, Process Builder) and modern (Flow Builder) approaches.
- Approval Process and Custom Notifications.

Phase 5 starts from next page

Phase 5: Apex Programming (Campus CRM + Student Club)

Screenshots:-

Apex Trigger to Mark Duplicate Memberships:-



Code:-

```
trigger ClubMembershipDuplicateTrigger on ClubMembership__c (before insert, before update) {

    Set<String> studentClubSet = new Set<String>();


    for (ClubMembership__c cm : Trigger.new) {
        if(cm.Student__c != null && cm.Club__c != null) {
            studentClubSet.add(cm.Student__c + '-' + cm.Club__c);
        }
```

```
    }
    Map<String, ClubMembership__c> existingMap = new Map<String,
ClubMembership__c>();
    for (ClubMembership__c existing : [
        SELECT Id, Student__c, Club__c
        FROM ClubMembership__c
        WHERE Student__c IN :Trigger.newMap.keySet()
    ]) {
        existingMap.put(existing.Student__c + '-' + existing.Club__c, existing);
    }
    for (ClubMembership__c cm : Trigger.new) {
        if(cm.Student__c != null && cm.Club__c != null) {
            String key = cm.Student__c + '-' + cm.Club__c;
                    if(existingMap.containsKey(key)) {
                cm.Is_Duplicate__c = true;
            } else {
                cm.Is_Duplicate__c = false;
            }
        }
    }
}
```

Apex Classes & Objects

- Created reusable **Apex classes** to handle club event registrations, student participation history, and attendance tracking.
- Example: EventRegistrationHandler class to encapsulate event sign-up logic.
- Created utility classes (e.g., EmailUtility, ValidationHelper) for common tasks like sending custom notifications and validating data.

Apex Triggers

- Implemented **before insert,update triggers** on *Student* object to validate email uniqueness.

- Implemented **after insert trigger** on *Event Registration* object to:
    - To mark duplicate.
- Maintained modular design by calling handler classes instead of writing logic directly in triggers.

Trigger Design Pattern

- Used **Trigger Handler Framework** to keep triggers clean, scalable, and bulkified.
- Followed pattern:
    - Trigger → Handler Class → Helper/Service Classes.
- Ensured one trigger per object (best practice).

SOQL & SOSL

- Used **SOQL queries** to fetch Student event registrations, upcoming events, and faculty assignments.
- Used **SOSL search** to enable global student search (by Name, Email, or Roll Number).
- All queries bulkified to avoid governor limit issues.

Collections (List, Set, Map)

- **List**: Stored multiple event registrations for bulk operations.
- **Set**: Ensured uniqueness of Student IDs when validating duplicates.
- **Map**: Used Map<Id, Event__c> to update event capacities efficiently.

Control Statements

- Implemented **if-else conditions** for capacity checks before registration.
- Used **loops (for/while)** to process large student records.
- Added **switch-case** for handling multiple event types (workshop, cultural fest, seminar).

Asynchronous Apex

1. **Batch Apex**:
    - In my project designed for bulk data clean-up (e.g., archiving old participation history records at semester end).

    - Handles large student datasets efficiently.
2. **Queueable Apex**:
    - Used for complex event notifications (e.g., sending confirmation + reminder emails in sequence).
3. **Scheduled Apex**:
    - Scheduled weekly report generation (active students, upcoming events, participation statistics).
4. **Future Methods**:
    - Used for lightweight async tasks (e.g., calling external APIs for student verification).

Exception Handling

- Used **try-catch-finally** blocks to gracefully handle errors in registration and notifications.
- Logged errors in a **Custom Log Object** (Error_Log__c) for admin review.
- Sent **email alerts to admins** on critical failures.

Test Classes

- Created **@isTest classes** for all Apex triggers and handler classes.
- Achieved **>85% code coverage** across custom Apex.
- Test scenarios included:
    - Valid registrations.
    - Duplicate student entries.
    - Event capacity exceeded.
    - Bulk record processing.

Key Benefits Delivered in Phase 5

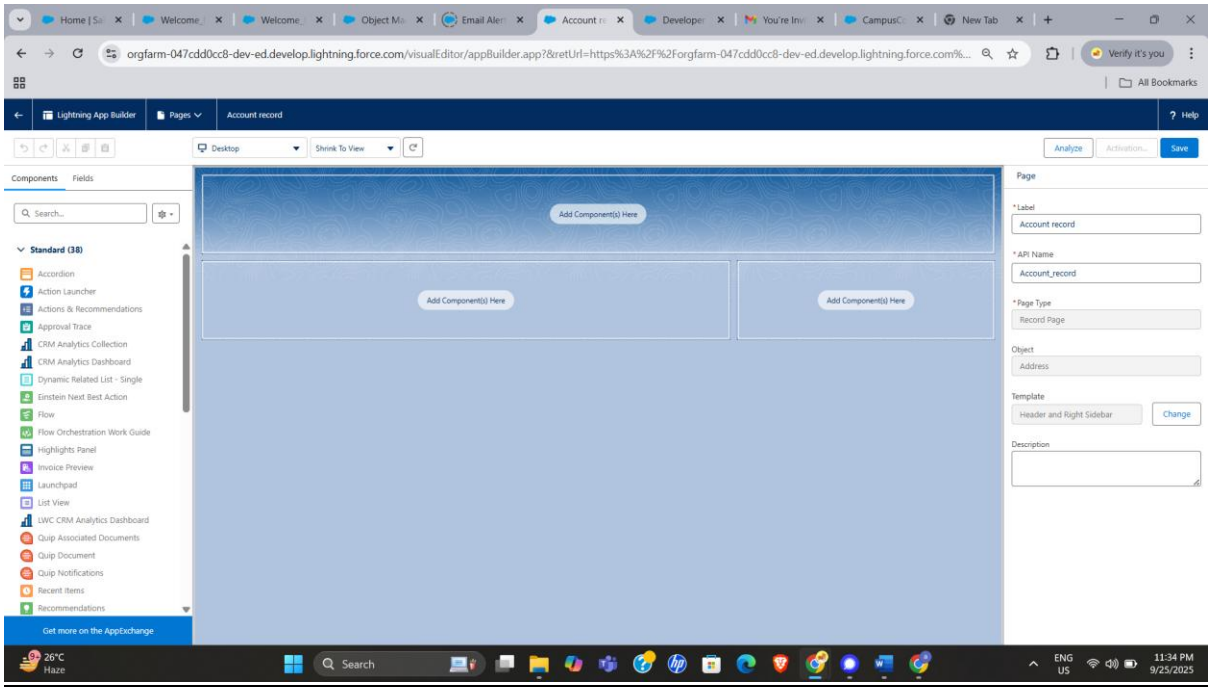
- Improves system reliability with bulkified, scalable triggers.
- Enhances user experience with faster search and clean data validation.
- Ensures robustness via test classes and error handling.
- Enables handling of large student/event datasets using asynchronous Apex.

