CAMPUSCLUB360

A Salesforce-based Application for Automating Student Activities and Campus Mart Management

Project Submitted By: Rakesh Yarra
College Name: Pragati Engineering College
Department: Electronics and Communication Engineering
Academic Year: 2025
Under the Guidence of: Mir Abdul Rehman

Salesforce Program 2025 – TCS Last Mile in association with SmartBridge

Pragati Engineering College

SMARTBRIDGE
Let's Bridge the Gap

salesforce
PARTNER

CampusClub360

Phase 1: Problem Understanding & Industry Analysis:

Problem Statement – CampusClub360

Educational institutions face inefficiencies in managing student clubs, sports, attendance, and campus store operations due to fragmented manual processes, resulting in errors, delays, and limited visibility for management. CampusClub360 addresses these challenges by offering a unified Salesforce-based platform that enables seamless registrations with fee tracking, QR code–based attendance, digital sports management, barcode-enabled campus mart operations for automated billing and inventory tracking, and centralized dashboards --reducing errors, saving time, and empowering management with real-time insights.

1. Requirement Gathering:

Student Club Management:

- Interested students approach faculty, submit their details, and pay the Club Pass fee of ₹2000.
- Faculty creates a student record in the Students tab
- Upon record creation, a QR Code Pass is generated for the student.
- Club Meetings are created by Faculty or Club Leaders.
- Students can view available club meetings.
- Attendance is captured via Club Attendance Scanner:
  - Faculty scans students' QR codes during meetings.
  - Attendance records are automatically stored in the **Attendance tab**.

Sports Registration & Management:

- Students submit their sports registration details at the **Campus Mart**.
- Salesperson or Store Manager fills out the **Sports Registration form.**
- After fee payment, clicking **Register** creates a record in the **Sports Registration Teams tab**.
- PT Head/PT Assistant assign team player names and enter match scores after matches start.
- Sports registration ensures proper team management, payment tracking, and match result recording.

Campus Mart (Store) Management:

- The **Campus Mart page** has three layers:
  - **Top Layer – Sports Registration:** Managed by salesperson and store manager.
  - **Left Layer – Item Scanner:** Salesperson scans student items.
  - **Right Layer – Cart:** Items added to the cart are displayed here.
- After scanning, clicking **Done** stores transactions in **Store Sales**.
- **Store Items** contains all available items; stock reduces automatically on sale.

2. Stakeholder Analysis:

Stakeholder	Responsibilities & Access
Dean	Full access (like System Admin). Oversees all modules, reports, and dashboards.
HOD	Access to Students, Club Meetings, Attendance, Sports Registration Teams, Reports & Dashboards**.**
Faculty	Create student records, issue QR Club Pass, create club meetings,scan attendance, Reports & Dashboards**.**
Club Leader	Create club meetings, access meetings & attendance.
PT Head / PT Assistant	Enter team player names, update match scores. Access Sports Registration Teams only.
Store Manager	Full access to Campus Mart, Store Items, Store Sales, Sports Registration Teams, Reports & Dashboards.
Store Salesperson	Operate Campus Mart: scan items, manage cart, record Store Sales, handle Sports Registration forms.
Students	Access Club Meetings and Attendance; view available clubs and their Attendance details.

3. Business Process Mapping:

Student Club Workflow:

- Student expresses interest → Faculty collects details & fee → Creates student record → QR Pass generated → Student attends club meetings → Faculty scans QR → Attendance stored in Attendance tab.

Sports Registration Workflow:

- Student submits registration at Campus Mart → Salesperson enters details and payment → Record stored in Sports Registration Teams → PT Head/PT Assistant assign players and record match scores.

Campus Mart Workflow:

- Students purchase items → Salesperson scans items → Adds to cart → Confirms sale → Transaction stored in Store Sales → Stock updated in Store Items.

4. Industry Use Case:

- Sector: Education Management + Campus Retail
- Solution Type: Salesforce-based centralized student engagement system integrating:
  - Club & Event Management
  - Sports Registration & Team Management
  - Attendance Automation via QR
  - Campus Store Sales & Inventory Management

5. AppExchange & Technology Exploration:

- QR Code/ BarCode Scanning: Using JSQR / ZXing libraries for attendance and Items automation.
- Inventory Management: Custom LWCs for scanning, cart management, stock control.
- Automation: Flows, Apex class and Apex Triggers for automatic attendance recording, stock updates, sports registration, and email notifications.

Phase 2: Org Setup & Configuration

1. Salesforce Edition:

- Developer Edition (used for this project, which provides all core Salesforce features including custom objects, Apex programming, Lightning Web Components, automation tools, and record-level security — similar to Enterprise Edition, but limited to development and testing purposes).

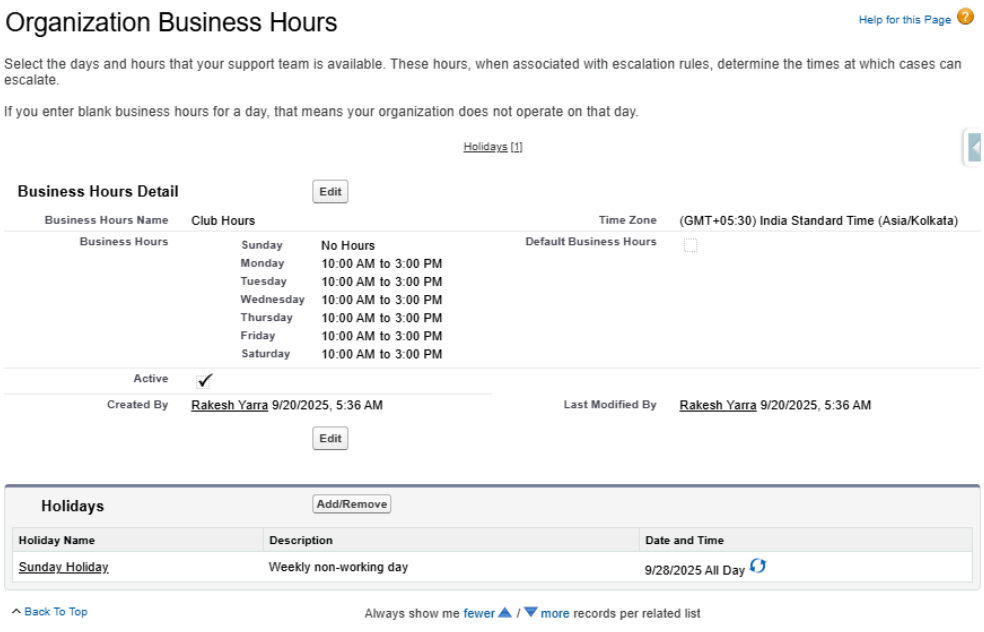
2. Company Profile Setup:

- **Company Name:** PRAG

3. Business Hours and Holidays:

Business Hours:

- **Club Hours: 10 AM – 3 PM (Mon–Sat)**



Organization Business Hours

Help for this Page

Select the days and hours that your support team is available. These hours, when associated with escalation rules, determine the times at which cases can escalate.

If you enter blank business hours for a day, that means your organization does not operate on that day.

Holidays [1]

Business Hours Detail Edit

Business Hours Name	Club Hours	Time Zone	(GMT+05:30) India Standard Time (Asia/Kolkata)
Business Hours	Sunday: No Hours Monday: 10:00 AM to 3:00 PM Tuesday: 10:00 AM to 3:00 PM Wednesday: 10:00 AM to 3:00 PM Thursday: 10:00 AM to 3:00 PM Friday: 10:00 AM to 3:00 PM Saturday: 10:00 AM to 3:00 PM	Default Business Hours	☐
Active	✓		
Created By	Rakesh Yarra 9/20/2025, 5:36 AM	Last Modified By	Rakesh Yarra 9/20/2025, 5:36 AM

Edit

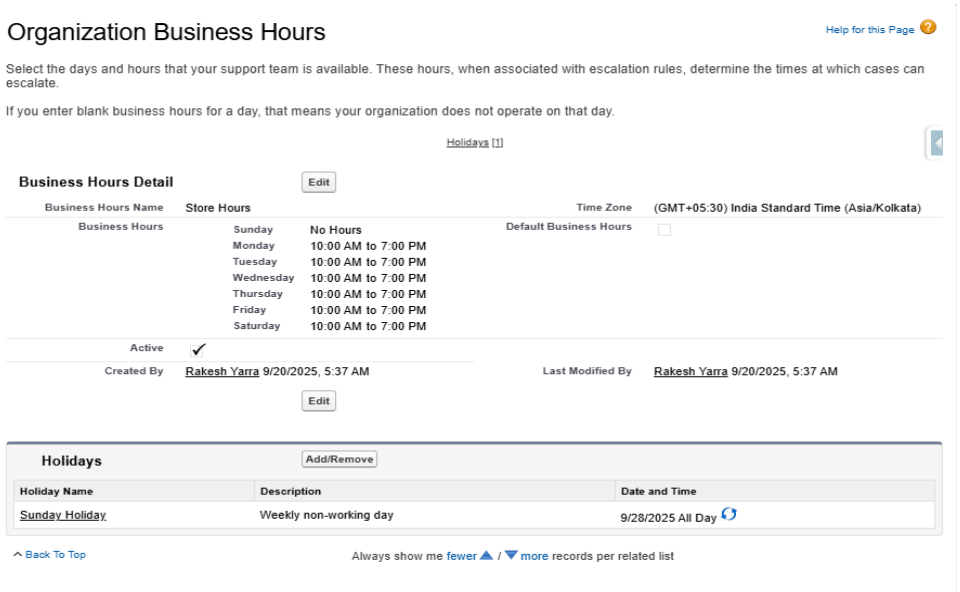
Holidays Add/Remove

Holiday Name	Description	Date and Time
Sunday Holiday	Weekly non-working day	9/28/2025 All Day

Back To Top

Always show me fewer ▲ / ▼ more records per related list

- **Store Hours: 10 AM – 11 PM (Mon–Sat)**



Organization Business Hours

Help for this Page

Select the days and hours that your support team is available. These hours, when associated with escalation rules, determine the times at which cases can escalate.

If you enter blank business hours for a day, that means your organization does not operate on that day.

Holidays [1]

Business Hours Detail Edit

Business Hours Name	Store Hours	Time Zone	(GMT+05:30) India Standard Time (Asia/Kolkata)
Business Hours	Sunday: No Hours Monday: 10:00 AM to 7:00 PM Tuesday: 10:00 AM to 7:00 PM Wednesday: 10:00 AM to 7:00 PM Thursday: 10:00 AM to 7:00 PM Friday: 10:00 AM to 7:00 PM Saturday: 10:00 AM to 7:00 PM	Default Business Hours	☐
Active	✓		
Created By	Rakesh Yarra 9/20/2025, 5:37 AM	Last Modified By	Rakesh Yarra 9/20/2025, 5:37 AM

Edit

Holidays Add/Remove

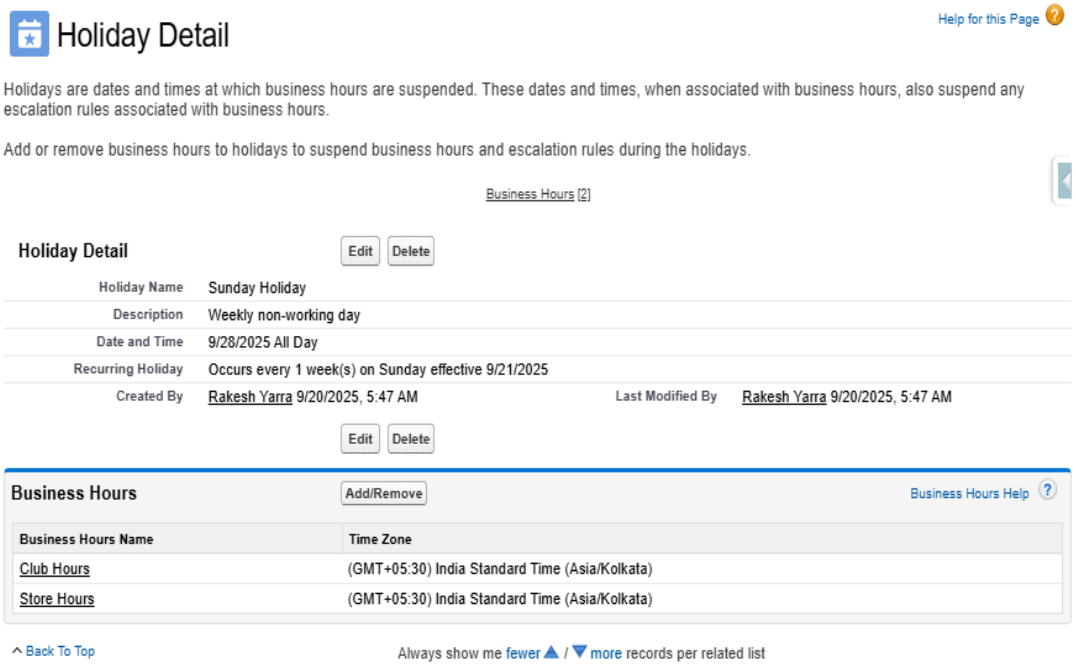
Holiday Name	Description	Date and Time
Sunday Holiday	Weekly non-working day	9/28/2025 All Day

Back To Top

Always show me fewer ▲ / ▼ more records per related list

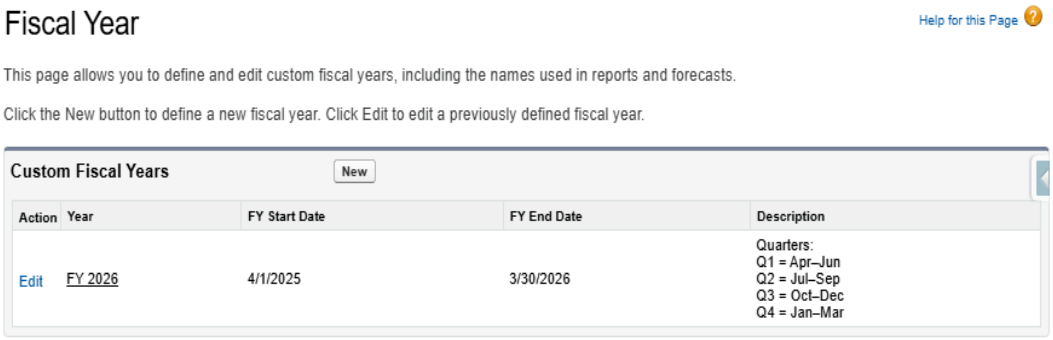
Holidays:

- **Sunday Holiday:** Assign to both Business Hours



4. Fiscal Year Settings:

- **FY 2026**



5. User Setup & Licenses:

- A user license is required for every user and sets the baseline for which features the user can access.

All Users

Help for this Page

On this page you can create, view, and manage users.

To get more licenses, use the Your Account app. Let's Go

View: All Users Edit | Create New View

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All

New User | Reset Password(s) | Add Multiple Users

Action	Full Name ↑	Alias	Username	Role	Active	Profile
Edit	Chatter Expert	Chatter	chatty.00dgk000006hfg1uac.reke01xruxqk@chatter.salesforce.com			Chatter Free User
Edit | Login	csefaculty, Raja	csefacul	faculty1@college.test	Faculty - CSE	✓	Faculty Profile
Edit | Login	ecestudent, satya	secef	studentece1@college.test	Students - ECE	✓	Club Leader Profile
Edit | Login	EPIC, OrgFarm	OEPIC	epic.b11bb8f367bd@orgfarm.com	Store Manager	✓	System Administrator
Edit | Login	K, cheru	ck	rryservicenowy230825@gmail.com	HOD – ECE	✓	Standard Platform User
Edit	K, Ram	RK	ramcharank@gmail.com			Chatter Free User
Edit	K, upasana	uK	upasanak@gmail.com			Chatter Moderator User
Edit	User, Integration	integ	integration@00dgk000006hfg1uac.com	Store Manager	✓	Analytics Cloud Integration User
Edit	User, Security	sec	insightssecurity@00dgk000006hfg1uac.com		✓	Analytics Cloud Security User
Edit	Yarra, Rakesh	rak	rakeshvsworld307@agentforce.com	Dean	✓	System Administrator

New User | Reset Password(s) | Add Multiple Users

- **Faculty User:** Manage student entries, club meetings, Club Attendance Scanning, attendance, Reports and Dashboards

User

Raja csefaculty

User ProfileHelp for this Page

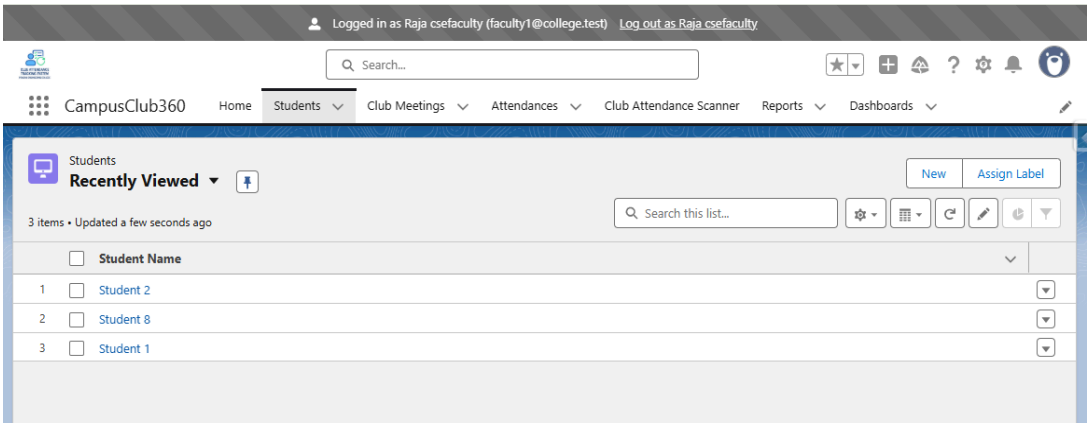
Permission Set Assignments [3] | Permission Set Assignments: Activation Required [0] | Permission Set Group Assignments [0] | Permission Set License Assignments [0] | Personal Groups [0] | Public Group Membership [1] | Queue Membership [0] | Team [0] | Managers in the Role Hierarchy [3] | OAuth Apps [0] | Third-Party Account Links [0] | Built-in Authenticators [0] | Installed Mobile Apps [0] | Authentication Settings for External Systems [0] | Login History [0+] | User Provisioning Accounts [0]

User Detail

Edit | Sharing | Reset Password | Login | Freeze | View Summary

Name	Raja csefaculty	Role	Faculty - CSE
Alias	csefacul	User License	Salesforce
Email	yarrarakesh39@gmail.com [Verified]	Profile	Faculty Profile
Username	faculty1@college.test	Active	✓
Nickname	User17516102830391672931	Marketing User	
Title		Offline User	
Company		Knowledge User	
Department		Flow User	
Division		Service Cloud User	
Address		Site.com Contributor User	
Time Zone	(GMT-07:00) Pacific Daylight Time (America/Los_Angeles)	Site.com Publisher User	
Locale	English (United States)	WDC User	
Language	English	Mobile Push Registrations	View
Delegated Approver		Data.com User Type	
Manager		Accessibility Mode (Classic Only)	
Receive Approval Request Emails	Only if I am an approver	Debug Mode	
Federation ID		High-Contrast Palette on Charts	
App Registration: One-Time Password Authenticator		Load Lightning Pages While Scrolling	✓
App Registration: Salesforce Authenticator		Salesforce CRM Content User	✓
Security Key (U2F or WebAuthn)		Receive Salesforce CRM Content Email Alerts	✓
Lightning Login		Receive Salesforce CRM Content Alerts as Daily Digest	✓

Verify:



6. Profiles:

- **Dean Profile** → System Admin equivalent.