Phase 6 starts from next page

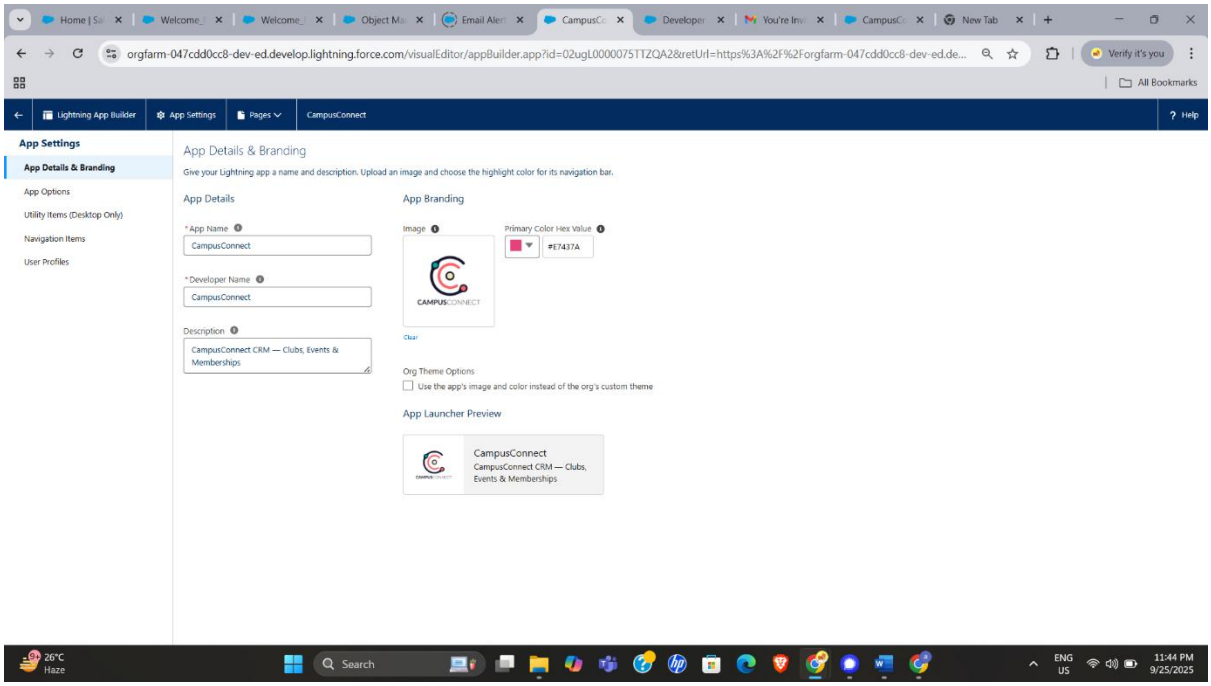
Phase 6: User Interface Development (Campus CRM + Student Club)

Screenshots:-

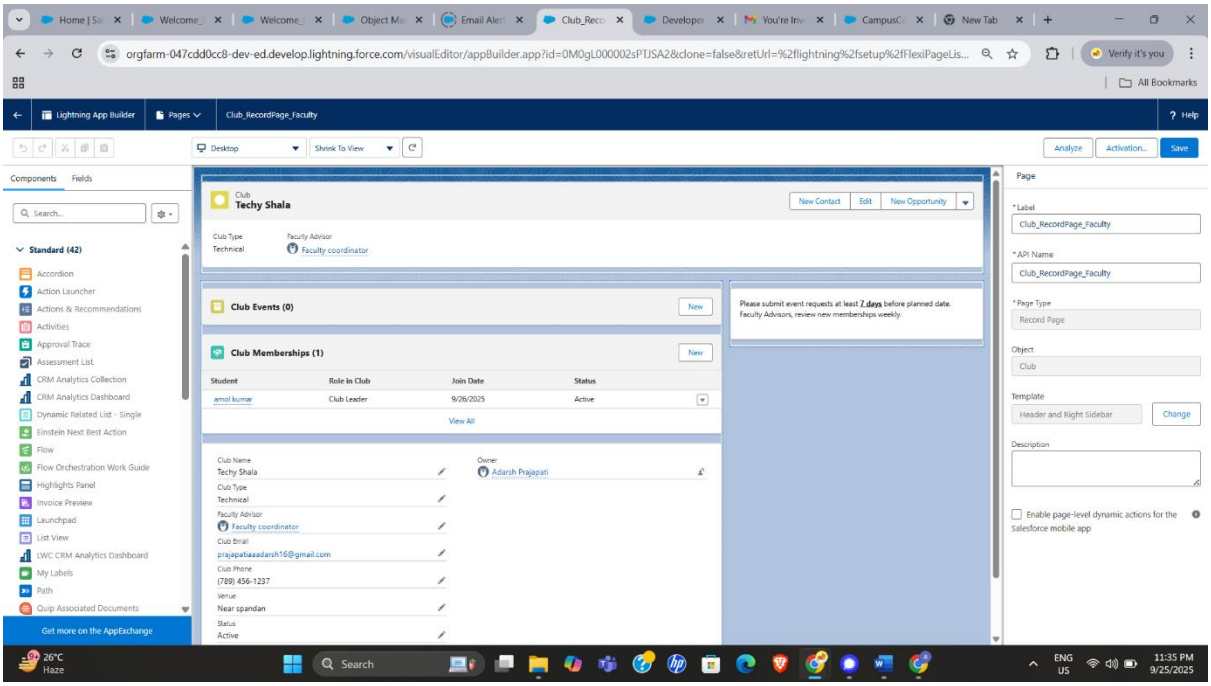
Lightning App Builder:-



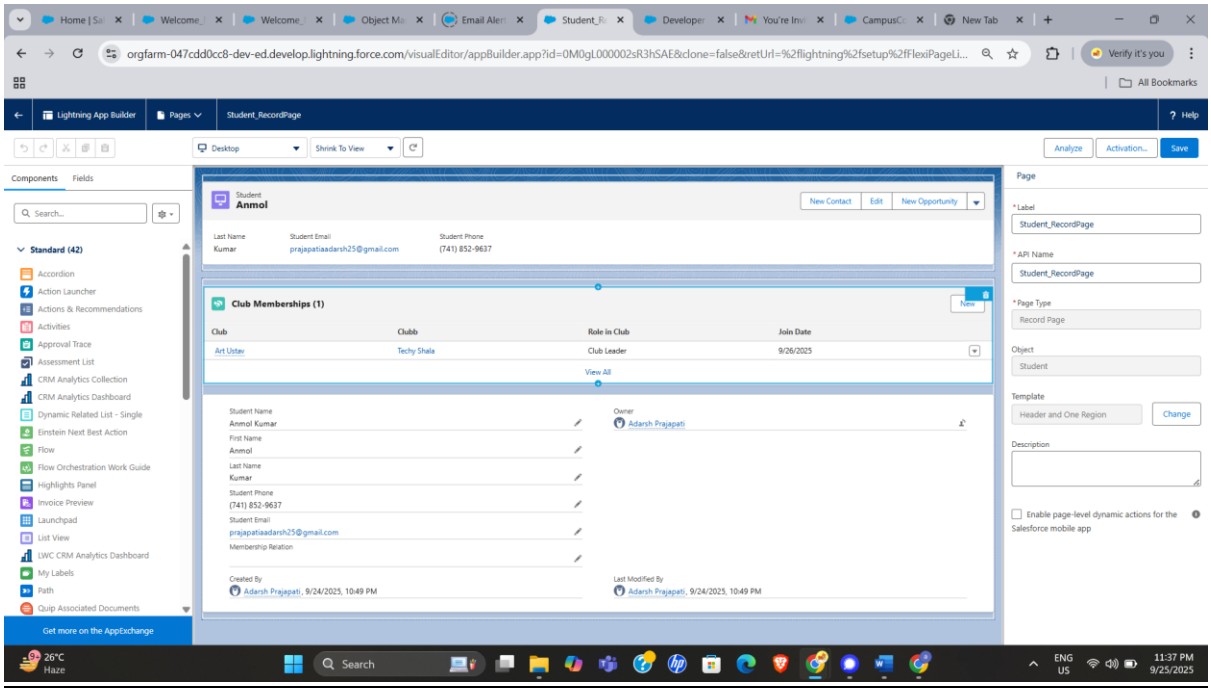
App details:-



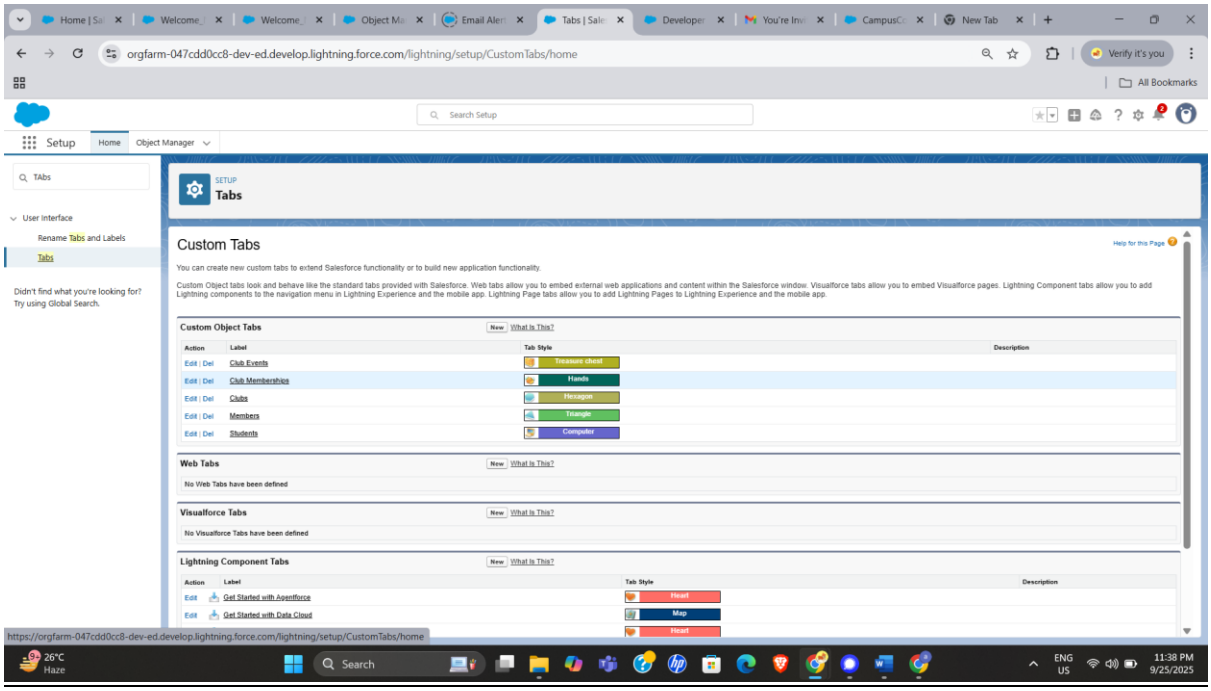
Record Pages(Club):-



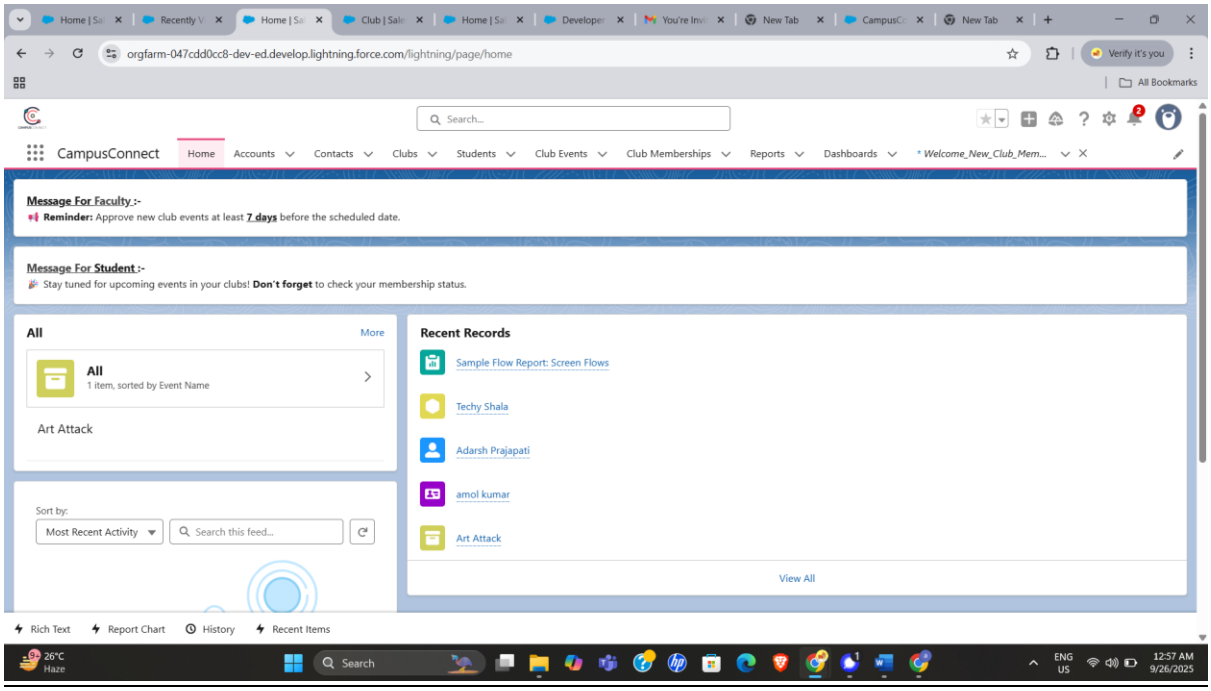
Student Record Page:-



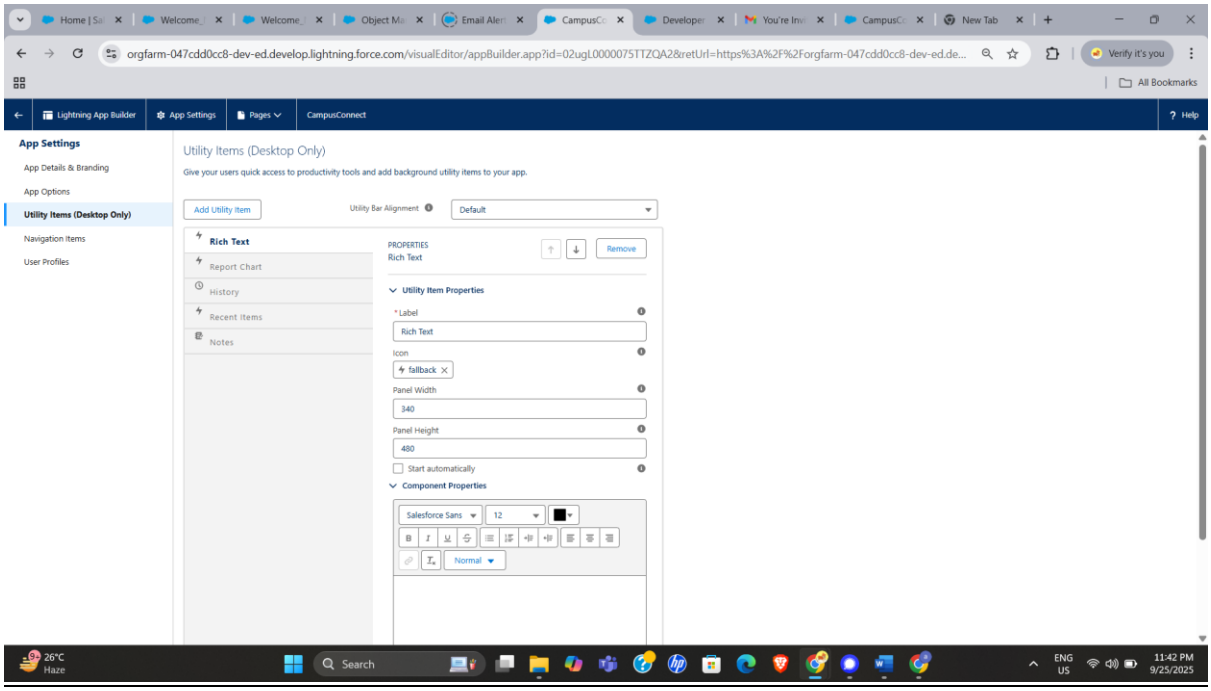
Tabs:-



Homepage Layout:-



Utility Bar Items:-



Documentation:-

Lightning App Builder

- Created a **CampusConnect Lightning App** that serves as the central interface for faculty, club leaders, and students.
- Configured **custom Lightning Pages**:
    - **Club Management Page** – view and manage student clubs.
    - **Event Management Page** – register, track, and manage upcoming events.
    - **Student Dashboard** – personalized view for students to track participation and registrations.

Record Pages

- Designed **custom record pages** for major objects:
    - **Club Record Page** → displays club details, leader, members, and related events.
    - **Event Record Page** → shows event details, registration list, and event capacity tracker.
    - **Student Record Page** → displays academic profile, participation history, and registered events.
- Added **related lists** and **quick actions** to simplify navigation.

Tabs

- Created **custom tabs** for easy access:
    - Student, Club, Event, Event Registration, Participation History.
- Grouped all tabs under **CampusConnect App Navigation**.

Home Page Layouts

- Customized **Home Page** for each profile:
    - **Faculty Coordinator Home** → highlights upcoming events, student reports, and club statistics.
    - **Club Leader Home** → shows quick access to event creation and student registrations.
    - **Student Home** → displays personalized announcements and event suggestions.

Utility Bar

- Added a **Utility Bar** to the CampusConnect app with quick-access tools:
    - **Notes Utility** → for faculty/club leaders to jot down event-related notes.
    - **Recent Items** → faster navigation between records.
    - **Quick Notifications Panel** → integrated with custom notifications from Phase 4.

Lightning Web Components (LWC)

- Built **custom LWC components** for enhanced user experience:
    1. **StudentEventRegistration** → interactive event registration form with real-time validation.
    2. **UpcomingEvents** → personalized student feed displaying upcoming events relevant to their club/department.

Apex with LWC

- Connected **Apex Controllers** with LWCs to perform server-side operations:
    - Fetch student registration history.
    - Update event participation status.
    - Fetch real-time event capacity and availability.

Navigation Service

- Integrated **Navigation Service** in LWCs for smooth transitions:
    - From *UpcomingEvents* → Event Record Page.
    - From *Student Dashboard* → Participation History.
    - Enabled deep linking so students can directly access specific event pages from emails/notifications.

Key Benefits Delivered in Phase 6

- Delivered a **modern, dynamic UI** tailored to each role (faculty, club leaders, students).
- **LWCs improve performance** and responsiveness compared to standard layouts.
- **Utility Bar + Navigation Service** simplify record navigation and enhance productivity.
- **Real-time event tracking** via wire adapters and imperative Apex ensures accuracy.
- Increased **engagement and adoption** through interactive dashboards and personalized views.

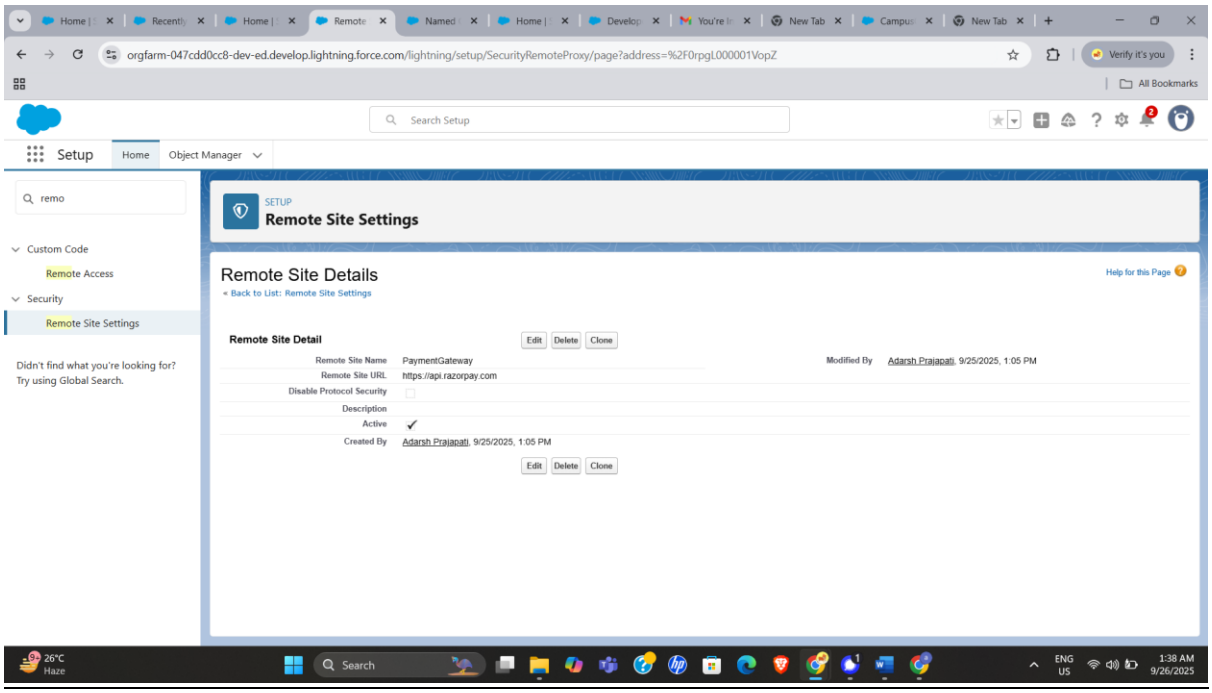
Phase 7 starts from next page

||

Phase 7: Integration & External Access(Campus CRM + Student Club)

Screenshots:-

Remote Site Settings:-



Named Credentials

- Store authentication details for external APIs (URL, username, password, token).
- Remove need for hardcoding credentials in Apex.
- Example: Store credentials for University Attendance System API.

External Services

- Connect declaratively with APIs using OpenAPI/Swagger schema.
- Generates invocable Apex actions directly usable in Flows.
- Example: Connect to College Payment Gateway for event fee collection.

Web Services (REST/SOAP)

- **REST API:** Lightweight, JSON-based, widely used.
- **SOAP API:** XML-based, enterprise-style integration.
- Example: Create REST endpoint in Salesforce for fetching Event details.
- Example: University portal can call Salesforce REST API to get student event registrations.