Profile Detail Edit | Clone | Delete | View Users

Name	Dean Profile		
User License	Salesforce	Custom Profile	✓
Description			
Created By	Rakesh Yarra, 9/23/2025, 2:04 AM	Modified By	Rakesh Yarra, 9/23/2025, 2:04 AM

- **HOD Profile** → Access to Students, Club Meetings, Attendance, Club Attendance Scanner, Sports Registration Teams, Reports & Dashboards.

Profile Detail Edit | Clone | Delete | View Users

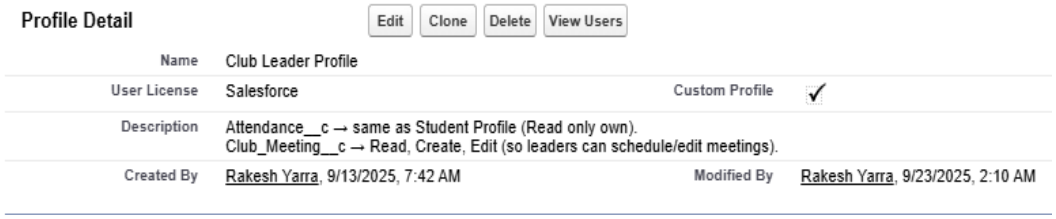
Name	HOD Profile		
User License	Salesforce	Custom Profile	✓
Description			
Created By	Rakesh Yarra, 9/20/2025, 7:14 AM	Modified By	Rakesh Yarra, 9/21/2025, 7:13 AM

- **Faculty Profile** → Access to Students, Club Meetings, Attendance, and QR Scanner tab.

Profile Detail Edit | Clone | Delete | View Users

Name	Faculty Profile		
User License	Salesforce	Custom Profile	✓
Description	Student__c: Read, Create, Edit. Club_Meeting__c: Read, Create, Edit. Attendance__c: Read, Create (faculty scans attendance), maybe Edit/Delete if needed.		
Created By	Rakesh Yarra, 9/13/2025, 7:08 AM	Modified By	Rakesh Yarra, 9/21/2025, 7:13 AM

- **Club Leader Profile** → Access to Club Meetings & Attendance only.

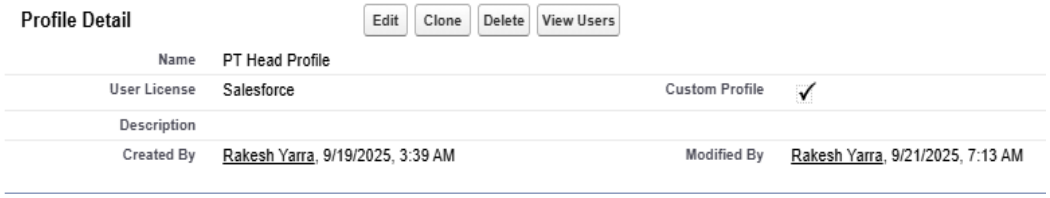


Profile Detail

Edit | Clone | Delete | View Users

Name	Club Leader Profile		
User License	Salesforce	Custom Profile	✓
Description	Attendance__c → same as Student Profile (Read only own). Club_Meeting__c → Read, Create, Edit (so leaders can schedule/edit meetings).		
Created By	Rakesh Yarra, 9/13/2025, 7:42 AM	Modified By	Rakesh Yarra, 9/23/2025, 2:10 AM

- **PT Head Profile** → Access to Sports Registration Teams (edit rights).

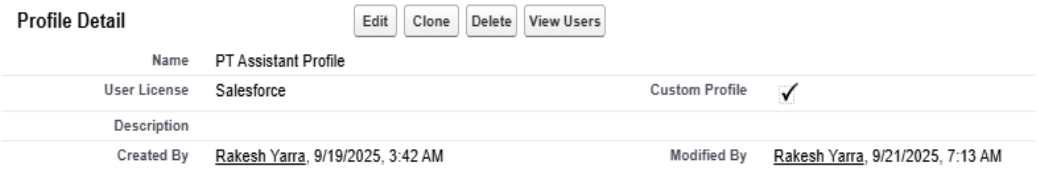


Profile Detail

Edit | Clone | Delete | View Users

Name	PT Head Profile		
User License	Salesforce	Custom Profile	✓
Description			
Created By	Rakesh Yarra, 9/19/2025, 3:39 AM	Modified By	Rakesh Yarra, 9/21/2025, 7:13 AM

- **PT Assistant Profile** → Access to Sports Registration Teams (edit rights).

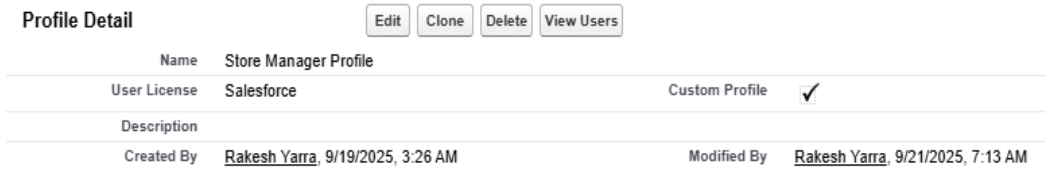


Profile Detail

Edit | Clone | Delete | View Users

Name	PT Assistant Profile		
User License	Salesforce	Custom Profile	✓
Description			
Created By	Rakesh Yarra, 9/19/2025, 3:42 AM	Modified By	Rakesh Yarra, 9/21/2025, 7:13 AM

- **Store Manager Profile** → Access to Campus Mart, Store Items, Store Sales, Sports Registration Teams, Reports & Dashboards.

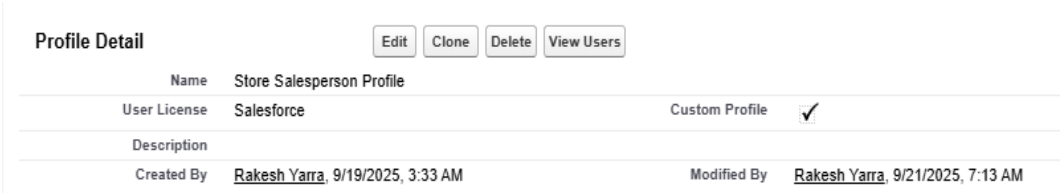


Profile Detail

Edit | Clone | Delete | View Users

Name	Store Manager Profile		
User License	Salesforce	Custom Profile	✓
Description			
Created By	Rakesh Yarra, 9/19/2025, 3:26 AM	Modified By	Rakesh Yarra, 9/21/2025, 7:13 AM

- **Store Salesperson Profile** → Access to Campus Mart, Store Items, Store Sales.

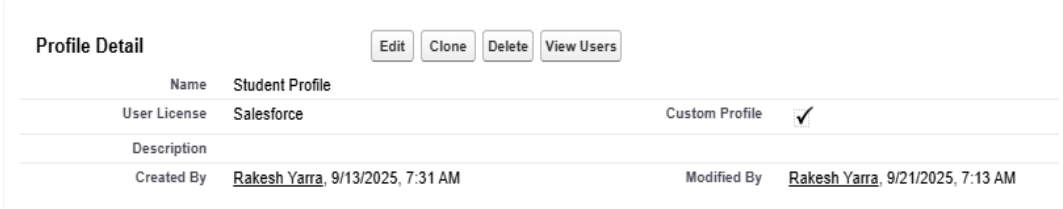


Profile Detail

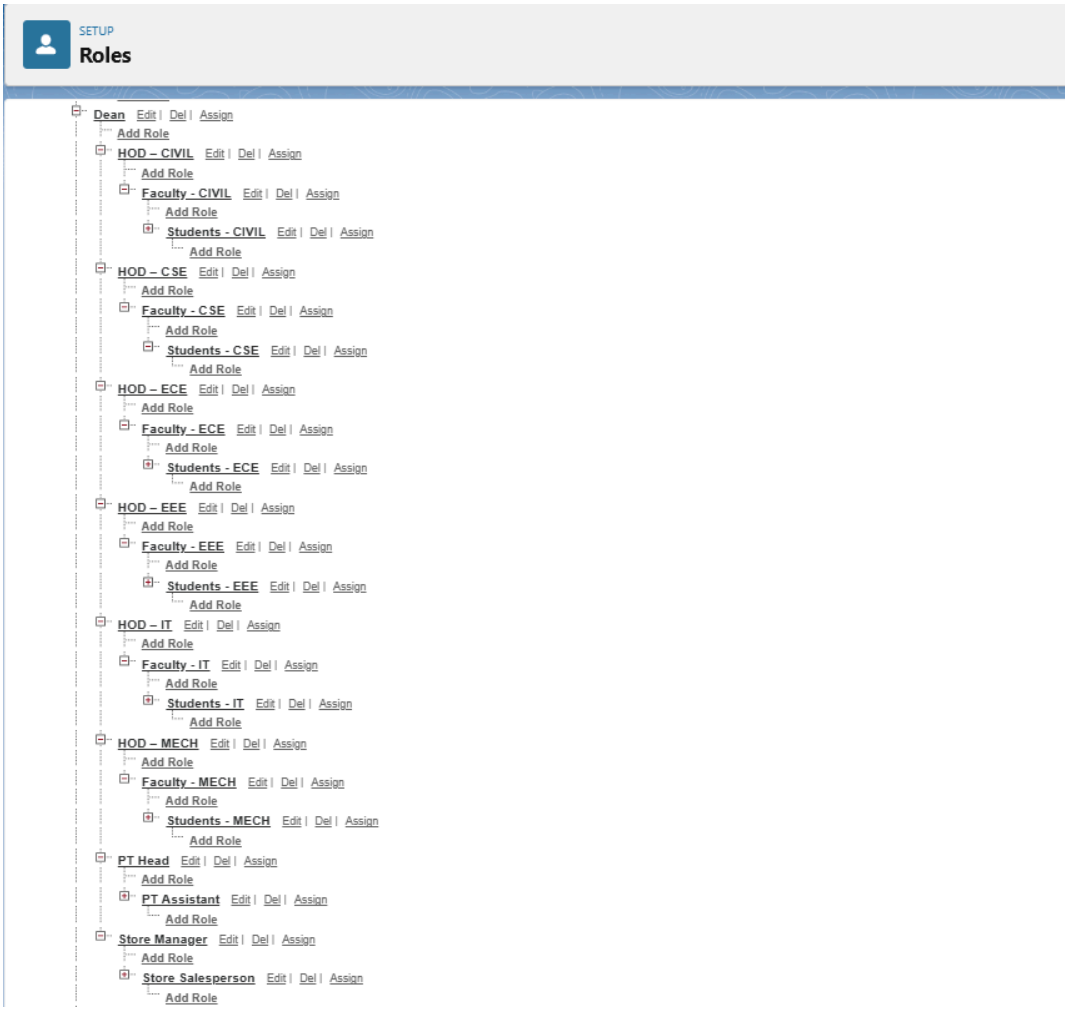
Edit | Clone | Delete | View Users

Name	Store Salesperson Profile		
User License	Salesforce	Custom Profile	✓
Description			
Created By	Rakesh Yarra, 9/19/2025, 3:33 AM	Modified By	Rakesh Yarra, 9/21/2025, 7:13 AM

- **Student Profile** → Access to view Club Meetings and Attendance (read-only for their records).



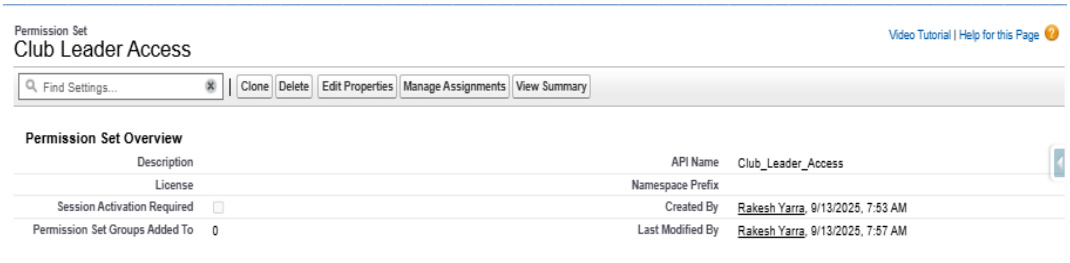
7. Roles:



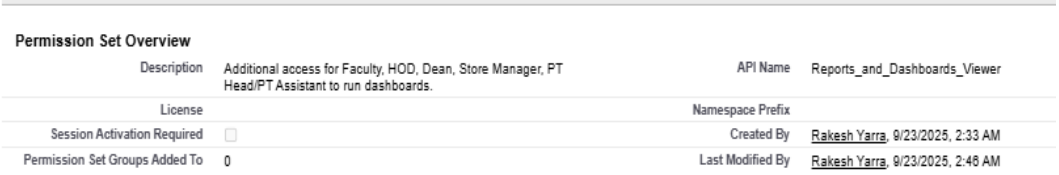
- **Dean:** Top of hierarchy, has access to all data.
- **HOD:** Oversees students, clubs, attendance, reports, dashboards, and sports teams.
- **Faculty:** Manage student entries, club meetings, attendance, reports and dashboards.
- **Club Leaders:** Manage club meetings, and attendance.
- **PT Roles:** Restricted to Sports Registration Teams.
- **Store Roles:** Manage Campus Mart, sales, and inventory.
- **Students:** Lowest level, access only to their relevant modules.

8. Permission Sets:

- **Club Leader Access** → Special permission for students promoted as Club Leaders to create/manage meetings.



- **Reports and Dashboards Viewer** → Additional access for Faculty, HOD, Dean, Store Manager, PT Head/PT Assistant to run reports and dashboards.



9. Organization-Wide Defaults (OWD)

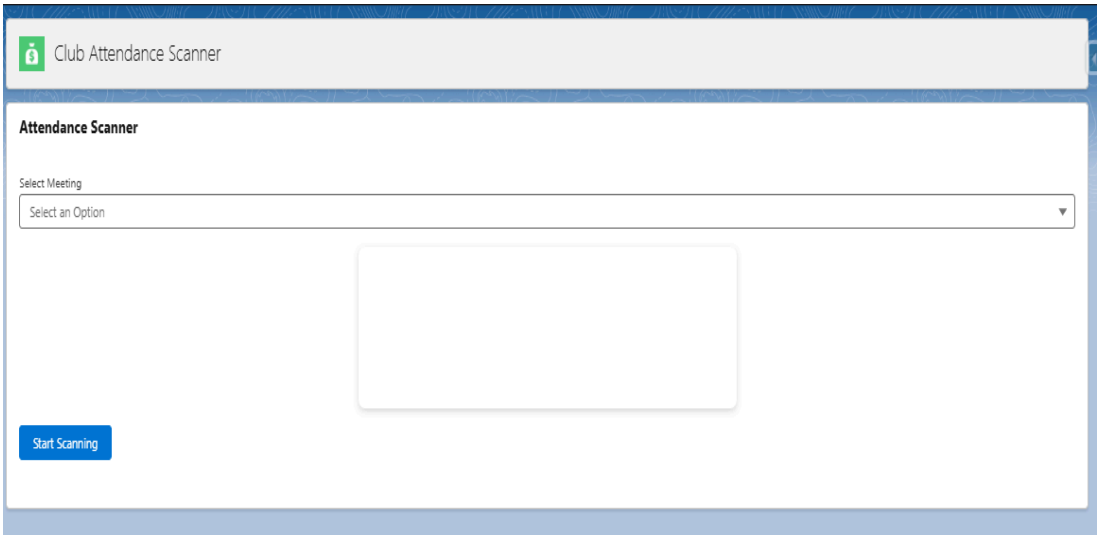
Object	OWD Setting	Justification
Students	Private	Only faculty, HOD, Dean should manage student data.
Attendance	Private	Sensitive record; visible only to Faculty, HOD, and Dean.
Club Meetings	Public Read/Write	Faculty & Club Leaders collaborate on scheduling.
Sports Registration Teams	Public Read/Write	Shared between PT Head, PT Assistant, Store Manager.
Sports Items	Private	Store Manager controls; Salesperson read-only.
Sports Sales	Private	Managed by Store Manager; partially visible to Salesperson.
Reports/Dashboards	Controlled by Parent	Dean, HOD, Store Manager, PT roles get access.

10. Sharing Rules:

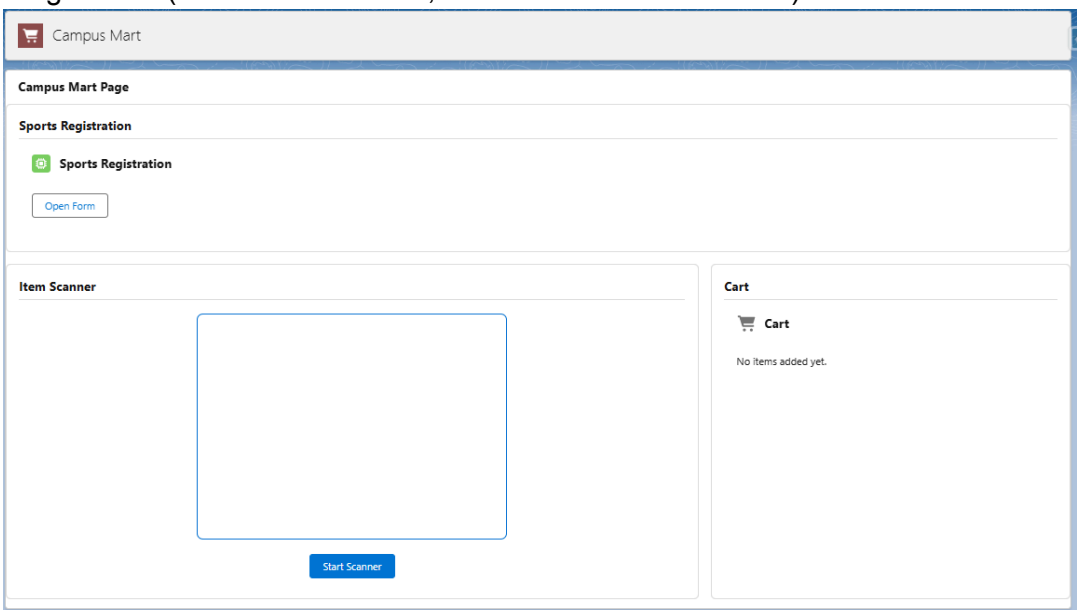
- **Attendance Sharing:** Share Attendance with Faculty, HODs, and Dean for oversight.
- **Club Meeting Sharing:** Share with Students, Club Leaders, Faculty, HOD and Dean for participation management.
- **Sports Registration Teams Sharing:** Share with PT Head and PT Assistant.
- **Store Sales Sharing:** Share with Store Manager.
- **Store Item Sharing:** Share with Store Manager (Read/Write) and Store Salespersons (Read-Only).
- **Student Sharing:** Share Faculty with HODs and Dean for academic oversight

11. Tab & App Setup:

- **Custom App:** CampusClub360
- **Tab Order:**
  - Students
  - Club Meetings
  - Attendance
  - Club Attendance Scanner (LWC)
  - CampusMart (LWC)
  - Sports Registration Teams
  - Store Items
  - Store Sales
  - Reports
  - Dashboards
- **Club Attendance Scanner:**
  - Custom LWC (AttendanceScanner) used by faculty to scan student QR passes.
  - Automatically creates records in the Attendance tab.