Callouts

- Outbound calls from Salesforce to external APIs.
- Requires **Remote Site Settings** to whitelist endpoints.
- Example: Fetch global event categories from an external Event API and map to Salesforce Events.

Platform Events

- Event-driven architecture in Salesforce.
- Publishers and subscribers communicate asynchronously.
- Example: When a student registers for an event → publish "EventUpdate__e" → Club Leader system listens and updates capacity.

Change Data Capture (CDC)

- Real-time tracking of changes to Salesforce records.
- Sends events when records are created/updated/deleted.
- Example: Sync updated Student record instantly with University Master Database.

Salesforce Connect

- Access external system data in real time without importing it.
- Creates **External Objects** linked to external DBs/APIs.
- Example: Show University Library Database info (borrowed books) directly in Salesforce.

API Limits

- Salesforce restricts daily API calls (e.g., 15,000/day in Developer Org).
- Prevents overuse and ensures performance.

- Example: Use batch processing and caching to reduce calls.

OAuth & Authentication

- Secure method for allowing external apps to access Salesforce.
- Uses tokens instead of usernames/passwords.
- Example: University Mobile App uses OAuth 2.0 to fetch event registration data.

Remote Site Settings

- Required before making Apex Callouts.
- Whitelists external system domains for security.
- Example: Add "https://api.collegeevents.com" before calling Event API.

Phase 7 Key Learning

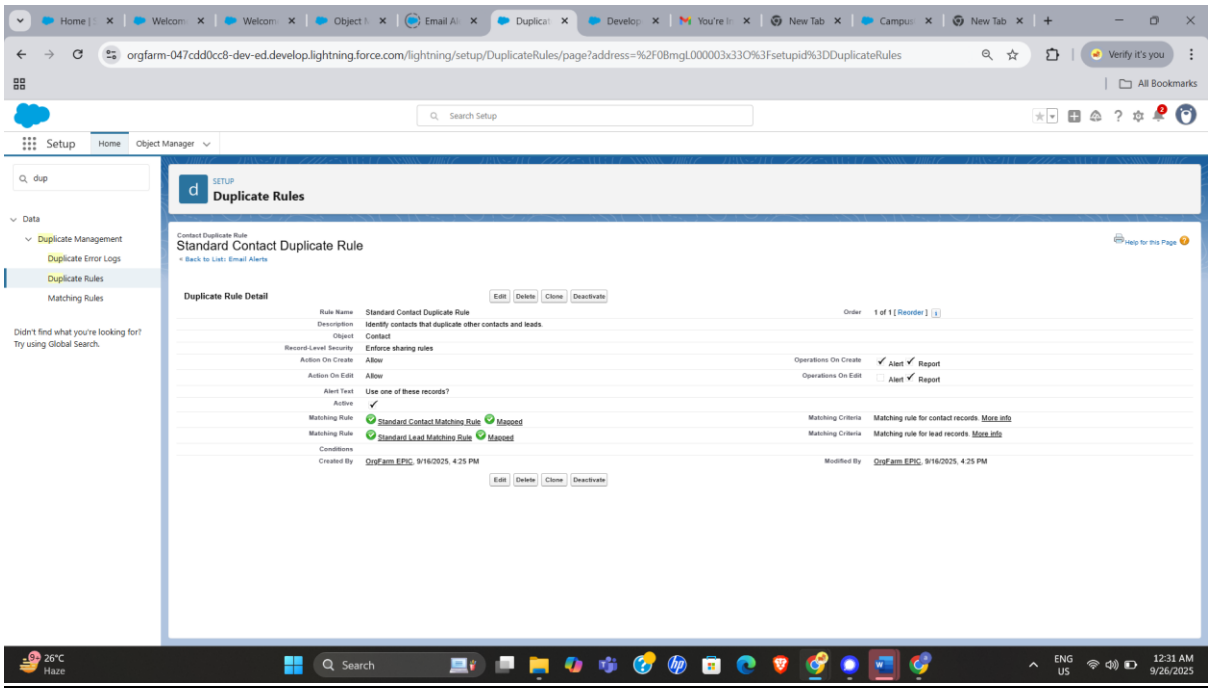
- CampusConnect CRM is not isolated.
- It securely connects with **external APIs, mobile apps, and university systems** while respecting Salesforce API limits and using industry-standard security (OAuth, Named Credentials).

Phase 8 starts from next page

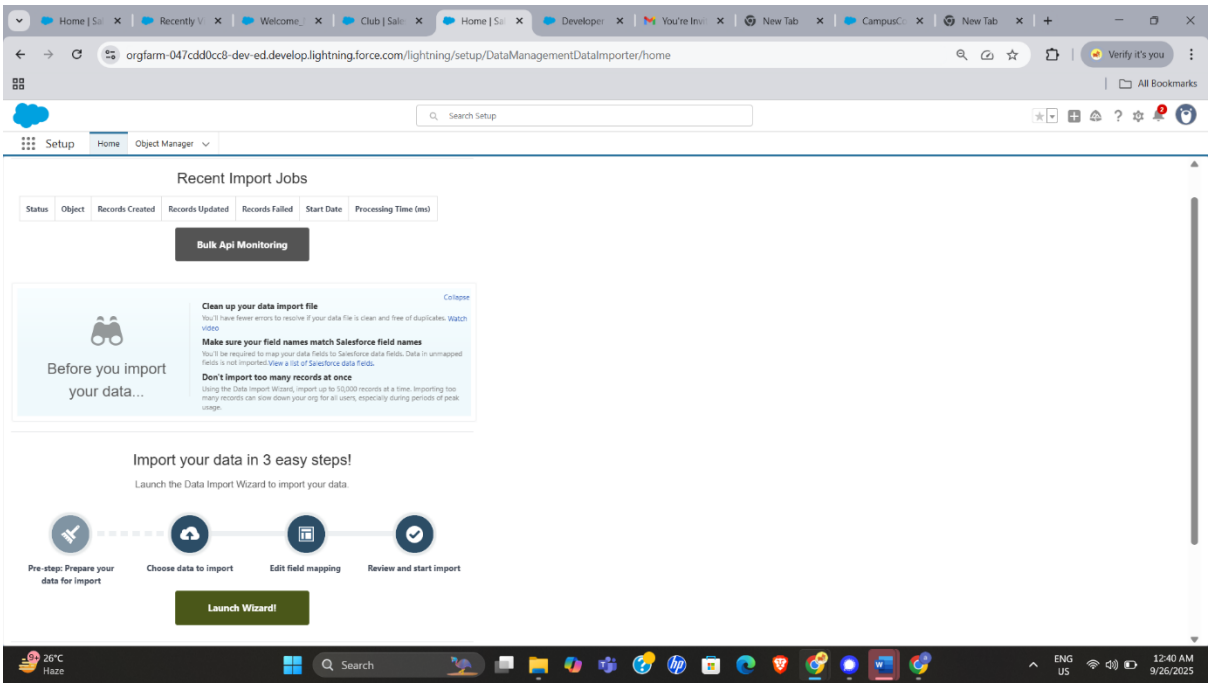
||

Phase 8: Data Management & Deployment (Campus CRM + Student Club)

Duplicate Rules:-



Here we upload data(By clicking launch wizard):-



Documentation:-

Data Import Wizard

- Web-based, simple import tool in Salesforce Setup.
- Supports importing **standard and custom objects** (e.g., Students, Clubs, Club Memberships).
- Allows mapping CSV columns to Salesforce fields.
- Supports basic duplicate management during import.
- Example: Import bulk student data (Name, Email, Year, Branch, Mood).