- **CampusMart Lightning Page Layout:**
  - Top: Sports Registration Form (Salesperson fills sport, team name, players, registration fee).
  - Left: Item Scanner (scan store items).
  - Right: Cart (shows added items, creates Store Sales record).



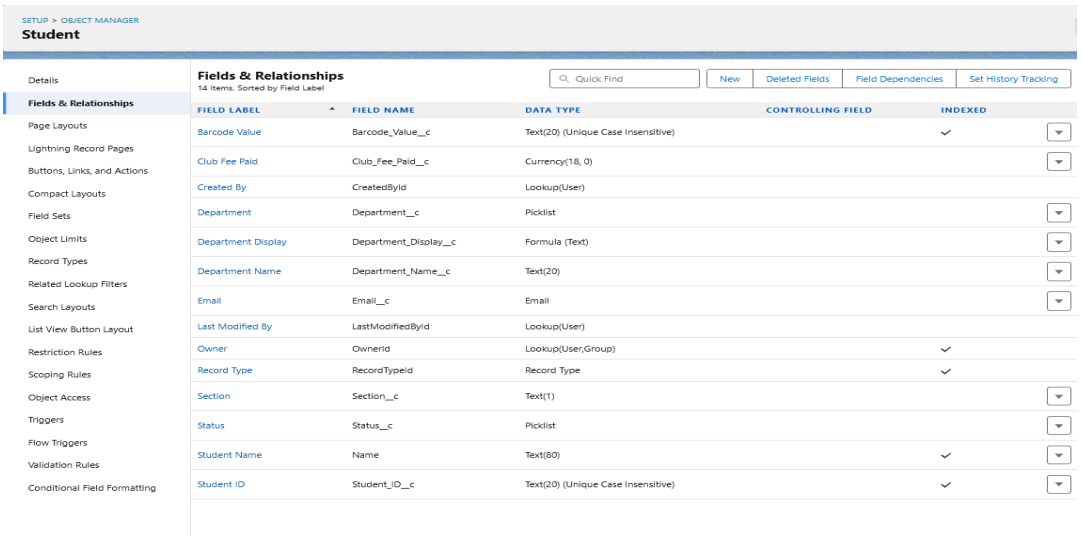
Phase 3: Data Modeling & Relationships

1. Custom Objects and Fields:

- **Student__c:** Stores details of students who register for clubs, meetings, and events.

  Key Fields:

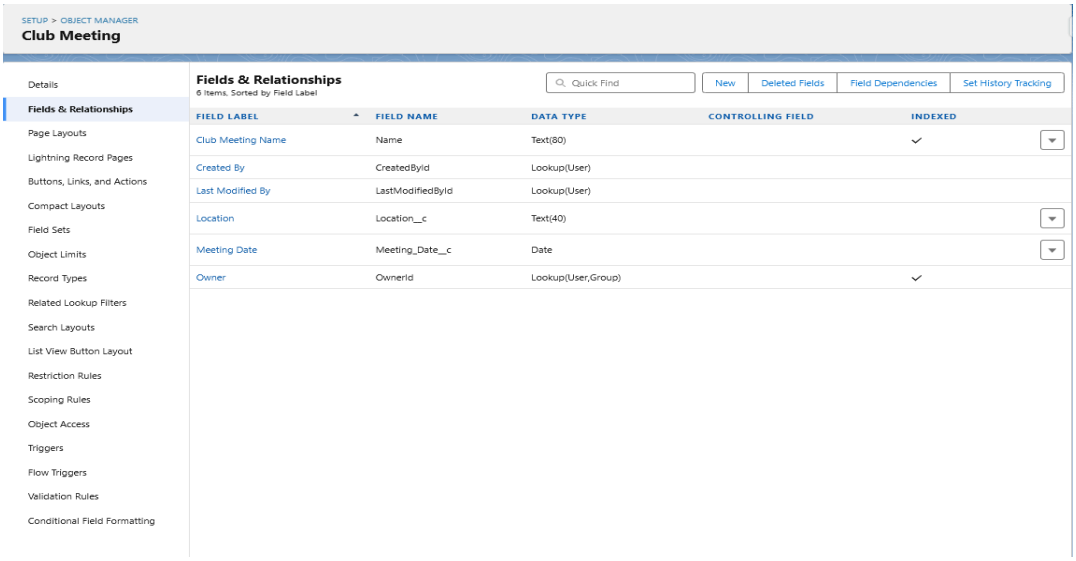
  - Student Name (Text, Required)
  - Student ID (Unique, Text 20)
  - Email (Email)
  - Barcode Value (Unique, Text 20)
  - Department (Picklist – for Normal Students)
  - Department Name (Text – for Others RecordType)
  - Department Display (Formula(Text))
  - Section (Text 1)
  - Status (Picklist)
  - Club Fee Paid (Currency 18,0)
  - System Fields: Owner, Created By, Last Modified By, Record Type



- **Club_Meeting__c:** Represents club meetings/webinars created by faculty or club leaders.

  Key Fields:

  - Club Meeting Name (Text 80)
  - Location (Text 40)
  - Meeting Date (Date)
  - System Fields: Owner, Created By, Last Modified By

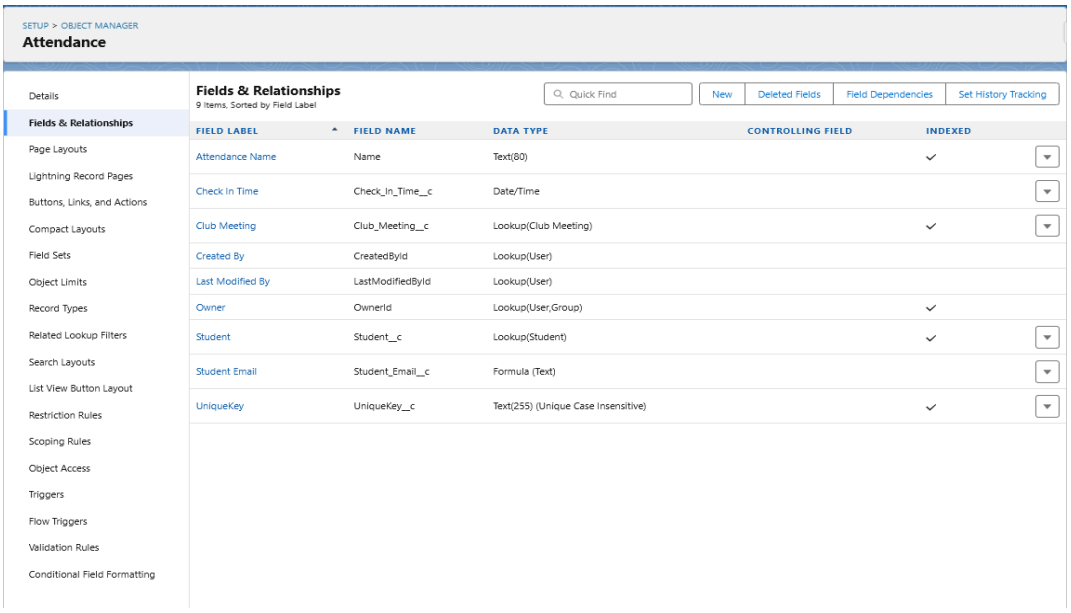


- **Attendance__c:** Tracks student attendance for club meetings and events.

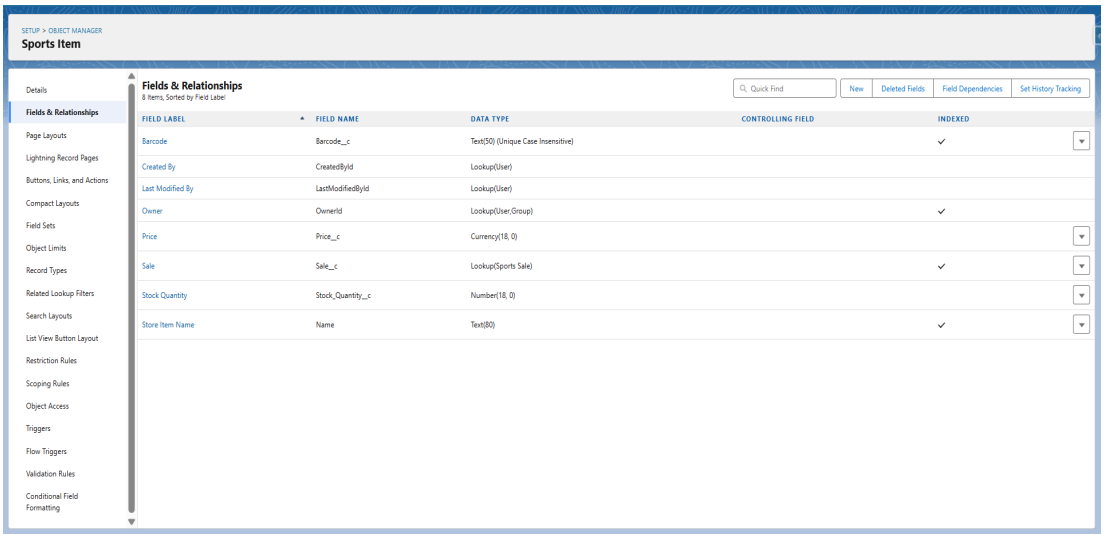
  Key Fields:

  - Attendance Name (Text 80)
  - Check In Time (Date/Time)
  - UniqueKey (Unique Text 255)
  - Student Email (Formula Text, from Student)

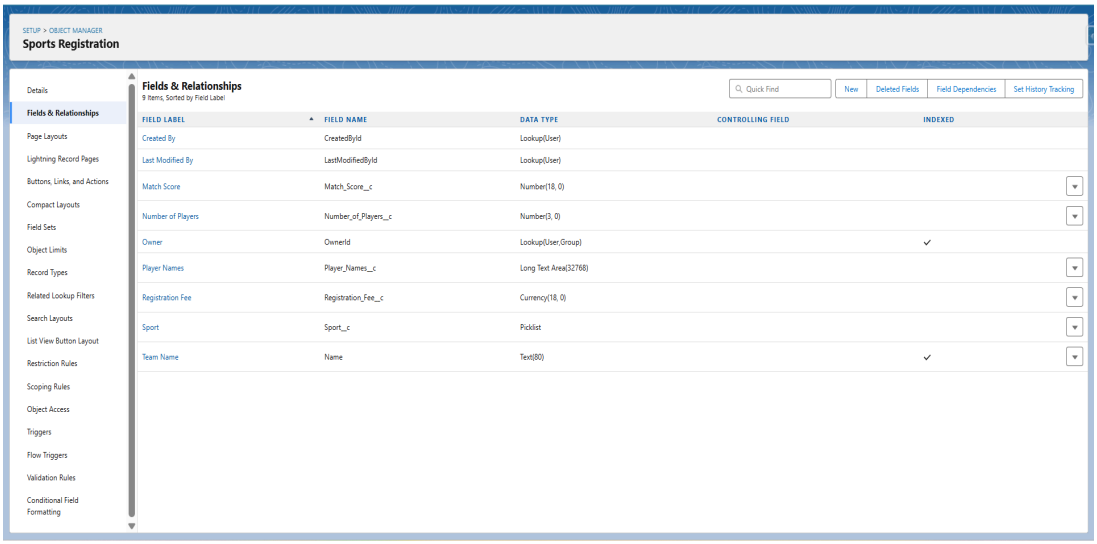
- Relationships: Student (Lookup), Club Meeting (Lookup)
- System Fields: Owner, Created By, Last Modified By



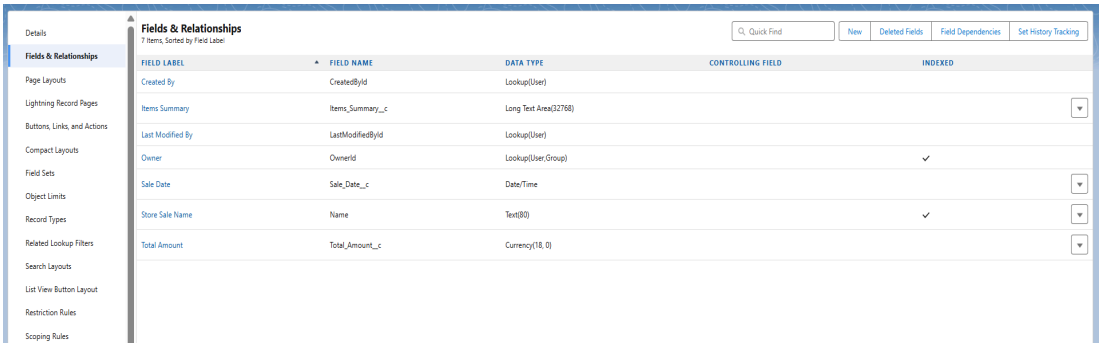
- **Sports_Item__c:** Manages all store items (sports, books, pens, stationery, etc.).
  Key Fields:
  - Store Item Name (Text 80)
  - Barcode (Unique Text 50)
  - Price (Currency 18,0)
  - Stock Quantity (Number 18,0)
  - Relationship: Sale (Lookup → Sports_Sale__c)
  - System Fields: Owner, Created By, Last Modified By



- **Sports_Registration__c:** Captures team registrations for sports activities.
  Key Fields:
  - Team Name (Text 80)
  - Sport (Picklist)
  - Registration Fee (Currency 18,0)
  - Number of Players (Number 3,0)
  - Player Names (Long Text Area)
  - Match Score (Number 18,0)
  - System Fields: Owner, Created By, Last Modified By

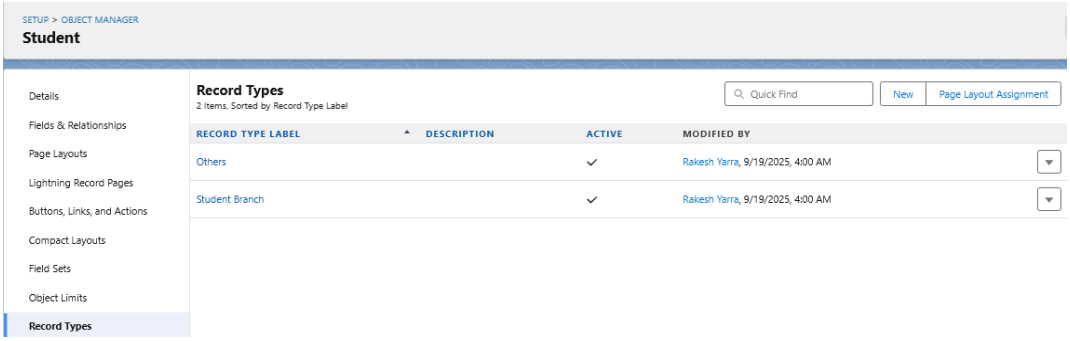


- **Sports_Sale__c:** Tracks sales transactions of items purchased from Campus Mart.
  Key Fields:
  - Store Sale Name (Text 80)
  - Sale Date (Date/Time)
  - Total Amount (Currency 18,0)
  - Items Summary (Long Text Area)
  - System Fields: Owner, Created By, Last Modified By



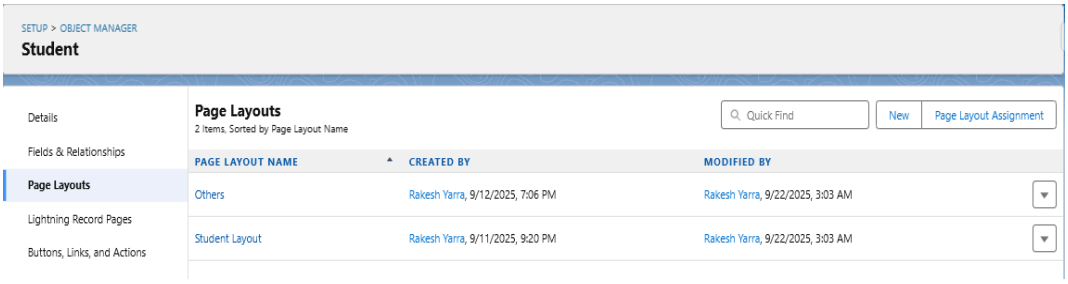
2. Record Types:

- **Student__c**
  - **Student Branch** → Uses Student Layout (Department Picklist available).
  - **Others** → Uses Others Layout (Department Name text required).



3. Page Layouts:

- **Student__c**
  - Student Layout → Department Picklist, no Department Name.
  - Others Layout → Department Name field visible.



4. Relationships:

- **Student__c** → **Attendance__c:** Lookup (one student can have multiple attendance records).
- **Club_Meeting__c** → **Attendance__c:** Lookup (one meeting can have multiple attendance records).
- **Sports_Sale__c** → **Sports_Item__c:** Lookup (one sale can include multiple items).

Phase 4: Process Automation (Admin)

1. Validation Rules:

- **Student__c**
  - **Require_Department_Name_For_Others**
    - Error Location: Department Name
    - Rule: Department Name is required if RecordType = Others.

- Error Message: “Department Name is required for Others record type.”

Student Validation Rule

Back to Student

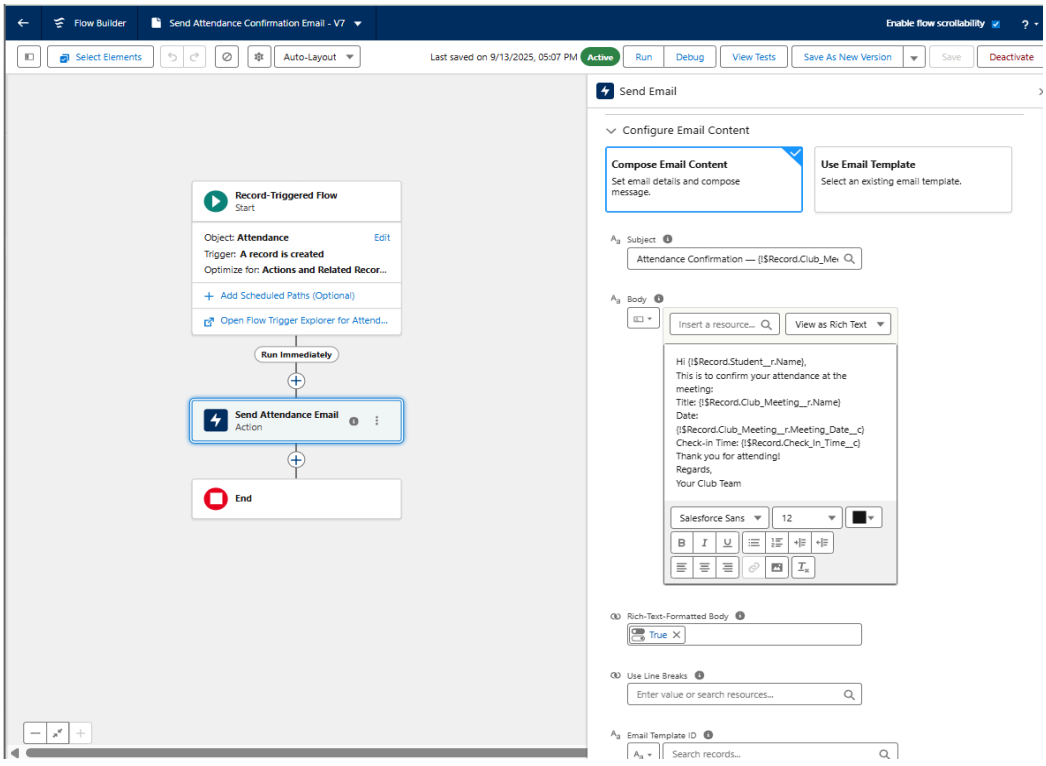
Validation Rule Detail Edit Clone

Rule Name	Require_Department_Name_For_Others	Active	✓
Error Condition Formula	AND($RecordType.Name = "Others", ISBLANK(Department_Name__c))		
Error Message	Department Name is required for Others record type.	Error Location	Department Name
Description	$RecordType.Name = "Others" → only applies when the record type is "Others". ISBLANK(TEXT(Department_Name__c)) → checks if Department Name (Text) is empty.		
Created By	Rakesh Yarra, 9/13/2025, 6:30 AM	Modified By	Rakesh Yarra, 9/13/2025, 6:30 AM

Edit Clone

2. Flow Builder:

- **Send Attendance Confirmation Flow**
  - **Type:** Record-Triggered Flow (on Attendance__c after insert).
  - **Logic:** When attendance is marked (via QR scan), send a confirmation email to the student.
  - **Email Content:** Includes Meeting Name, Date, and Check-In Time.

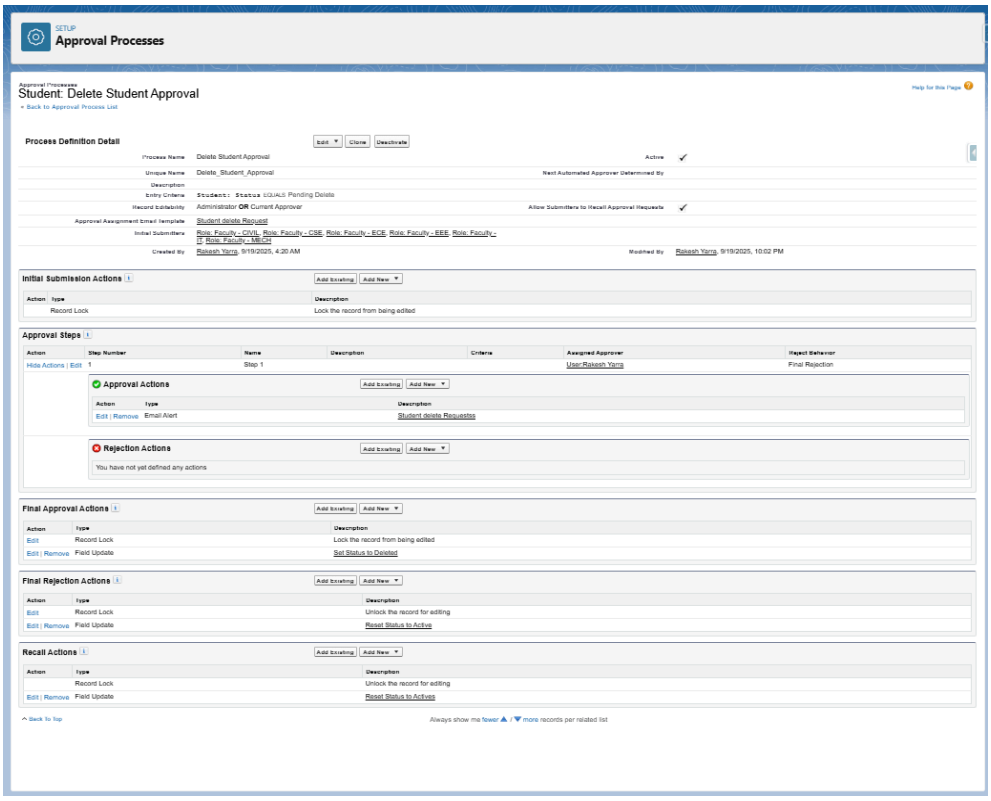


3. Approval Processes:

- **Delete Student Approval**
  - **Initiator:** Faculty member tries to delete a Student record, He has to Submit

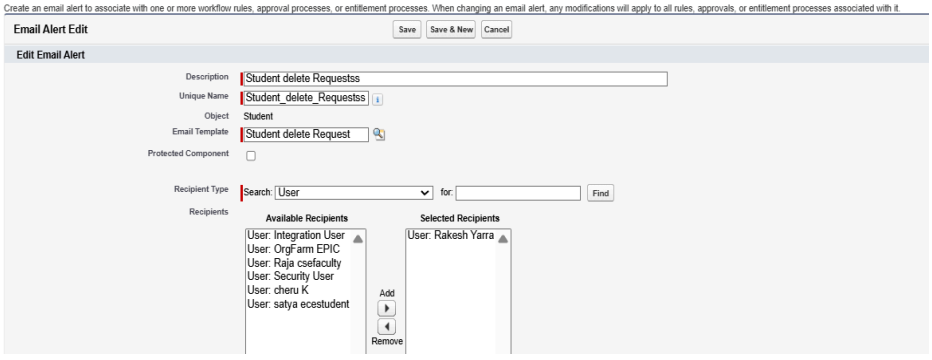
Approval Request to Dean.

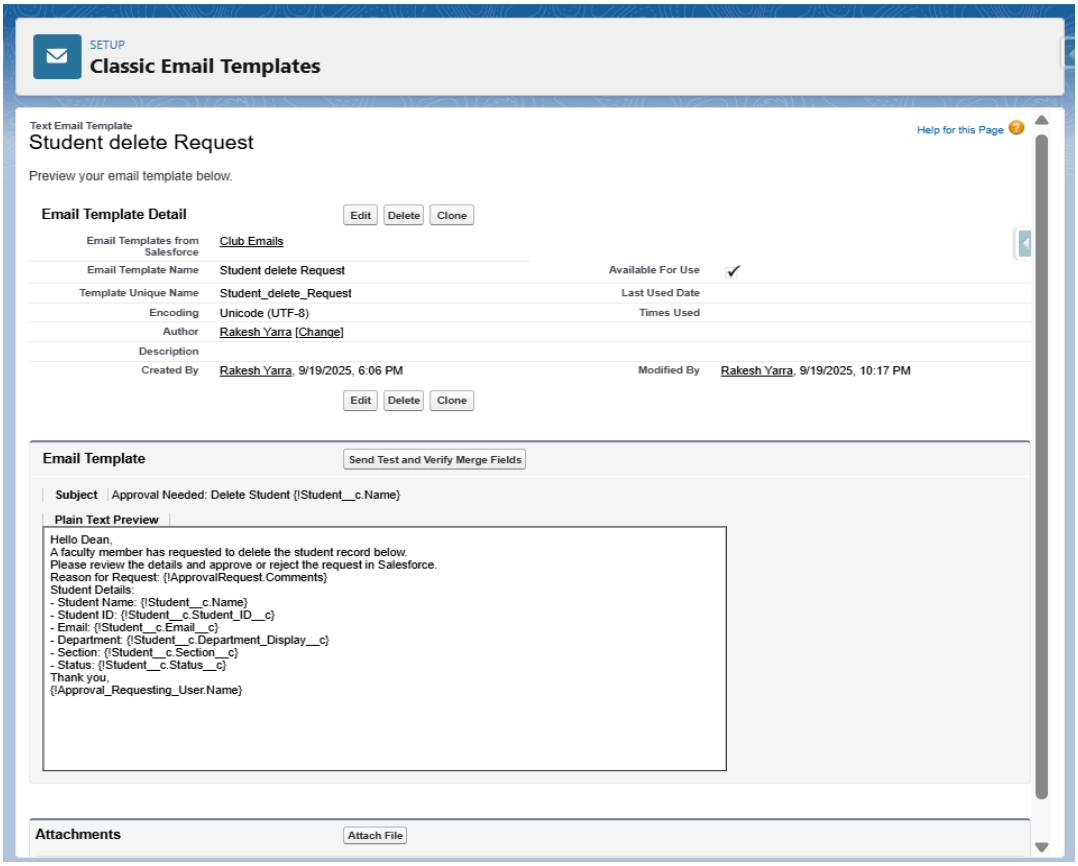
- **Approval:** Routed to **Dean** for approval.
- **Email Notification:** Sent to Dean with student details.
- **Action:**
  - **If Approved** → Faculty Can delete Student record.
  - **If Rejected** → Faculty don't have to delete Student record.



4. Email Templates & Alerts:

- **Student Delete Request**
  - Sent to Dean when Faculty requests Student deletion.





Phase 5: Apex Programming (Developer)

Apex Classes:

- **AttendanceController**

  - Purpose: Manage attendance records through LWC (QR scan integration).
  - Key Functions: Create attendance, fetch student details, validate duplicates.

```
public with sharing class AttendanceController {
    // Wrapper to return attendance record + flag whether it was newly created
    public class AttendanceResult {
        @AuraEnabled public Attendance__c attendance;
        @AuraEnabled public Boolean isNew;
        public AttendanceResult(Attendance__c a, Boolean n) {
            this.attendance = a;
            this.isNew = n;
        }
    }

    @AuraEnabled(cacheable=true)
    public static List<Club_Meeting__c> getMeetings() {
    return [
```

```
    SELECT Id, Name, Meeting_Date__c
    FROM Club_Meeting__c
    ORDER BY Meeting_Date__c ASC
 ];
 }


@AuraEnabled(cacheable=true)
public static Student__c findStudentByBarcode(String barcodeValue) {
    List<Student__c> s = [
        SELECT Id, Name, Student_ID__c, Barcode_Value__c
        FROM Student__c
        WHERE Barcode_Value__c = :barcodeValue
        LIMIT 1
    ];
    return s.isEmpty() ? null : s[0];
}

@AuraEnabled
public static AttendanceResult createAttendance(Id studentId, Id meetingId) {
    // Attempt to find existing attendance
    List<Attendance__c> existing = [
        SELECT Id, Check_In_Time__c, Student__r.Name, Club_Meeting__r.Name
        FROM Attendance__c
        WHERE Student__c = :studentId AND Club_Meeting__c = :meetingId
        LIMIT 1
    ];
    if (!existing.isEmpty()) {
        return new AttendanceResult(existing[0], false);
    }

    // Create new Attendance (also populate UniqueKey__c if you created it)
    Attendance__c a = new Attendance__c(
        Student__c = studentId,
        Club_Meeting__c = meetingId,
        Check_In_Time__c = System.now()
    );

    // If you have a UniqueKey__c text unique field, set it here:
    // a.UniqueKey__c = studentId + '-' + meetingId;

    insert a;
```

```
        // Re-query to include relationship fields for the UI
        a = [
            SELECT Id, Check_In_Time__c, Student__r.Name, Club_Meeting__r.Name
            FROM Attendance__c
            WHERE Id = :a.Id
            LIMIT 1
        ];

        return new AttendanceResult(a, true);
    }
}
```

- **SportsRegistrationController:**
  - Purpose: Handle sports team registrations.
  - Key Functions: Save team details on Sports Registration Teams

```
public with sharing class SportsRegistrationController {
    @AuraEnabled
    public static void createRegistration(String sport, String teamName, Integer players,
Decimal fee) {
        Sports_Registration__c reg = new Sports_Registration__c();
        reg.Name = teamName; // use standard Name
        reg.Sport__c = sport;
        reg.Number_of_Players__c = players;
        reg.Registration_Fee__c = fee;
        insert reg;
    }
}
```

- **SportsItemController**
  - Purpose: Manage inventory of store items.
  - Key Functions: Retrieve items, update stock, prevent duplicate barcodes.

```
public with sharing class SportsItemController {

    @AuraEnabled(cacheable=true)
    public static Sports_Item__c getItemByBarcode(String barcode) {
        try {
            List<Sports_Item__c> items = [
                SELECT Id, Name, Price__c, Barcode__c, Stock_Quantity__c
                FROM Sports_Item__c
                WHERE Barcode__c = :barcode
                LIMIT 1
```

```
            ];
            if (!items.isEmpty()) {
                return items[0];
            }
            return null;
        } catch (Exception e) {
            throw new AuraHandledException('Error fetching item: ' + e.getMessage());
        }
    }

    // Helper inner class to receive cart data from LWC
    public class CartItem {
        @AuraEnabled public String id;
        @AuraEnabled public String name;
        @AuraEnabled public Decimal price;
        @AuraEnabled public String barcode;
        @AuraEnabled public Integer quantity;
    }

    @AuraEnabled
    public static String finalizeSale(List<CartItem> items, Decimal totalAmount) {
        if (items == null || items.isEmpty()) {
            throw new AuraHandledException('Cart is empty. Nothing to finalize.');
        }

        try {
            // Create Sale record
            Sports_Sale__c sale = new Sports_Sale__c();
            sale.Name = 'Sale - ' + DateTime.now().format('yyyy-MM-dd HH:mm');
            sale.Sale_Date__c = System.now();
            sale.Total_Amount__c = totalAmount;
            insert sale;

            // Build item summary text
            String summary = '';
            List<Sports_Item__c> updates = new List<Sports_Item__c>();

            for (CartItem ci : items) {
                summary += ci.name + ' (x' + ci.quantity + ') - ₹' + (ci.price * ci.quantity) + '\n';

                if (ci.id != null) {
                    // Try to fetch by Id
                    List<Sports_Item__c> results = [
```

```
                    SELECT Id, Stock_Quantity__c
                    FROM Sports_Item__c
                    WHERE Id = :ci.id
                    LIMIT 1
                ];
                if (!results.isEmpty()) {
                    Sports_Item__c item = results[0];
                    Integer currentStock = item.Stock_Quantity__c == null ? 0 :
Integer.valueOf(item.Stock_Quantity__c);
                    item.Stock_Quantity__c = Math.max(0, currentStock - ci.quantity);
                    updates.add(item);
                }
            }
        }

        if (!updates.isEmpty()) {
            update updates;
        }

        // Update sale with summary
        sale.Items_Summary__c = summary;
        update sale;

        return 'Sale finalized successfully. Id: ' + sale.Id;
    } catch (Exception e) {
        throw new AuraHandledException('Failed to finalize sale: ' + e.getMessage());
    }
  }
}
```

- **SportsSaleController:**
  - Purpose: Manage sales transactions in Campus Mart.
  - Key Functions: Create sales records, calculate total amount, integrate with cart.

```
public with sharing class SportsSaleController {

  // Wrapper to map cart items coming from LWC
  public class CartItemWrapper {
      @AuraEnabled public String id;
      @AuraEnabled public String name;
      @AuraEnabled public Decimal price;
      @AuraEnabled public String barcode;
      @AuraEnabled public Integer quantity;
```

```
    }

    @AuraEnabled
    public static Id finalizeSale(String itemsJson, Decimal totalAmount) {
        if (String.isBlank(itemsJson)) {
            throw new AuraHandledException('Cart is empty. Cannot finalize sale.');
        }

        // Deserialize JSON to wrapper list
        List<CartItemWrapper> items =
            (List<CartItemWrapper>) JSON.deserialize(itemsJson,
List<CartItemWrapper>.class);

        if (items.isEmpty()) {
            throw new AuraHandledException('Cart is empty. Cannot finalize sale.');
        }

        // Build summary for Sale record
        String summary = '';
        for (CartItemWrapper item : items) {
            summary += item.name + ' (x' + item.quantity + ') - ₹' +
                    String.valueOf(item.price * item.quantity) + '\n';
        }

        // Create Sale record
        Sports_Sale__c sale = new Sports_Sale__c();
        sale.Name = 'Sale - ' + DateTime.now().format('yyyy-MM-dd HH:mm');
        sale.Sale_Date__c = System.now();
        sale.Total_Amount__c = totalAmount;
        sale.Items_Summary__c = summary;
        insert sale;

        // Reduce stock
        List<Sports_Item__c> updates = new List<Sports_Item__c>();
        for (CartItemWrapper item : items) {
            Sports_Item__c dbItem = [
                SELECT Id, Stock_Quantity__c
                FROM Sports_Item__c
                WHERE Id = :item.id
                LIMIT 1
            ];
            if (dbItem.Stock_Quantity__c >= item.quantity) {
                dbItem.Stock_Quantity__c -= item.quantity;
```

```
                updates.add(dbItem);
            } else {
                throw new AuraHandledException(
                    'Not enough stock for item: ' + item.name
                );
            }
        }
        if (!updates.isEmpty()) {
            update updates;
        }

        return sale.Id;
    }
}
```

2. Scheduler Apex:

- **DailyMeetingCleanupScheduler:**
  - Purpose: Schedule a job to clean old club meeting data automatically.
  - Key Functions: Executes scheduled jobs daily to trigger batch cleanup.

```
// Scheduler class
global class DailyMeetingCleanupScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        DailyMeetingCleanupBatch batch = new DailyMeetingCleanupBatch();
        Database.executeBatch(batch, 50);  // batch size 50
    }
}
```

3. Batch Apex:

- **DailyMeetingCleanupBatch:**
  - Purpose: Delete expired or past club meetings in bulk.
  - Key Functions: Query old records, delete in batches for performance.

```
// Batch class
global class DailyMeetingCleanupBatch implements Database.Batchable<SObject>,
Database.Stateful {

    global Database.QueryLocator start(Database.BatchableContext bc) {
        Date yesterday = Date.today().addDays(-1);
        return Database.getQueryLocator([
            SELECT Id, Name, CreatedBy.Email, CreatedBy.Name
            FROM Club_Meeting__c
            WHERE Meeting_Date__c = :yesterday
        ]);
```

```
    }

    global void execute(Database.BatchableContext bc, List<Club_Meeting__c>
meetings) {
        for (Club_Meeting__c meeting : meetings) {

            // Get attendance for this meeting
            List<Attendance__c> attendanceList = [
                SELECT Id, Student__r.Name, Student_Email__c
                FROM Attendance__c
                WHERE Club_Meeting__c = :meeting.Id
            ];

            if (!attendanceList.isEmpty()) {

                // Build HTML table
                String htmlTable = '<table border="1" style="border-collapse:collapse;">' +
                                   '<tr><th>Student Name</th><th>Email</th></tr>';
                for (Attendance__c att : attendanceList) {
                    String studentName = (att.Student__r != null) ? att.Student__r.Name :
'Unknown';
                    String studentEmail = (att.Student_Email__c != null) ?
att.Student_Email__c : 'N/A';
                    htmlTable += '<tr><td>' + studentName + '</td><td>' + studentEmail +
'</td></tr>';
                }
                htmlTable += '</table>';

                // Send email to meeting creator
                Messaging.SingleEmailMessage mail = new
Messaging.SingleEmailMessage();
                mail.setToAddresses(new String[] { meeting.CreatedBy.Email });
                mail.setSubject('Attendance Report for ' + meeting.Name);
                mail.setHtmlBody('Dear ' + meeting.CreatedBy.Name + ',<br/><br/>' +
                                 'Here is the attendance for your meeting: ' + meeting.Name +
'<br/><br/>' +
                                 htmlTable +
                                 '<br/><br/>Regards,<br/>System');

                Messaging.sendEmail(new Messaging.SingleEmailMessage[] { mail });

                // Delete attendance
                delete attendanceList;
            }
```

```
        }
    }

    global void finish(Database.BatchableContext bc) {
        // Optional logging
    }
}
```