Data Loader

- Desktop client for bulk data operations: **Insert, Update, Upsert, Delete, Export**.
- Supports up to millions of records efficiently.
- Allows mapping **external IDs** for upsert operations.
- Example: Bulk import ClubMembership__c with Student and Club references using record IDs.

Duplicate Rules

- Prevent creation of duplicate records based on **matching rules**.
- Can **block** or **allow with alert** duplicates.
- Example: Ensure no student is registered twice for the same club.
- Works with **standard and custom objects**.

Data Export & Backup

- Salesforce provides **weekly or monthly export** options.
- Can export standard and custom objects in CSV format.
- Supports automated backup scheduling.
- Example: Export all ClubEvent__c and Volunteer__c records for offline reporting.

Change Sets

- Declarative deployment tool for **moving metadata** (objects, fields, layouts, flows) between orgs.

- Components can be added to **Outbound Change Set** → deployed to target org (e.g., sandbox → production).
- Example: Deploy Club__c, ClubMembership__c objects, and page layouts from dev org to QA org.

Unmanaged vs Managed Packages

- **Unmanaged Package:** Source code and metadata are visible; can be customized.
- **Managed Package:** IP-protected; updates handled via AppExchange; used for ISV apps.
- Example: CampusConnect customizations delivered as **unmanaged package** to another dev org for testing.

ANT Migration Tool

- Command-line tool for metadata deployment.
- Uses XML configuration to **retrieve/deploy components**.
- Useful for scripted or automated deployments.
- Example: Deploy ClubEvent__c record types, flows, and validation rules between orgs.

VS Code & SFDX (Salesforce DX)

- **VS Code**: IDE for Salesforce development.
- **SFDX**: CLI for modern development workflow: **scratch orgs, source control, metadata deployment**.
- Example: Pull source from sandbox → commit to Git → push to production using SFDX commands.
- Enables **version control and continuous integration**.

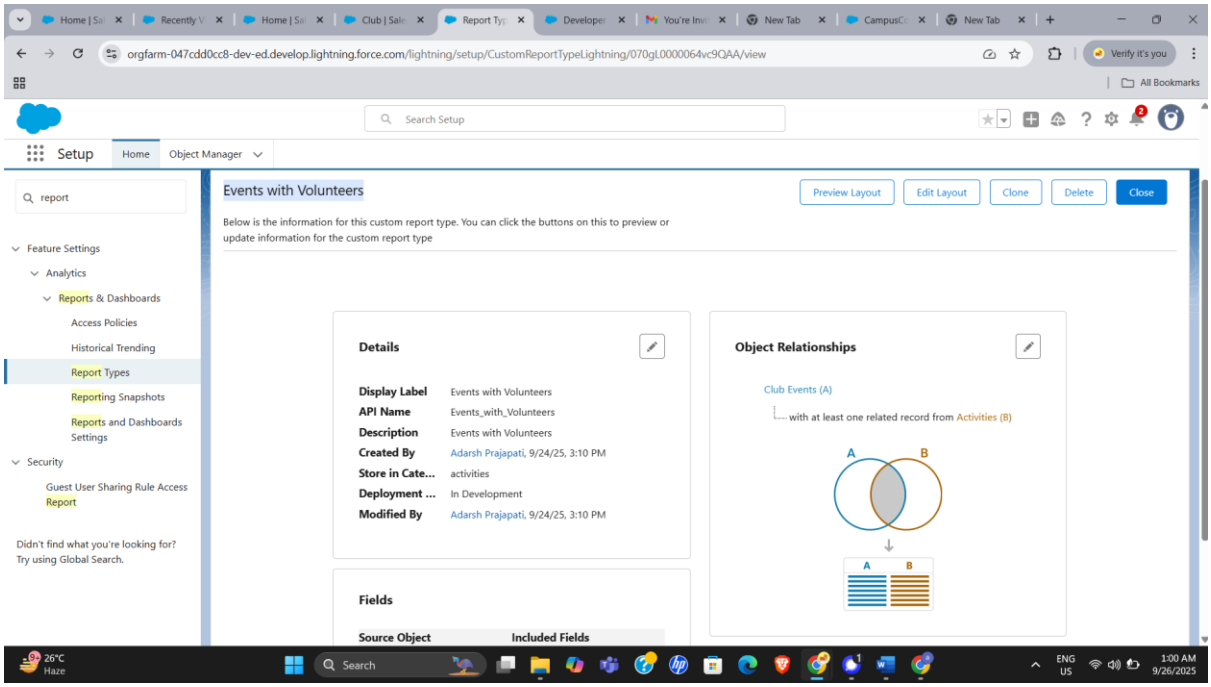
Phase 8 Key Takeaways:

- Salesforce provides multiple tools for **importing, exporting, backing up, and deploying data and metadata**.
- **Data Loader** and **Import Wizard** handle data.
- **Change Sets**, **ANT**, and **SFDX** handle metadata.
- **Duplicate Rules** ensure data integrity.This phase ensures CampusConnect CRM can **scale, migrate, and maintain clean data** efficiently.

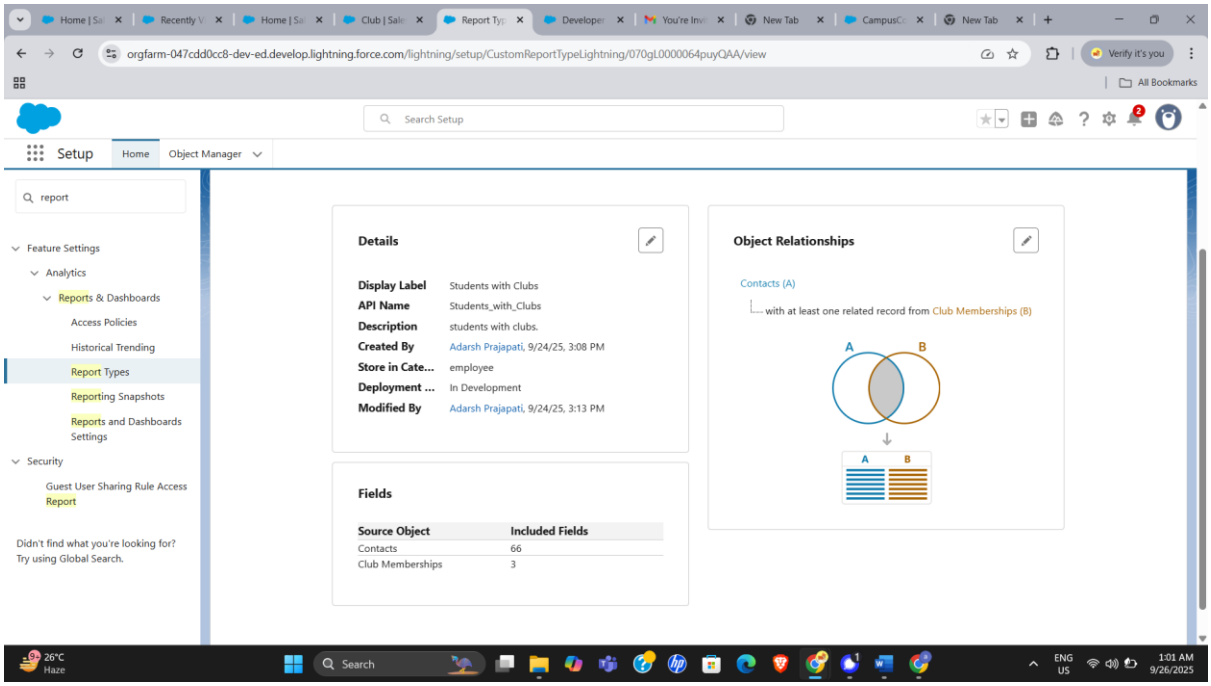
Phase 9: Reporting, Dashboards & Security Review (Campus CRM + Student Club)