4. Apex Triggers:

- **AttendanceTrigger:**
  - **Purpose:** Auto-create attendance records when a student is scanned.
  - **Business Hours Logic:** Ensures attendance is only recorded if the scan happens within the **defined Club Hours**
  - **Key Logic:**
    - On insert → check current time vs. club hours.
    - If outside hours → throw error or block attendance.

```
trigger AttendanceTrigger on Attendance__c (before insert, before update) {
    // Query Club Business Hours once
    BusinessHours clubHours = [
        SELECT Id
        FROM BusinessHours
        WHERE Name = 'Club Hours'
        LIMIT 1
    ];

    for (Attendance__c att : Trigger.new) {
        // Skip if no check-in time provided
        if (att.Check_In_Time__c == null) continue;

        // Check if inside Club business hours
        Boolean isValid = BusinessHours.isWithin(
            clubHours.Id,
            att.Check_In_Time__c
        );

        if (!isValid) {
            att.Check_In_Time__c.addError(
                'Attendance not allowed outside Club timings (10 AM – 3 PM, Mon–Sat,
excluding holidays).'
            );
        }
    }
```

```
}
```

- **SportsSaleTrigger:**
  - **Purpose:** Manage stock updates when an item is sold.
  - **Business Hours Logic:** Ensures sales can only be recorded during defined Store Hours.
  - **Key Logic:** On insert → check sale time vs. store hours.

```
trigger SportsSaleTrigger on Sports_Sale__c (before insert, before update) {
 // Query Store Business Hours once
 BusinessHours storeHours = [
    SELECT Id
    FROM BusinessHours
    WHERE Name = 'Store Hours'
    LIMIT 1
 ];

 for (Sports_Sale__c sale : Trigger.new) {
    // Skip if no sale date provided
    if (sale.Sale_Date__c == null) continue;

    // Check if inside Store business hours
    Boolean isValid = BusinessHours.isWithin(
       storeHours.Id,
       sale.Sale_Date__c
    );

    if (!isValid) {
       sale.Sale_Date__c.addError(
          'Store purchase not allowed outside Store timings (10 AM – 7 PM, Mon–Sat, excluding Sundays/holidays).'
       );
    }
 }
}
```

5. Test classes:

- **AttendanceTriggerTest**

```
@isTest
public class AttendanceTriggerTest {

   @testSetup
```

```
    static void setupData() {
        // Create a sample student
        Student__c s = new Student__c(Name = 'Test Student');
        insert s;

        // Create a sample Club Meeting
        Club_Meeting__c m = new Club_Meeting__c(Name = 'Weekly Meet');
        insert m;
    }

    @isTest
    static void testValidAttendance() {
        // Get IDs
        Student__c s = [SELECT Id FROM Student__c LIMIT 1];
        Club_Meeting__c m = [SELECT Id FROM Club_Meeting__c LIMIT 1];

        // Valid Check-In (weekday, within 10 AM – 3 PM)
        Attendance__c att = new Attendance__c(
            Name = 'Valid Attendance',
            Student__c = s.Id,
            Club_Meeting__c = m.Id,
            Check_In_Time__c = Datetime.newInstance(2025, 9, 18, 11, 0, 0) // Thu 11 AM
        );
        Test.startTest();
        insert att;
        Test.stopTest();

        System.assertNotEquals(null, att.Id, 'Attendance should be saved inside business
hours');
    }

    @isTest
    static void testInvalidAttendance() {
        Student__c s = [SELECT Id FROM Student__c LIMIT 1];
        Club_Meeting__c m = [SELECT Id FROM Club_Meeting__c LIMIT 1];

        // Invalid Check-In (weekday, but 5 PM = outside Club Hours)
        Attendance__c att = new Attendance__c(
            Name = 'Invalid Attendance',
            Student__c = s.Id,
            Club_Meeting__c = m.Id,
            Check_In_Time__c = Datetime.newInstance(2025, 9, 18, 17, 0, 0) // Thu 5 PM
        );
```

```
        Test.startTest();
        try {
            insert att;
            System.assert(false, 'Expected error for attendance outside hours');
        } catch (DmlException e) {
            System.assert(e.getMessage().contains('Attendance not allowed'), 'Error should
mention attendance hours');
        }
        Test.stopTest();
    }
}
```

- **SportsSaleTriggerTest:**

```
@isTest
public class SportsSaleTriggerTest {

    @testSetup
    static void setupData() {
        // Create a sample Sports Item
        Sports_Item__c item = new Sports_Item__c(
            Name = 'Football',
            Barcode__c = 'FB123',
            Price__c = 500,
            Stock_Quantity__c = 10
        );
        insert item;
    }

    @isTest
    static void testValidSale() {
        Sports_Item__c item = [SELECT Id, Price__c FROM Sports_Item__c LIMIT 1];

        Sports_Sale__c sale = new Sports_Sale__c(
            Name = 'Valid Sale',
            Sale_Date__c = Datetime.newInstance(2025, 9, 20, 18, 0, 0), // Sat 6 PM
            Total_Amount__c = item.Price__c,
            Items_Summary__c = 'Football x1'
        );

        Test.startTest();
        insert sale;
```

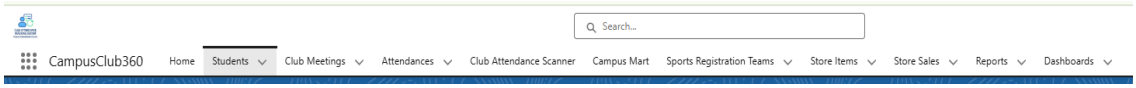
```
        Test.stopTest();

        System.assertNotEquals(null, sale.Id, 'Sale should be saved inside business
hours');
    }

    @isTest
    static void testInvalidSale() {
        Sports_Item__c item = [SELECT Id, Price__c FROM Sports_Item__c LIMIT 1];

        Sports_Sale__c sale = new Sports_Sale__c(
            Name = 'Invalid Sale',
            Sale_Date__c = Datetime.newInstance(2025, 9, 21, 11, 0, 0), // Sunday 11 AM
            Total_Amount__c = item.Price__c,
            Items_Summary__c = 'Football x1'
        );

        Test.startTest();
        try {
            insert sale;
            System.assert(false, 'Expected error for sale outside hours');
        } catch (DmlException e) {
            System.assert(e.getMessage().contains('Store purchase not allowed'), 'Error
should mention store hours');
        }
        Test.stopTest();
    }
}
```

Phase 6: User Interface Development

1. Custom App

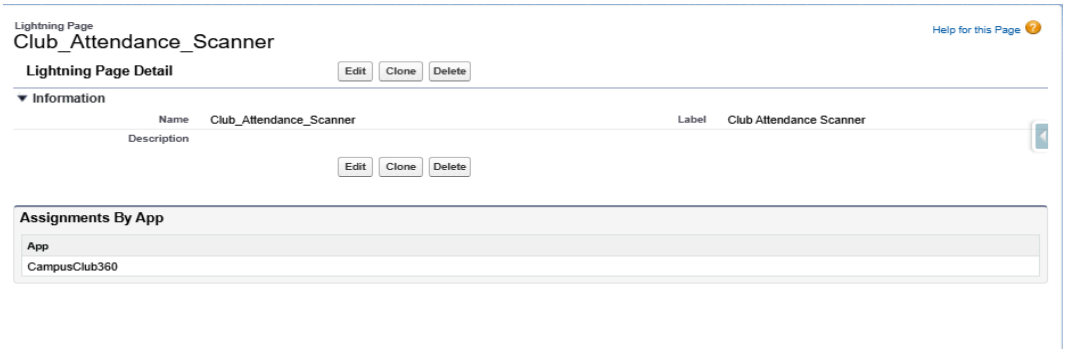
- **CampusClub360** → Central Salesforce app for managing students, clubs, sports, and store operations.
- **Tabs Included:**
  - Students
  - Club Meetings
  - Attendance
  - Club Attendance Scanner
  - Campus Mart
  - Sports Registration Teams
  - Store Items
  - Store Sales

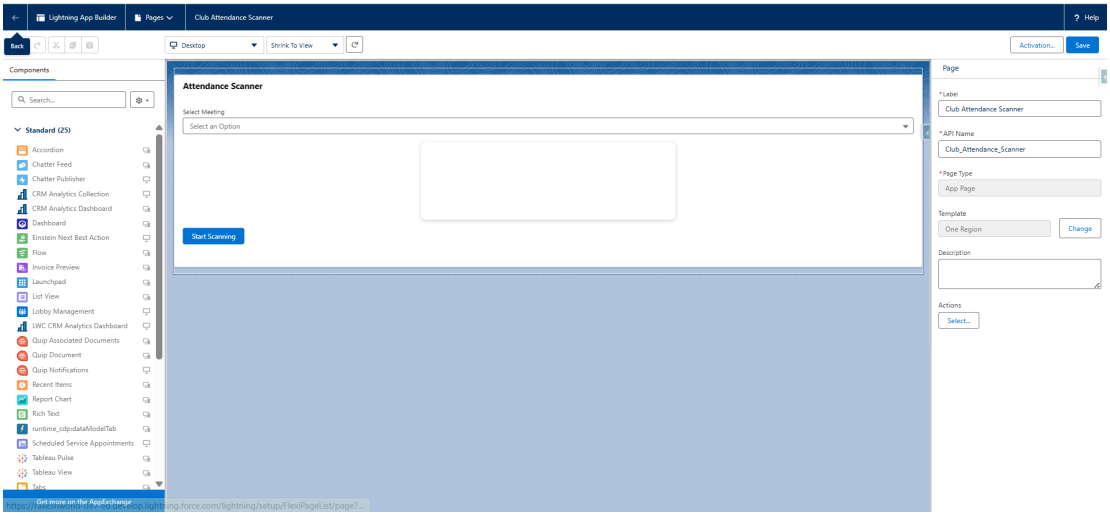
- Reports
- Dashboards



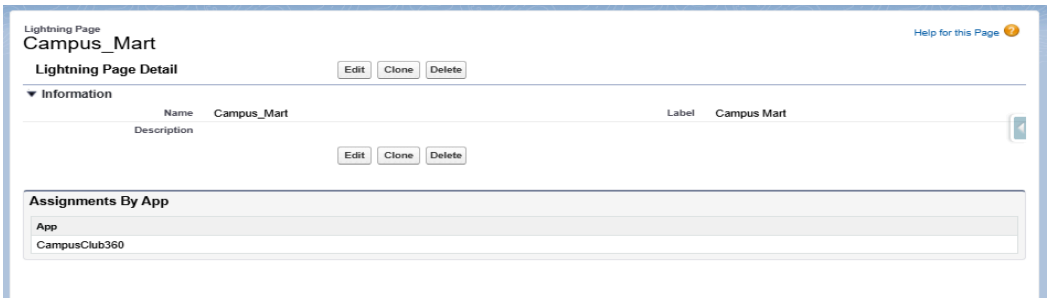
2. Lightning App Builder Pages

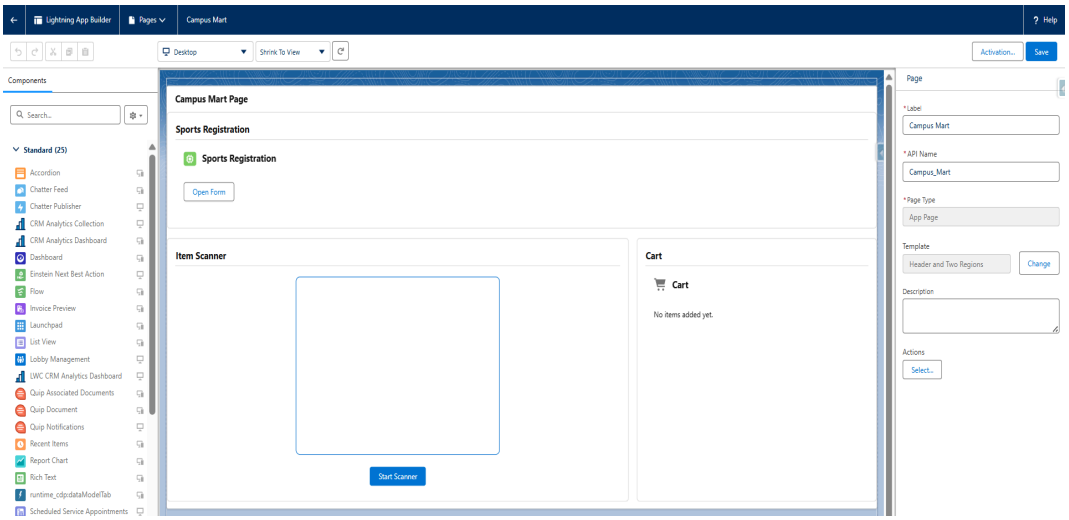
- **Club Attendance Scanner Page:**
  - **Purpose:** Allow faculty/club leaders to scan students' QR codes for attendance.
  - **Contains:** Custom LWC `attendanceScanner`.





- **Campus Mart Page:**
  - **Purpose**: Enable store salespersons to register sports teams and manage store sales.
  - **Layout:**
    - Top: Sports Registration Form (via LWC).
    - Left: Item Scanner (via LWC).
    - Right: Cart (store sales integration).





3. Apex with LWC:

- **attendanceScanner:**
  - **Purpose:** Scans QR code of students for attendance.
  - **Logic:**
    - Prevents duplicate scan entries.
    - Stores attendance record in `Attendance__c`.
  - **Backend:** Integrated with `AttendanceController`.

---

attendanceScanner.html

```
<template>
    <lightning-card title="Attendance Scanner">
        <div class="slds-p-around_medium">

            <!-- Current meeting display -->
            <template if:true={selectedMeetingName}>
                <p class="meetingInfo"><strong>Current Meeting:</strong>
{selectedMeetingName}</p>
            </template>
```

```
            <!-- Meeting selection -->
            <lightning-combobox
               name="meeting"
               label="Select Meeting"
               options={meetingOptions}
               value={selectedMeeting}
               onchange={handleMeetingChange}>
            </lightning-combobox>

            <!-- Video feed -->
            <div class="slds-m-top_medium scannerContainer">
               <video class="scanner-video" autoplay playsinline></video>
               <canvas class="scanner-canvas" style="display:none"></canvas>
            </div>

            <!-- Start/Stop button -->
            <div class="slds-m-top_medium">
               <lightning-button
                  variant="brand"
                  label={buttonLabel}
                  onclick={toggleScan}>
               </lightning-button>
            </div>

            <!-- Scanned results -->
            <div class="slds-m-top_medium results">
               <template if:true={scannedList.length}>
                  <h3>Recent Scans</h3>
                  <template for:each={scannedList} for:item="s">
                     <p key={s.key} class={s.cssClass}>
                        {s.name} — {s.time}
                     </p>
                  </template>
               </template>
            </div>

         </div>
      </lightning-card>
   </template>
```

attendanceScanner.css

```
.success {
   color: #0a8f08; /* green */
```

```
    font-weight: 600;
}
.duplicate {
    color: #d9822b; /* orange */
    font-weight: 600;
}
.error {
    color: #c23934; /* red */
    font-weight: 600;
}
.meetingInfo {
    margin-bottom: 0.5rem;
}
.scannerContainer {
    display: flex;
    justify-content: center;
    align-items: center;
}
.scanner-video {
    width: 100%;
    max-width: 480px;
    border-radius: 8px;
    box-shadow: 0 2px 6px rgba(0,0,0,0.15);
}
```

attendanceScanner.js

```
import { LightningElement, track } from 'lwc';
import { loadScript } from 'lightning/platformResourceLoader';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';

import JSQR from '@salesforce/resourceUrl/JSQR';
import getMeetings from '@salesforce/apex/AttendanceController.getMeetings';
import findStudentByBarcode from
'@salesforce/apex/AttendanceController.findStudentByBarcode';
import createAttendance from
'@salesforce/apex/AttendanceController.createAttendance';

export default class AttendanceScanner extends LightningElement {
    @track meetingOptions = [];
    selectedMeeting = '';
    selectedMeetingName = '';
    isScanning = false;
    jsqrLoaded = false;
```

```
    stream = null;
    @track scannedList = [];

    get buttonLabel() {
        return this.isScanning ? 'Stop Scanning' : 'Start Scanning';
    }

    renderedCallback() {
        if (this.jsqrLoaded) return;

        loadScript(this, JSQR)
            .then(() => {
                this.jsqrLoaded = true;
                return getMeetings();
            })
            .then(result => {
                // Map meetings; label shows name (you can add date if desired)
                this.meetingOptions = result.map(m => ({ label: m.Name, value: m.Id }));
            })
            .catch(error => {
                console.error('Error loading JSQR or fetching meetings', error);
                this.showToast('Load Error', 'Could not load scanner or meetings', 'error');
            });
    }

    handleMeetingChange(event) {
        this.selectedMeeting = event.detail.value;
        const meeting = this.meetingOptions.find(m => m.value === this.selectedMeeting);
        this.selectedMeetingName = meeting ? meeting.label : '';
    }

    async toggleScan() {
        if (!this.isScanning) {
            if (!this.selectedMeeting) {
                this.showToast('Select Meeting', 'Please select a meeting first.', 'warning');
                return;
            }
            await this.startCamera();
            this.isScanning = true;
            this.scanLoop();
        } else {
            this.stopCamera();
            this.isScanning = false;
```

```
        }
    }

    async startCamera() {
        const video = this.template.querySelector('video');
        try {
            this.stream = await navigator.mediaDevices.getUserMedia({ video: {
facingMode: 'environment' } });
            video.srcObject = this.stream;
            await video.play();
        } catch (err) {
            console.error('Camera error', err);
            this.showToast('Camera Error', 'Cannot access camera. Allow permission or try
a different browser.', 'error');
        }
    }

    stopCamera() {
        if (this.stream) {
            this.stream.getTracks().forEach(t => t.stop());
            this.stream = null;
        }
    }

    disconnectedCallback() {
        // ensure camera is stopped when component removed
        this.stopCamera();
    }

    async scanLoop() {
        const video = this.template.querySelector('video');
        const canvas = this.template.querySelector('canvas');
        if (!canvas) return;
        const ctx = canvas.getContext('2d');

        const loop = async () => {
            if (!this.isScanning) return;
            if (video.readyState === video.HAVE_ENOUGH_DATA) {
                canvas.width = video.videoWidth;
                canvas.height = video.videoHeight;
                ctx.drawImage(video, 0, 0, canvas.width, canvas.height);
                const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);
```

```
            const code = window.jsQR ? window.jsQR(imageData.data, canvas.width,
canvas.height) : null;
            if (code && code.data) {
                await this.handleDetected(code.data);
                await this.sleep(1200); // debounce
            }
        }
        requestAnimationFrame(loop);
    };
    requestAnimationFrame(loop);
  }

  sleep(ms) {
    return new Promise(res => setTimeout(res, ms));
  }

  async handleDetected(barcodeValue) {
    try {
        const student = await findStudentByBarcode({ barcodeValue });
        if (!student) {
            this.showToast('Not Found', `No student found for code: ${barcodeValue}`,
'error');
            // show in recent list as error
            this.prependScan({ key: barcodeValue + '-' + Date.now(), name: `Unknown
(${barcodeValue})`, time: new Date().toLocaleString(), cssClass: 'error' });
            return;
        }

        // call server to create attendance (or return existing)
        const result = await createAttendance({ studentId: student.Id, meetingId:
this.selectedMeeting });
        const att = result.attendance;
        const wasNew = result.isNew;

        if (wasNew) {
            this.showToast('Attendance Recorded', `${att.Student__r ?
att.Student__r.Name : student.Name} checked in successfully.`, 'success');
        } else {
            this.showToast('Duplicate', `${att.Student__r ? att.Student__r.Name :
student.Name} is already checked in for this meeting.`, 'warning');
        }

        this.prependScan({
            key: att.Id,
```

```
                name: att.Student__r ? att.Student__r.Name : student.Name,
                time: new Date(att.Check_In_Time__c).toLocaleString(),
                cssClass: wasNew ? 'success' : 'duplicate'
            });

        } catch (err) {
            console.error('Error processing scan', err);
            this.showToast('Error', 'An error occurred while processing the scan (or) Attendance not allowed outside Club timings (10 AM – 3 PM, Mon–Sat, excluding holidays', 'error');
        }
    }

    prependScan(scanObj) {
        this.scannedList = [scanObj, ...this.scannedList].slice(0, 10);
    }

    showToast(title, message, variant='info') {
        this.dispatchEvent(new ShowToastEvent({ title, message, variant }));
    }
}
```

attendanceScanner.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>58.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
        <target>lightning__RecordPage</target>
    </targets>
</LightningComponentBundle>
```

- **sportsRegistrationForm:**
  - **Purpose:** Allows salespersons to register sports teams.
  - **Fields:** Team Name, Number of Players, Sport, Registration Fee.
  - **Logic:** On submit → record created in `Sports_Registration__c`.
  - **Backend:** Integrated with `SportsRegistrationController`.

---

sportsRegistrationForm.html

```
<template>
    <lightning-card title="Sports Registration" icon-name="custom:custom63">
```

```
        <div class="slds-p-around_medium">
            <!-- Toggle Button -->
            <lightning-button label={toggleLabel} onclick={toggleForm}></lightning-button>
        </div>

        <!-- Registration Form -->
        <template if:true={isFormOpen}>
            <div class="slds-p-around_medium">
                <lightning-combobox
                    name="sport"
                    label="Select Sport"
                    value={sport}
                    placeholder="Choose a sport"
                    options={sportOptions}
                    onchange={handleChange}>
                </lightning-combobox>

                <lightning-input
                    name="teamName"
                    label="Team Name"
                    value={teamName}
                    onchange={handleChange}>
                </lightning-input>

                <lightning-input
                    name="players"
                    label="Number of Players"
                    type="number"
                    value={players}
                    onchange={handleChange}>
                </lightning-input>

                <lightning-input
                    name="fee"
                    label="Registration Fee"
                    type="number"
                    value={fee}
                    onchange={handleChange}>
                </lightning-input>

                <div class="slds-m-top_medium">
                    <lightning-button variant="brand" label="Register"
onclick={handleRegister}></lightning-button>
```

```
            </div>
          </div>
        </template>
    </lightning-card>
</template>
```

sportsRegistrationForm.js

```
import { LightningElement, track } from 'lwc';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import createRegistration from
'@salesforce/apex/SportsRegistrationController.createRegistration';

export default class SportsRegistrationForm extends LightningElement {
    @track isFormOpen = false;
    @track sport = '';
    @track teamName = '';
    @track players = 0;
    @track fee = 0;

    get toggleLabel() {
        return this.isFormOpen ? 'Close Form' : 'Open Form';
    }

    get sportOptions() {
        return [
            { label: 'Cricket', value: 'Cricket' },
            { label: 'Kabaddi', value: 'Kabaddi' },
            { label: 'Volley Ball', value: 'Volley Ball' },
            { label: 'Throw Ball', value: 'Throw Ball' },
            { label: 'Basket Ball', value: 'Basket Ball' },
            { label: 'Tennikoit', value: 'Tennikoit' },
            { label: 'Badminton', value: 'Badminton' },
            { label: 'Kho-Kho', value: 'Kho-Kho' },
            { label: 'Chess', value: 'Chess' },
            { label: 'Tug of War', value: 'Tug of War' },
            { label: 'Carroms', value: 'Carroms' },
            { label: 'Table-Tennis', value: 'Table-Tennis' },
            { label: 'Others', value: 'Others' }
        ];
    }

    toggleForm() {
        this.isFormOpen = !this.isFormOpen;
```

```
    }

    handleChange(event) {
        this[event.target.name] = event.target.value;
    }

    handleRegister() {
        createRegistration({
            sport: this.sport,
            teamName: this.teamName,
            players: this.players,
            fee: this.fee
        })
            .then(() => {
                this.dispatchEvent(
                    new ShowToastEvent({
                        title: 'Success',
                        message: 'Team Registered Successfully!',
                        variant: 'success'
                    })
                );
                this.resetForm();
            })
            .catch(error => {
                this.dispatchEvent(
                    new ShowToastEvent({
                        title: 'Error',
                        message: error.body.message,
                        variant: 'error'
                    })
                );
            });
    }

    resetForm() {
        this.sport = '';
        this.teamName = '';
        this.players = 0;
        this.fee = 0;
    }
}
```

sportsRegistrationForm.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>64.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

- **sportsItemScanner (child component):**
  - **Purpose:** Scans barcodes of store items**.**
  - **Logic:**
    - Displays scanned item.
    - On "Add-To-Cart" → item moved to `SportsItemList`.
  - **Backend:** Integrated with `SportsItemController`.

---

sportsItemScanner.html

```
<template>
    <div class="scanner-container">
        <div class="video-box">
            <video></video>
        </div>

        <div class="scanner-footer">
            <lightning-button
                label={buttonLabel}
                onclick={toggleScanner}
                variant="brand">
            </lightning-button>
        </div>

        <!-- Show result -->
        <template if:true={scannedItem}>
            <div class="result-success">
                <p>{scannedItem.Name}</p>
                <p>Price: ₹{scannedItem.Price__c}</p>
                <lightning-button
                    label="Add to Cart"
                    onclick={addToCart}
                    variant="success">
```

```
                </lightning-button>
            </div>
        </template>

        <template if:true={errorMsg}>
            <div class="result-error">{errorMsg}</div>
        </template>
    </div>
</template>
```

sportsItemScanner.css

```
.scanner-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    padding: 10px;
}

.video-box {
    width: 100%;
    max-width: 400px;
    height: 300px;
    border: 2px solid #0070d2; /* Salesforce blue */
    border-radius: 8px;
    overflow: hidden;
    margin-bottom: 10px;
}

video {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

.scanner-footer {
    margin-top: 10px;
    text-align: center;
}

.result-success {
    color: green;
    font-weight: bold;
```

```
    margin-top: 10px;
}

.result-error {
    color: red;
    font-weight: bold;
    margin-top: 10px;
}
```

sportsItemScanner.js

```
import { LightningElement, track } from 'lwc';
import { loadScript } from 'lightning/platformResourceLoader';
import ZXingLib from '@salesforce/resourceUrl/ZXing';
import getItemByBarcode from
'@salesforce/apex/SportsItemController.getItemByBarcode';

export default class SportsItemScanner extends LightningElement {
    @track scannedItem;   // Store scanned item details
    @track errorMsg;
    scannerRunning = false;
    zxingLoaded = false;
    codeReader;

    renderedCallback() {
        if (this.zxingLoaded) return;

        loadScript(this, ZXingLib)
            .then(() => {
                this.zxingLoaded = true;
                this.codeReader = new window.ZXing.BrowserBarcodeReader();
                console.log('✅ ZXing loaded successfully');
            })
            .catch(error => {
                this.errorMsg = 'Failed to load ZXing: ' + error;
                console.error(this.errorMsg);
            });
    }

    async startScanner() {
        try {
            const video = this.template.querySelector('video');
            const result = await this.codeReader.decodeOnceFromVideoDevice(undefined,
video);
```

```
            // Call Apex to get item details
            const item = await getItemByBarcode({ barcode: result.text });
            if (item) {
                this.scannedItem = item;
                this.errorMsg = null;
            } else {
                this.scannedItem = null;
                this.errorMsg = '⚠️ Item not found in system';
            }

            this.stopScanner();
        } catch (err) {
            this.errorMsg = '⚠️ Scan failed: ' + err.message;
            console.error(err);
        }
    }

    stopScanner() {
        if (this.codeReader) {
            this.codeReader.reset();
        }
        this.scannerRunning = false;
    }

    toggleScanner() {
        if (this.scannerRunning) {
            this.stopScanner();
        } else {
            this.startScanner();
            this.scannerRunning = true;
        }
    }

    get buttonLabel() {
        return this.scannerRunning ? 'Stop Scanner' : 'Start Scanner';
    }

    // Add item to cart
    // sportsItemScanner.js
    addToCart() {
    if (this.scannedItem) {
        this.dispatchEvent(new CustomEvent('addtocart', {
```

```
            detail: {
                id: this.scannedItem.Id,   //  very important
                name: this.scannedItem.Name,
                price: Number(this.scannedItem.Price__c),
                barcode: this.scannedItem.Barcode__c,
                quantity: 1
            }
        }));
        this.scannedItem = null;
    }
    }

}
```

sportsItemScanner.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>64.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

- **sportsItemList (child component):**
  - **Purpose:** Displays all items added from the scanner.
  - **Logic:**
  - **Backend:** Integrated with `SportsSaleController`.

---

sportsItemList.html

```
<template>
    <lightning-card title="Cart" icon-name="utility:cart">
        <template if:true={cartItems.length}>
            <template for:each={cartItems} for:item="item">
                <div key={item.id} class="cart-item">
                    <span class="item-name">{item.name}</span>
                    <span class="item-price">
                        ₹{item.price} × {item.quantity} = ₹{item.lineTotal}
                    </span>
                    <lightning-button-icon
                        icon-name="utility:delete"
```

```
                        variant="destructive"
                        alternative-text="Remove"
                        title="Remove"
                        onclick={removeItem}
                        data-id={item.id}>
                    </lightning-button-icon>
                </div>
            </template>
            <div class="total">Total: ₹{totalAmount}</div>
            <lightning-button
                label="Done"
                variant="brand"
                onclick={finalizeSale}>
            </lightning-button>
        </template>
        <template if:false={cartItems.length}>
            <p class="slds-p-around_medium">No items added yet.</p>
        </template>

        <!--Custom Toast inside component -->
        <template if:true={toast.visible}>
            <div class={toastClass}>
                {toast.message}
            </div>
        </template>
    </lightning-card>
</template>
```

sportsItemList.css

```
.cart-item {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin: 0.3rem 0;
}

.item-name {
    font-weight: bold;
}

.item-price {
    margin-left: 1rem;
    flex-grow: 1;
```

```
}

.total {
    font-weight: bold;
    margin-top: 1rem;
    text-align: right;
}

/*Custom Toast Styles */
.success {
    background: #2ecc71;
    color: white;
    padding: 0.5rem;
    margin-top: 1rem;
    border-radius: 0.25rem;
}

.error {
    background: #e74c3c;
    color: white;
    padding: 0.5rem;
    margin-top: 1rem;
    border-radius: 0.25rem;
}
```

sportsItemList.js

```
import { LightningElement, track, api } from 'lwc';
import finalizeSale from '@salesforce/apex/SportsSaleController.finalizeSale';

export default class SportsItemList extends LightningElement {
    @track cartItems = [];
    @track totalAmount = 0;

    // toast state
    @track toast = { visible: false, message: '', type: '' };

    showToast(message, type = 'info') {
        this.toast = { visible: true, message, type };

        // auto hide after 3 sec
        setTimeout(() => {
            this.toast.visible = false;
        }, 3000);
```

```
}

// Add item
@api addItem(item) {
    let existing = this.cartItems.find(i => i.id === item.id);
    if (existing) {
        existing.quantity += 1;
        existing.lineTotal = existing.price * existing.quantity;
    } else {
        this.cartItems = [
            ...this.cartItems,
            { ...item, quantity: 1, lineTotal: item.price }
        ];
    }
    this.calculateTotal();
}

removeItem(event) {
    const id = event.currentTarget.dataset.id;
    let index = this.cartItems.findIndex(i => i.id === id);
    if (index > -1) {
        if (this.cartItems[index].quantity > 1) {
            this.cartItems[index].quantity -= 1;
            this.cartItems[index].lineTotal =
                this.cartItems[index].price * this.cartItems[index].quantity;
        } else {
            this.cartItems.splice(index, 1);
        }
        this.cartItems = [...this.cartItems];
    }
    this.calculateTotal();
}

calculateTotal() {
    this.totalAmount = this.cartItems.reduce((sum, i) => sum + i.lineTotal, 0);
}

finalizeSale() {
    finalizeSale({
        itemsJson: JSON.stringify(this.cartItems),
        totalAmount: this.totalAmount
    })
        .then(() => {
```

```
                this.showToast('✅ Sale finalized successfully!', 'success');
                this.cartItems = [];
                this.totalAmount = 0;
            })
            .catch((err) => {
                let message = 'Unknown error';
                if (err && err.body && err.body.message) {
                    message = err.body.message;
                } else if (err && err.message) {
                    message = err.message;
                }
                this.showToast('❌ Failed to finalize sale: ' + message, 'error');
            });
    }
}
```

sportsItemList.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>64.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

---

4. LWC (Lightning Web Components):

- **sportsClubPage (parent component):**
  - **Purpose:** Acts as the main container that connects:
    - **Top:** Sports Registration Form
    - **Left:** Sports Item Scanner
    - **Right:** Sports Item List
  - **Logic:** Coordinates data flow → scanner → list → sales record.

---

sportsClubPage.html

```
<template>
    <lightning-card title="Campus Mart Page">
        <div class="sports-club-container">

            <!-- Top: Sports Registration -->
            <div class="section registration">
```

```
            <h2>Sports Registration</h2>
            <c-sports-registration-form></c-sports-registration-form>
          </div>

          <!-- Bottom split: Scanner and Cart -->
          <div class="content">
            <div class="section scanner">
              <h2>Item Scanner</h2>
              <c-sports-item-scanner
                onaddtocart={handleAddToCart}>
              </c-sports-item-scanner>
            </div>

            <div class="section cart">
              <h2>Cart</h2>
              <c-sports-item-list
                onfinalize={handleFinalize}>
              </c-sports-item-list>
            </div>
          </div>
        </div>
    </lightning-card>
</template>
```

sportsClubPage.css

```
.sports-club-container {
    display: flex;
    flex-direction: column;
    gap: 1rem;
}

.section {
    border: 1px solid #dcdcdc;  /* light grey border */
    border-radius: 6px;
    padding: 1rem;
    background: #ffffff;
}

.section h2 {
    font-size: 1rem;
    font-weight: bold;
    margin-bottom: 0.5rem;
    border-bottom: 1px solid #ddd;
```

```
    padding-bottom: 0.3rem;
}

.content {
    display: flex;
    gap: 1rem;
}

.scanner {
    flex: 2;
}

.cart {
    flex: 1;
}
```

sportsClubPage.js

```
import { LightningElement } from 'lwc';