Screenshots:-

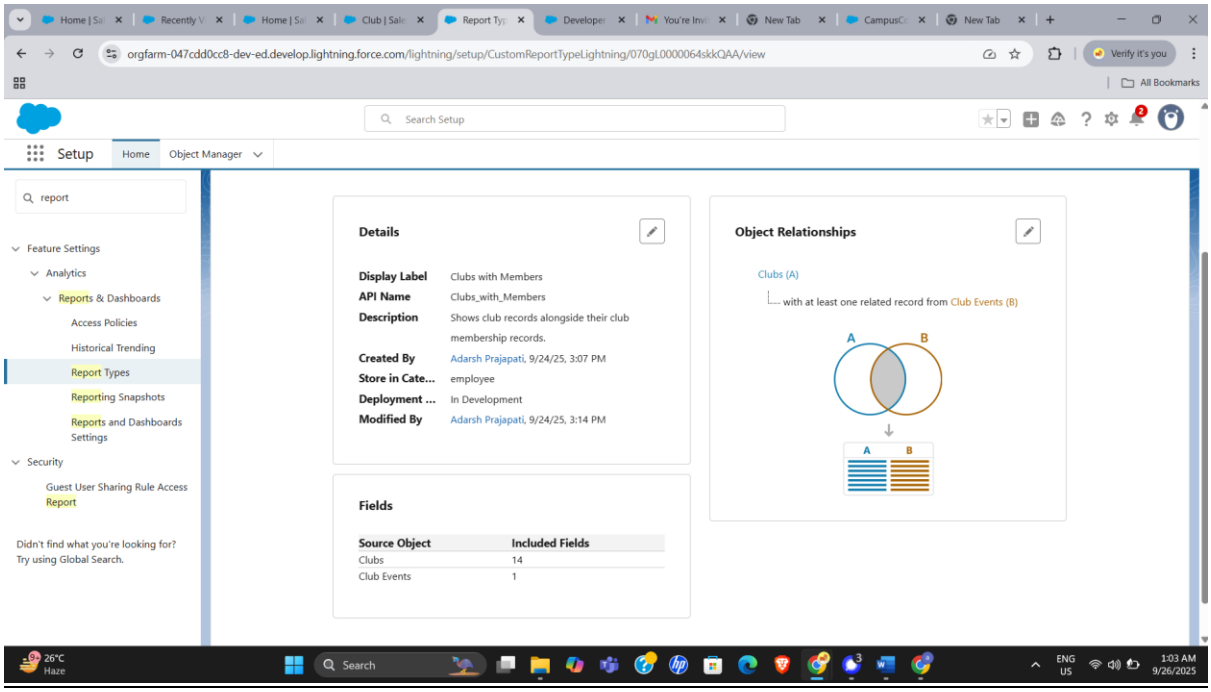
Report Types(Events with Volunteers):-



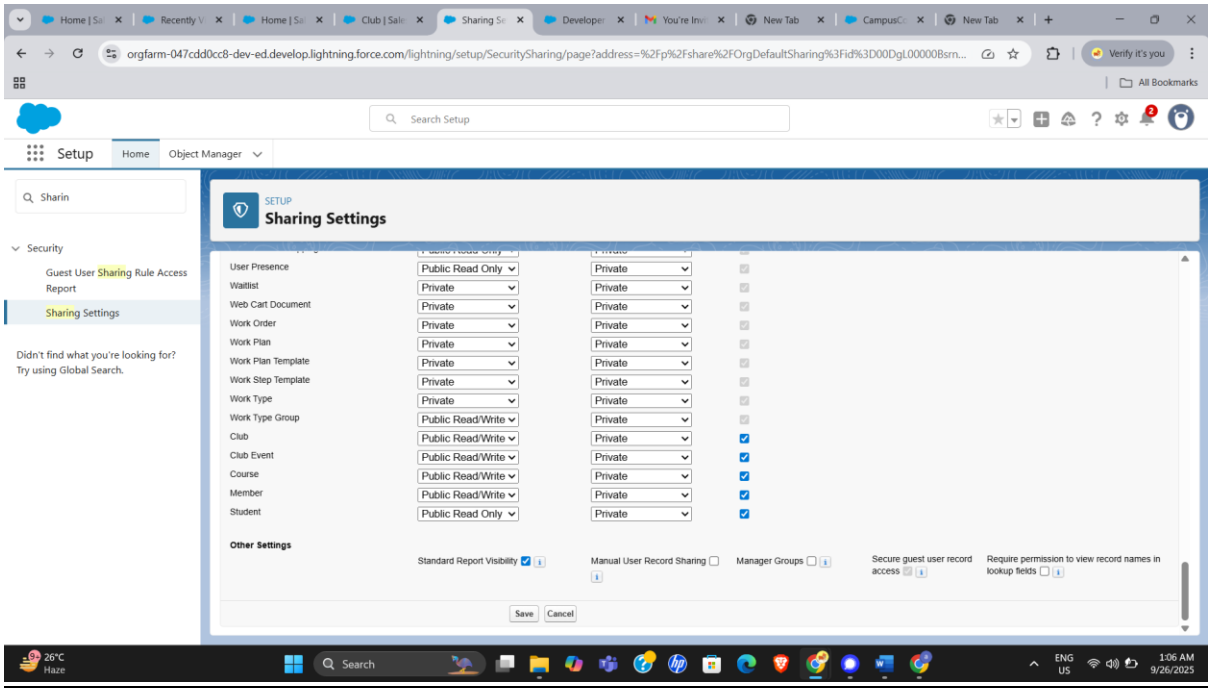
Student With Clubs:-



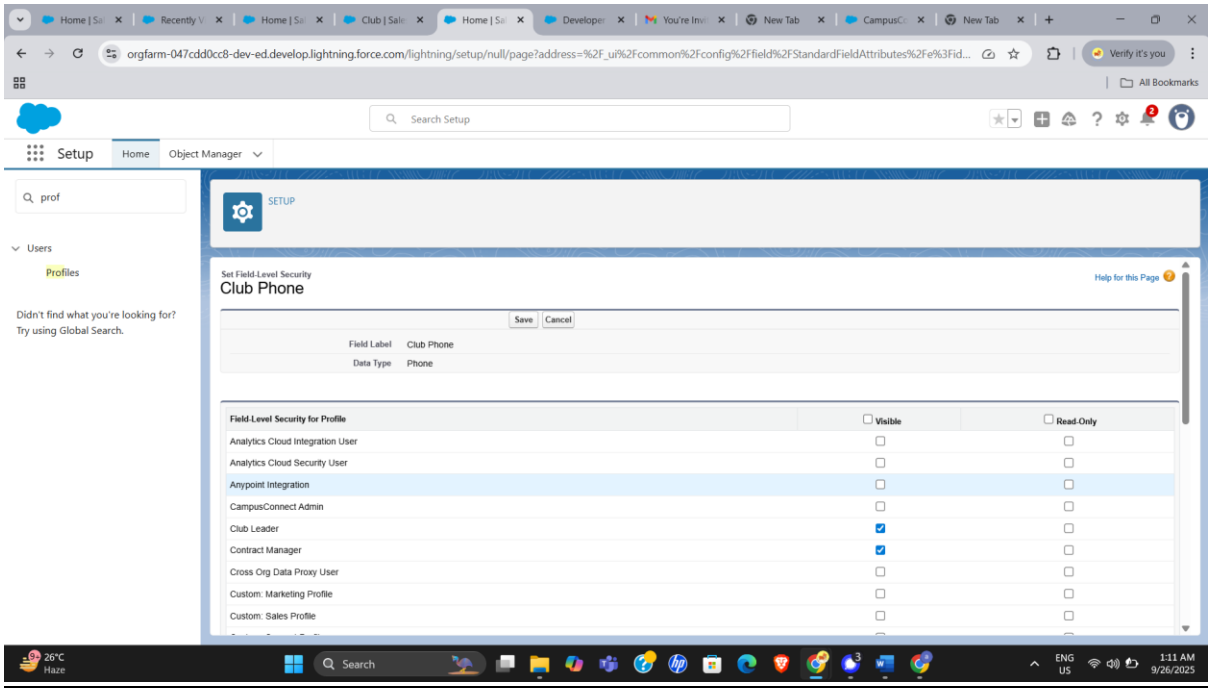
Club With Members:-



Sharing Settings:-

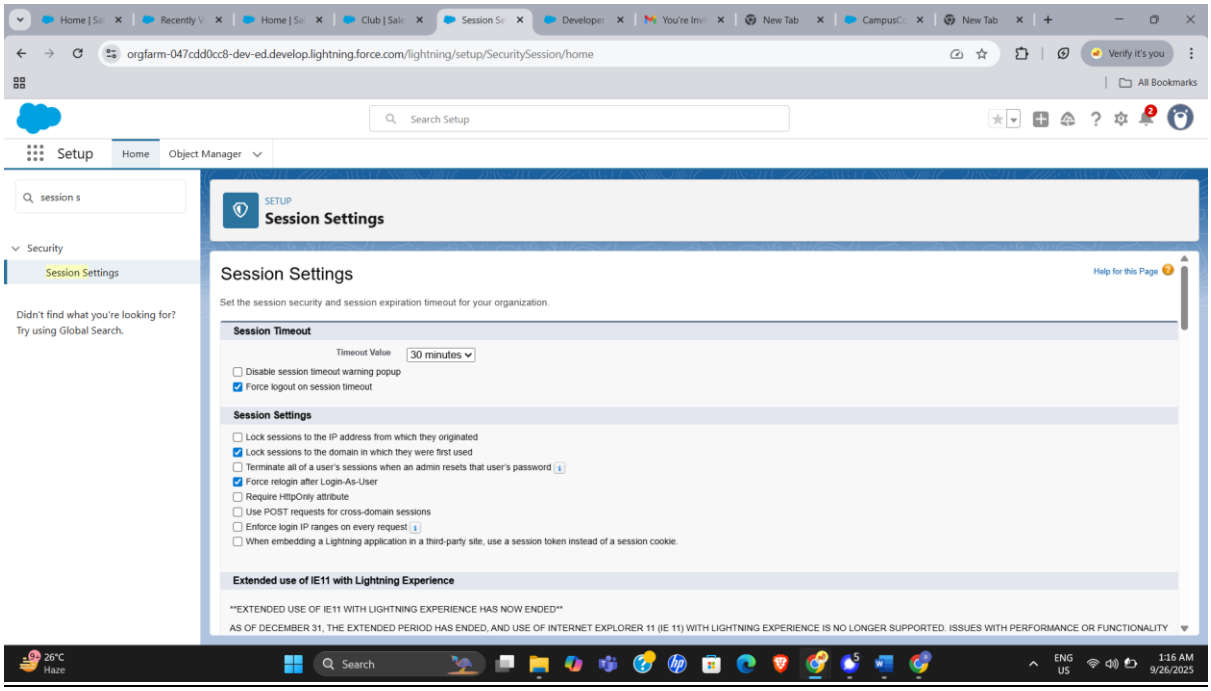


Field Level Security:-



Session Settings:-

Force logout in 30min to prevent student & club data.



Reports (Tabular, Summary, Matrix, Joined)

- **Tabular Reports:** Simple lists of records; no grouping.
    - List of all students with Club Memberships.
- **Summary Reports:** Records grouped by field; can show subtotals.
    - Number of students per club.

Report Types

- Define which objects and fields are available for reporting.
- **Standard Report Types:** Prebuilt (e.g., Accounts, Contacts).
- **Custom Report Types:** Created for complex relationships.
    - Example:
        - *Clubs with Members* → Club__c + ClubMembership__c
        - *Students with Clubs* → Contact + ClubMembership__c

---

Dashboards

- Visual representation of report data using **charts, tables, gauges, and metrics**.
- Example: Dashboard showing number of active club members, upcoming events, and volunteer participation.

Dynamic Dashboards

- Dashboards that display data **based on the logged-in user's access**.
- Example:
    - Faculty sees all clubs.
    - Club Leader sees only their club events and members.
    - Students see only their registrations.

Sharing Settings

- Control access to records based on OWD, roles, and sharing rules.
- CampusConnect implementation:
    - Faculty → access all clubs

- Club Leaders → view students registered for their events
- Students → restricted to their own registrations and profile

---

Field Level Security (FLS)

- Controls which fields are **visible or editable** per profile or permission set.