export default class SportsClubPage extends LightningElement {

    // Handle add-to-cart from scanner
    handleAddToCart(event) {
        const cart = this.template.querySelector('c-sports-item-list');
        if (cart) {
            cart.addItem(event.detail);
        }
    }
}
```

sportsClubPage.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>64.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

---

5. Student QR Codes:

Student 1

Student 2

Student 3

Student 4

Student 5

Student 6

Student 7

Student 8

Student 9

Student 10

6.Sports Item Barcodes:

Cricket Ball 1

Cricket Ball 2

Cricket Bat 1

Cricket Bat 2

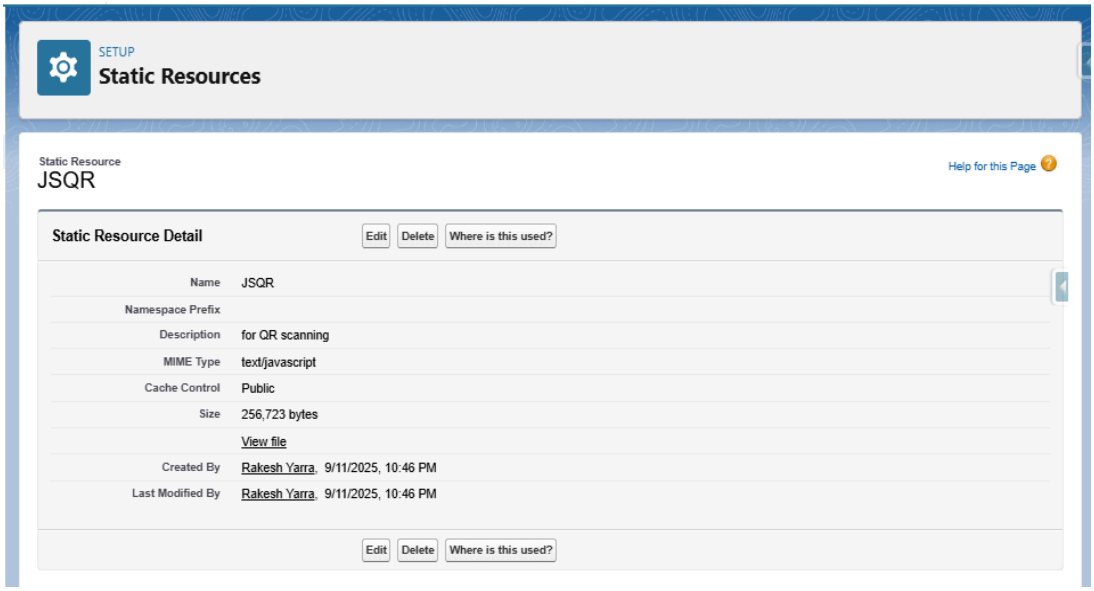
Chemistry Textbook

Backpack (College)

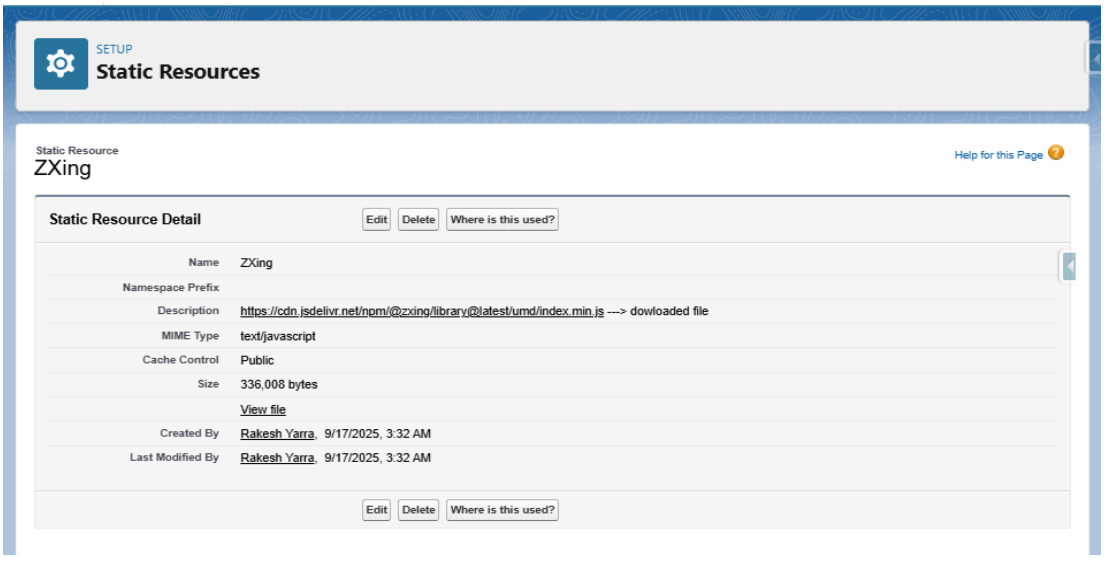
Phase 7: Integration & External Access

1. Static Resources:

- **JSQR → JavaScript library used for QR code scanning in `attendanceScanner` LWC.**



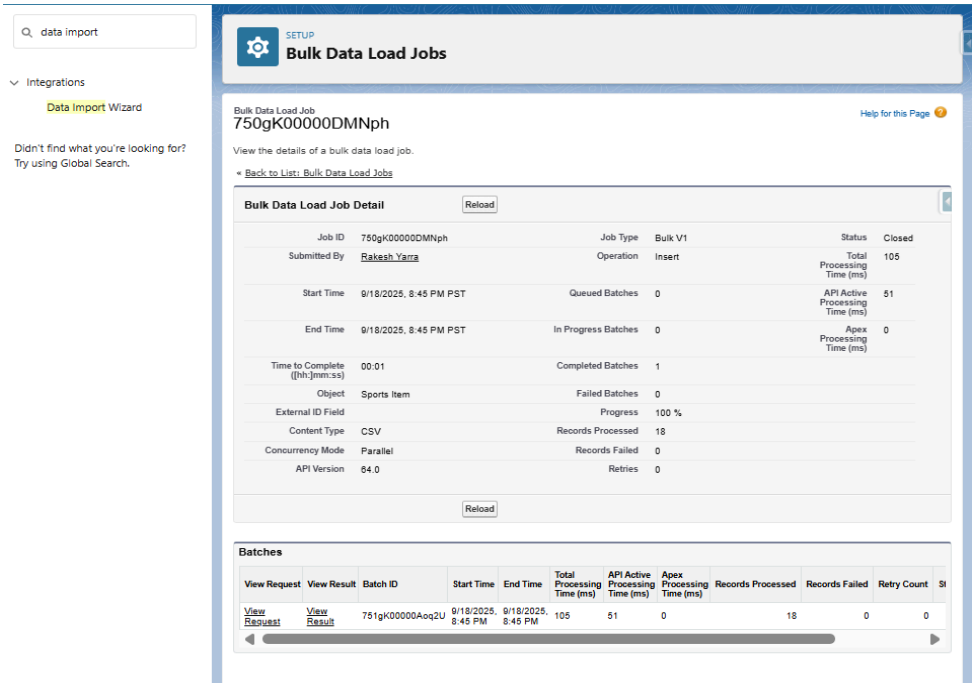
- **ZXing → Barcode scanner library used in `SportsItemScanner` for scanning store items.**



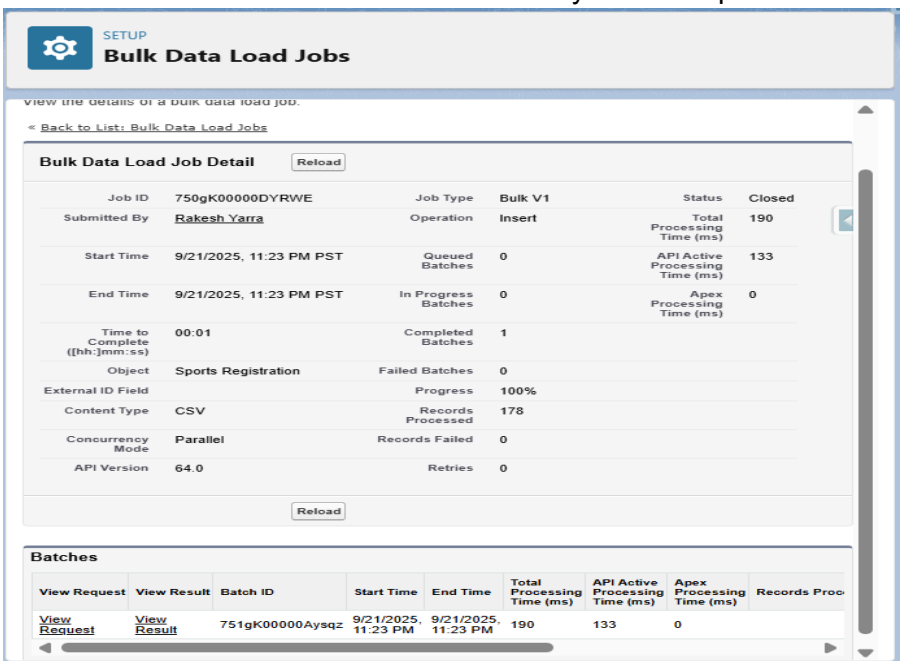
Phase 8: Data Management & Deployment:

1. Data Import Wizard

- Used to import initial data into Salesforce.
- **Imported two main datasets:**
  - **Sports Registration Data** → Pre-loaded student/team registrations into the Sports Registration table.



  - **Store Items Data** → Loaded all the inventory into the Sports Items table.



- **Purpose:** Ensure the system is ready with baseline data for sports teams and store sales before demo/use.

2. Development Tools & Workflow:

- Lightning Framework: Used only for Lightning Web Components (LWC).
- VS Code: Used for developing and deploying LWC (attendanceScanner, sportsRegistrationForm, sportsItemScanner, sportsItemList, sportsClubPage).
- Developer Console: Used for creating and testing Apex Classes and Triggers (AttendanceController, SportsSaleTrigger, etc.).
- **GitHub** → For **version control and backup**.

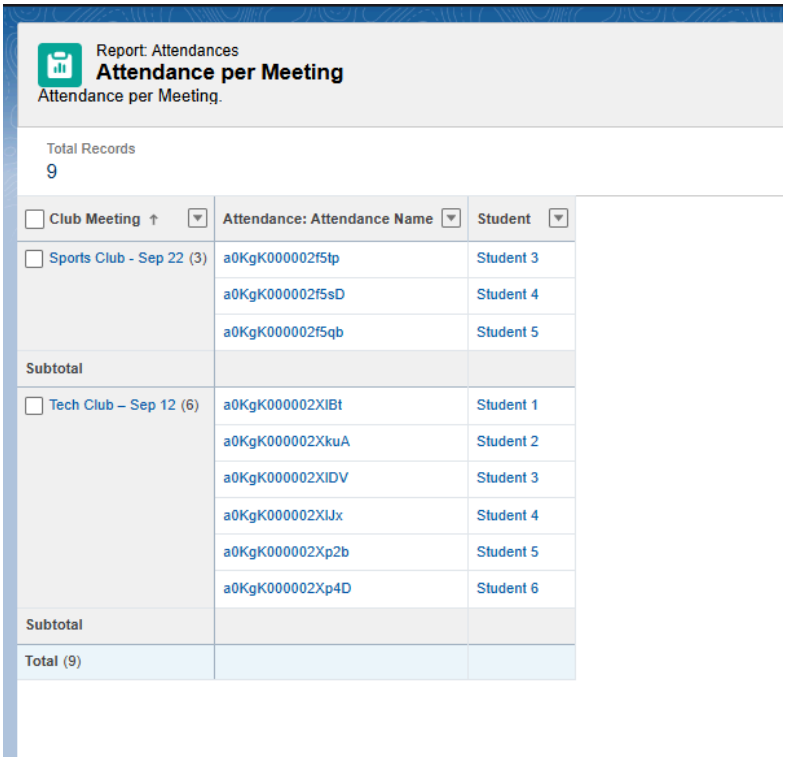
3. Deployment:

- Connected VS Code to Salesforce Org using SFDX authentication.
- Downloaded metadata (Objects, Flows, Apex Classes, Triggers, LWCs) from the Org into VS Code.
- Integrated VS Code with GitHub, pushing all project files into a repository.
- Used this GitHub repo as a source of truth for project artifacts.

Phase 9: Reporting, Dashboards & Security Review

1. Reports: I created custom report folders to organize insights for Attendance and Store Sales:

- **Attendance Reports:**
  - **Attendance per Meeting** → Shows student participation in each club meeting.

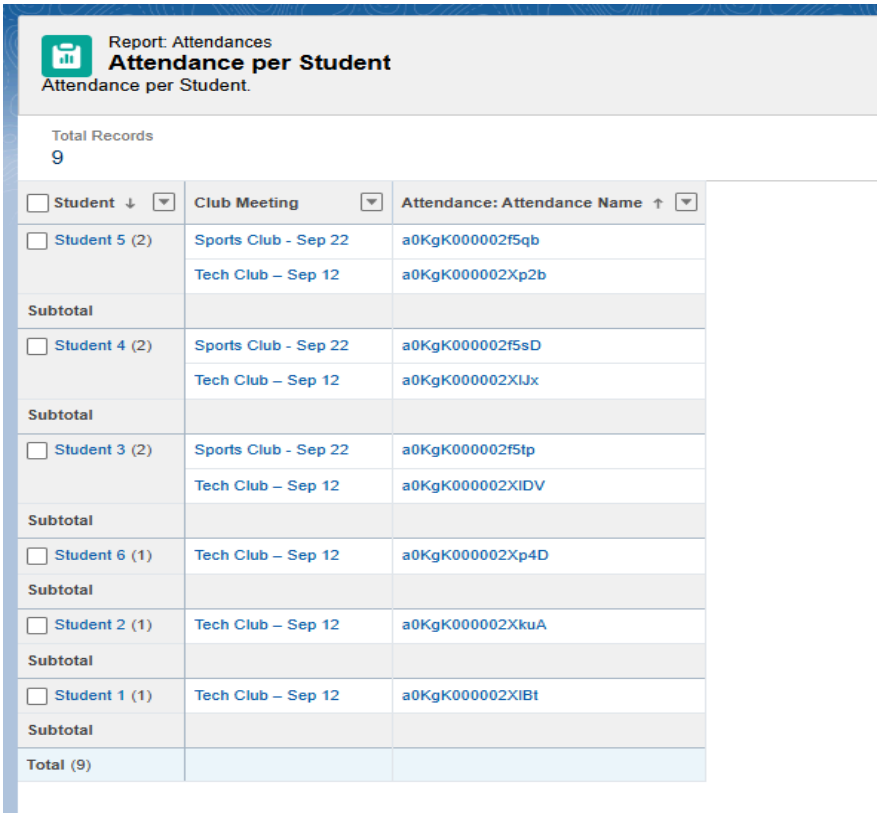


Report: Attendances
Attendance per Meeting
Attendance per Meeting.

Total Records
9

Club Meeting	Attendance: Attendance Name	Student
Sports Club - Sep 22 (3)	a0KgK000002f5tp	Student 3
	a0KgK000002f5sD	Student 4
	a0KgK000002f5qb	Student 5
Subtotal		
Tech Club – Sep 12 (6)	a0KgK000002XIBt	Student 1
	a0KgK000002XkuA	Student 2
	a0KgK000002XIDV	Student 3
	a0KgK000002XIJx	Student 4
	a0KgK000002Xp2b	Student 5
	a0KgK000002Xp4D	Student 6
Subtotal		
Total (9)		

- **Attendance per Student** → Tracks each student's overall club attendance.

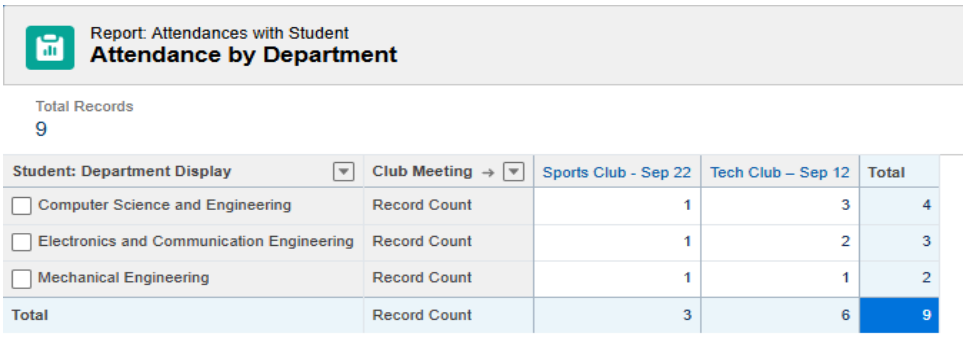


Report: Attendances
Attendance per Student
Attendance per Student.

Total Records
9

Student	Club Meeting	Attendance: Attendance Name
Student 5 (2)	Sports Club - Sep 22	a0KgK000002f5qb
	Tech Club – Sep 12	a0KgK000002Xp2b
Subtotal		
Student 4 (2)	Sports Club - Sep 22	a0KgK000002f5sD
	Tech Club – Sep 12	a0KgK000002XIJx
Subtotal		
Student 3 (2)	Sports Club - Sep 22	a0KgK000002f5tp
	Tech Club – Sep 12	a0KgK000002XIDV
Subtotal		
Student 6 (1)	Tech Club – Sep 12	a0KgK000002Xp4D
Subtotal		
Student 2 (1)	Tech Club – Sep 12	a0KgK000002XkuA
Subtotal		
Student 1 (1)	Tech Club – Sep 12	a0KgK000002XIBt
Subtotal		
Total (9)		

- **Attendance per Department** → Summarizes participation at the department level.



Report: Attendances with Student
Attendance by Department

Total Records
9

Student: Department Display	Club Meeting	Sports Club - Sep 22	Tech Club – Sep 12	Total
Computer Science and Engineering	Record Count	1	3	4
Electronics and Communication Engineering	Record Count	1	2	3