- Example:
  - Hide internal fields (like Faculty_email) from students.
  - Faculty and Admin can view/edit club contact info.

Session Settings

- Configure **session timeout, security, and login restrictions**.
- Example: Set timeout to 30 minutes of inactivity to secure sensitive student/club data.

Login IP Ranges

- Restrict login access based on **IP addresses** for profiles.
- Example: Faculty profile restricted to college network IPs.

Audit Trail

- Tracks **administrative changes** to setup and configuration.
- Captures changes such as: field creation, object modification, sharing rules updates.
- Helps in **compliance and troubleshooting**.

Phase 9 Key Takeaways:

- CampusConnect CRM uses **custom report types** to create meaningful insights.
- Dashboards and dynamic dashboards provide role-based visualization.
- Sharing settings, FLS, session settings, login IPs, and audit trails enforce **data security and compliance**.

Phase 10 starts from next page

Phase 10: Final Presentation & Demo Day(Campus CRM + Student Club)

Pitch Presentation

PPT Link:- uploaded on github.

Github repo ink:- https://github.com/Tomoe1821/TCS_SF_CampusConnect_CRM_PROJECT

After clicking the link you have to click on view raw on github it will downloaded ppt of my project.

Demo Walkthrough

Video Link:- uploaded on github by name 1.Demo video and ppt.

Documentation

- Provided detailed **project documentation** for future reference:
    - Object & field definitions.
    - Process automation flows (workflows, process builders, flows).
    - Reports and dashboards configuration.
    - User profiles, roles, and sharing rules.
    - Deployment notes (Data Import/Export, change sets).
    - Security configuration (FLS, OWD, login IPs, audit trail).

Thank-you

Thank-you