Mechanical Engineering	Record Count	1	1	2
Total	Record Count	3	6	9

- Students with Club Fee → Highlights students who paid the club fee.

Report: Students
Students with Club Fee

Total Records	Total Club Fee Paid
9	₹18,000

Department Display	Student: Student Name	Club Fee Paid
Chemical Engineering (1)	Student 10	₹2,000
Subtotal		₹2,000
Civil Engineering (1)	Student 7	₹2,000
Subtotal		₹2,000
Computer Science and Engineering (3)	Student 2	₹2,000
	Student 5	₹2,000
	Student 6	₹2,000
Subtotal		₹6,000
Data Science (1)	Student 9	₹2,000
Subtotal		₹2,000
Electronics and Communication Engineering (2)	Student 1	₹2,000
	Student 4	₹2,000
Subtotal		₹4,000
Mechanical Engineering (1)	Student 3	₹2,000
Subtotal		₹2,000
Total (9)		₹18,000

- **Store Reports:**

- Monthly Total Store Revenue → Displays monthly sales revenue from Campus Mart.

Report: Sports Sale History
Monthly Total Store Revenue

Total Records	Total Total Amount
8	₹7,410

Sale Date	Total Amount
September 2025 (8)	₹1,740
	₹2,080
	₹160
	₹150
	₹150
	₹1,580
	₹1,000
	₹550
Subtotal	₹7,410
Total (8)	₹7,410

- **Top Selling Items** → Identifies the most popular sports items.

Report: Store Sales with Store Items
Top Selling Items Report

Total Records: 8
Total Total Amount: ₹7,410

Sale Date	Store Item Name	Items Summary	Total Amount
9/18/2025 (7)	- (7)	Cricket Ball 2 (x1) - ₹80 Cricket Bat 2 (x1) - ₹2000	₹2,080
		Circket Ball (x2) - ₹240 Cricket Bat (x1) - ₹1500	₹1,740
		Cricket Bat (x1) - ₹1500 Cricket Ball 2 (x1) - ₹80	₹1,580
		Water Bottle (1L) (x1) - ₹250 Volleyball (x1) - ₹750	₹1,000
		Cricket Ball 2 (x2) - ₹160	₹160
		Cricket Ball 1 (x1) - ₹150	₹150
		Cricket Ball 1 (x1) - ₹150	₹150
	Subtotal		₹6,860
Subtotal			₹6,860
9/21/2025 (1)	- (1)	Water Bottle (1L) (x1) - ₹250 Shuttlecock Pack (6) (x1) - ₹300	₹550
	Subtotal		₹550
Subtotal			₹550
Total (8)			₹7,410

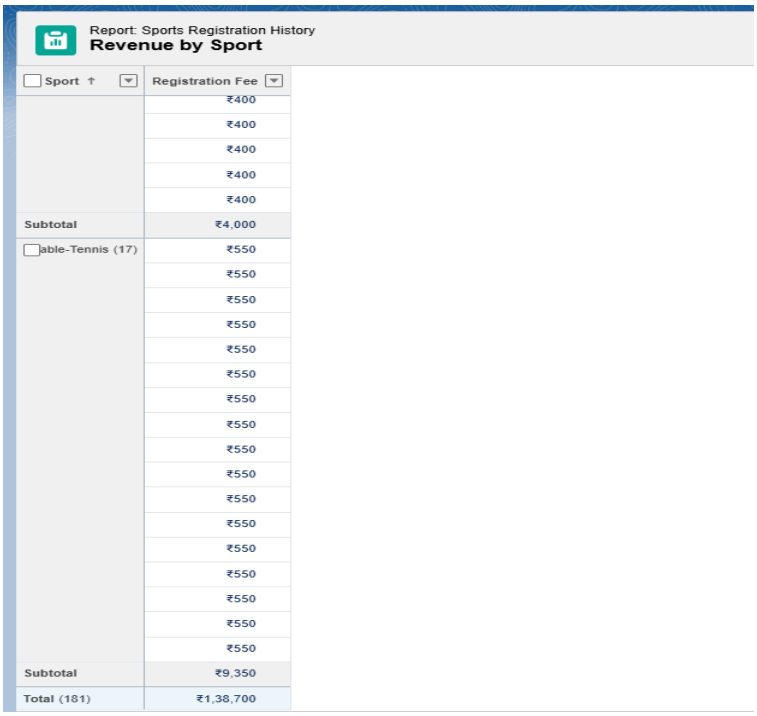
- **Number of Sales** → Counts transactions for analysis.

Report: Sports Sale History
Number of Sales Report

Total Records: 8

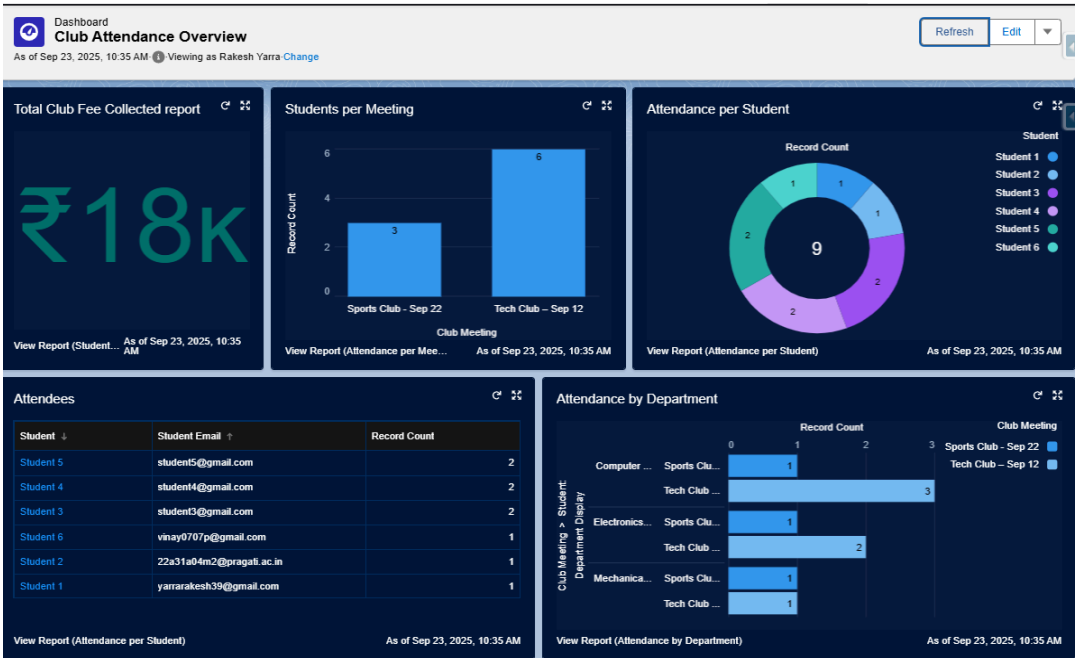
Sale Date	Sports Sale: Store Sale Name	Items Summary
September 2025 (8)	Sale - 2025-09-18 05:56	Circket Ball (x2) - ₹240 Cricket Bat (x1) - ₹1500
	Sale - 2025-09-18 06:47	Cricket Ball 2 (x1) - ₹80 Cricket Bat 2 (x1) - ₹2000
	Sale - 2025-09-18 07:17	Cricket Ball 2 (x2) - ₹160
	Sale - 2025-09-18 19:09	Cricket Ball 1 (x1) - ₹150
	Sale - 2025-09-18 19:11	Cricket Ball 1 (x1) - ₹150
	Sale - 2025-09-18 19:13	Cricket Bat (x1) - ₹1500 Cricket Ball 2 (x1) - ₹80
	Sale - 2025-09-18 21:39	Water Bottle (1L) (x1) - ₹250 Volleyball (x1) - ₹750
	Sale - 2025-09-21 21:35	Water Bottle (1L) (x1) - ₹250 Shuttlecock Pack (6) (x1) - ₹300
Subtotal		
Total (8)		

- **Revenue by Sport** → Groups revenue based on sport categories.

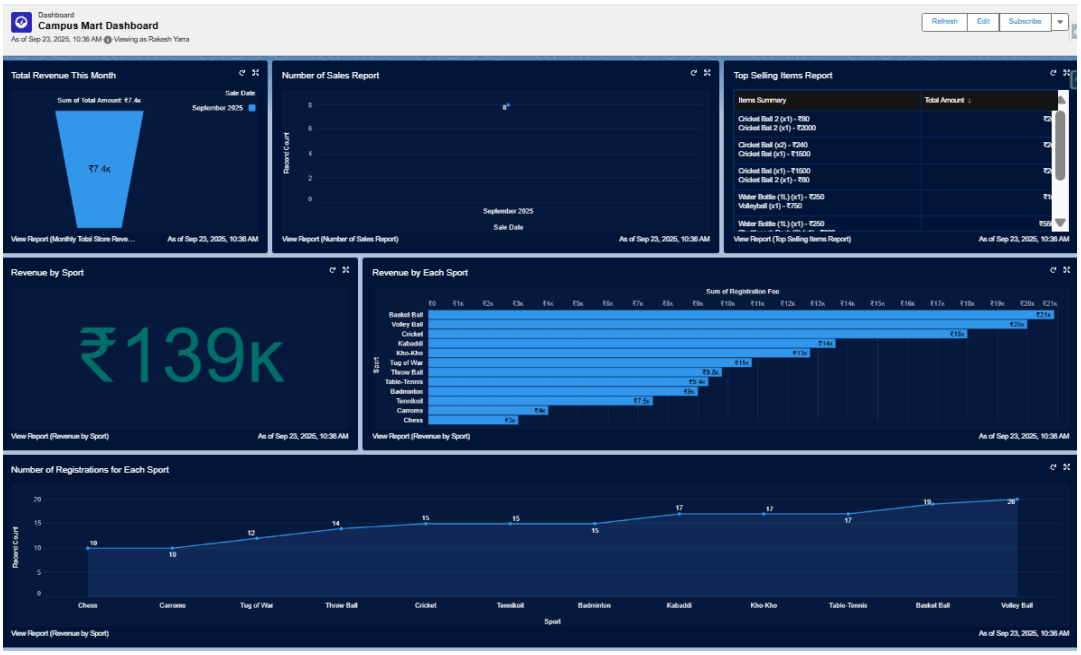


2. Dashboards: I designed dashboards for a visual overview

- **Club Attendance Overview Dashboard:**
  - **Key charts:** Attendance by Department, Attendance Trend by Month, Top 10 Active Students.

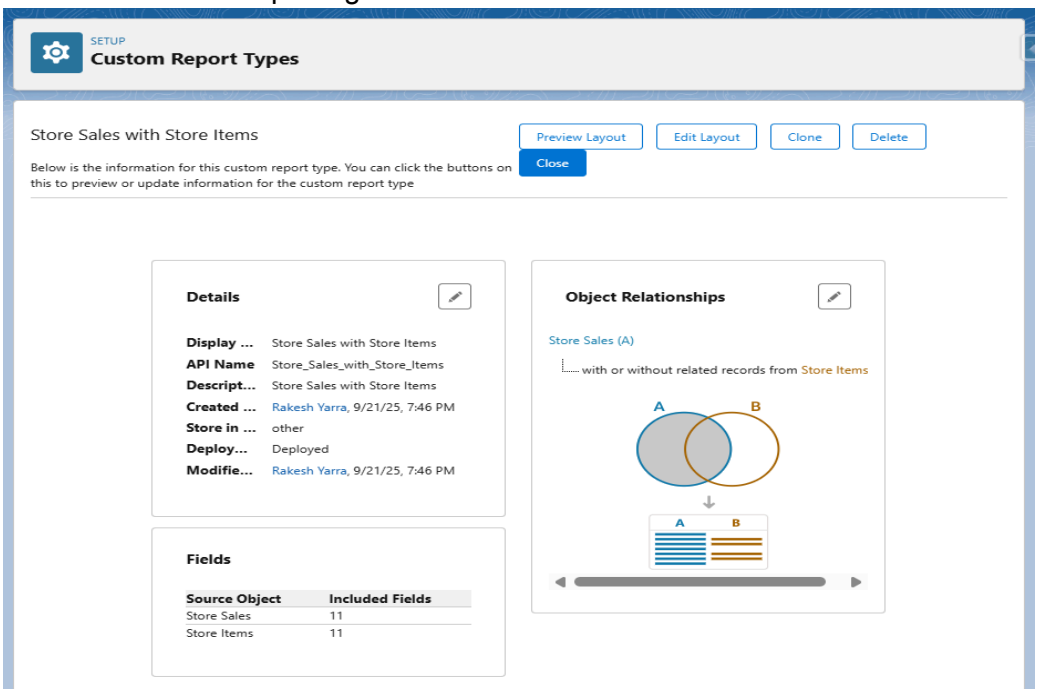


- **Campus Mart Dashboard:**
  - **Key charts:** Monthly Revenue, Top 5 Items Sold, Revenue Breakdown by Sport.



3. Report Types:

- **Store Sales with Store Items**
  - Allows combined reporting of Store Sales and Store Items.

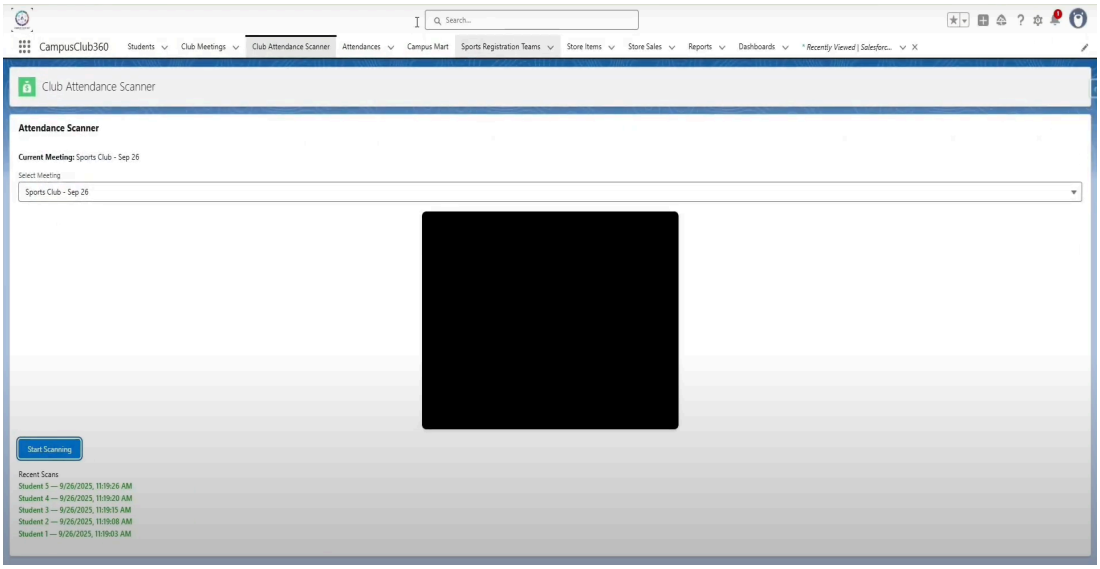


4. Testing & Results:

- **Club Attendance & Scanner**

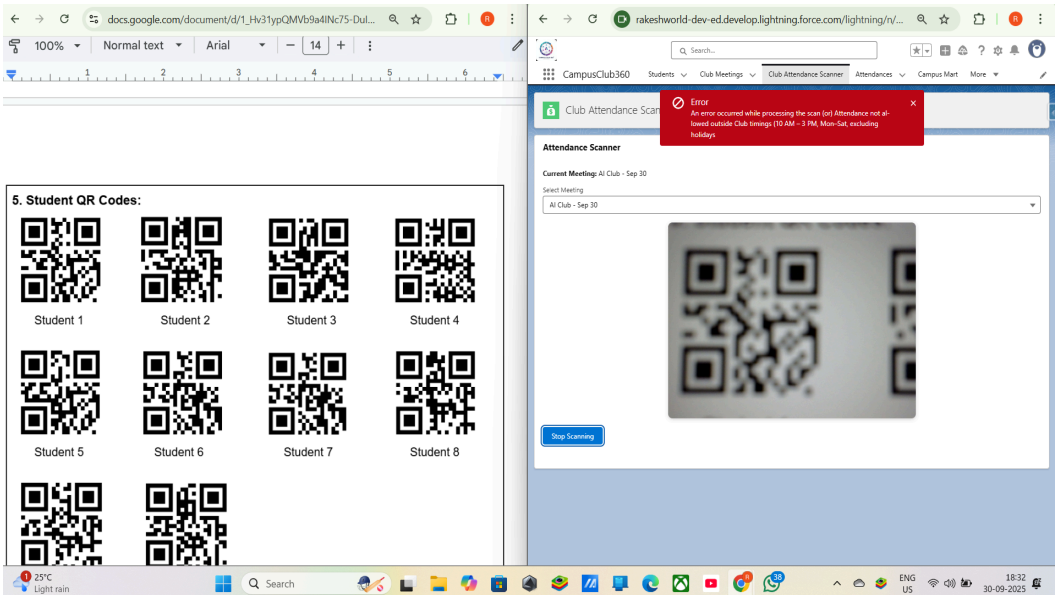
Test Case 1 (Within Hours: 10 AM – 3 PM)

- **Input:** Faculty scans student QR code.
- **Expected Result:** Attendance record created successfully.



Test Case 2 (Outside Hours: After 3 PM or Before 10 AM)

- **Input:** Faculty scans student QR code at 4:00 PM.
- **Expected Result:** Error message displayed → “Attendance can only be marked between 10 AM and 3 PM.” Record not created.

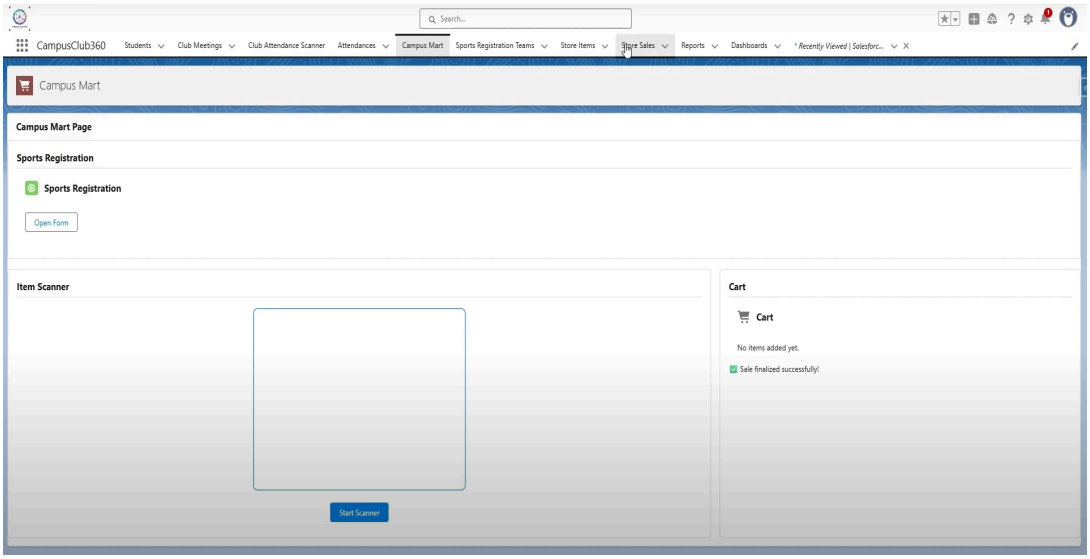


- **Campus Mart (Store Management)**

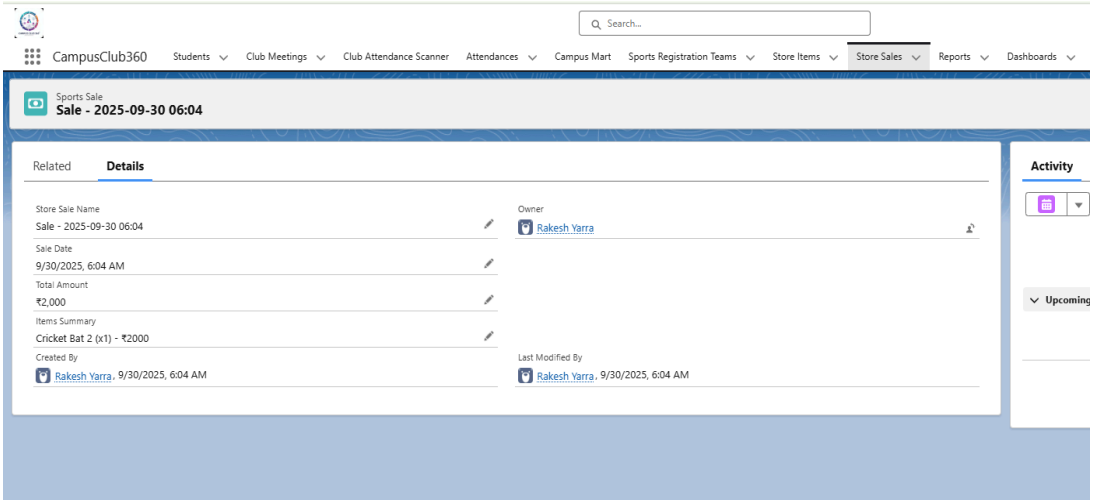
  Test Case 1 (Within Hours: 10 AM – 7 PM)

  - Input: Salesperson scans item barcode.

  - Expected Result: Item added to cart, transaction recorded in Store Sales, stock reduced.



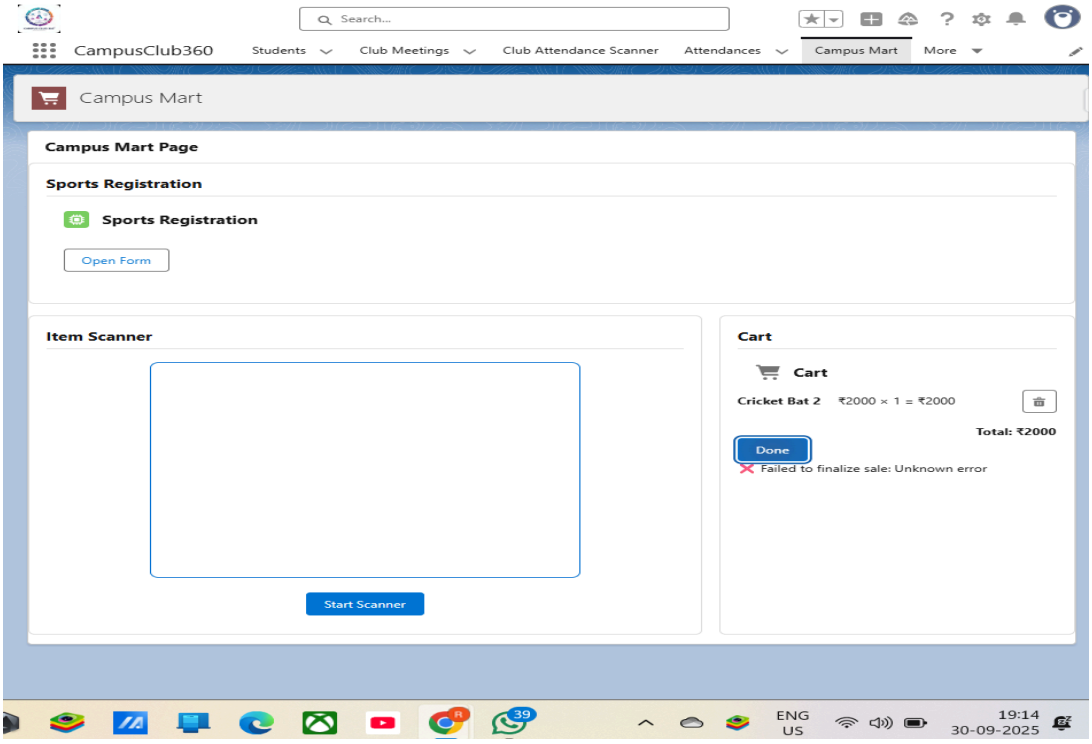
- Record is created on Store Sales.



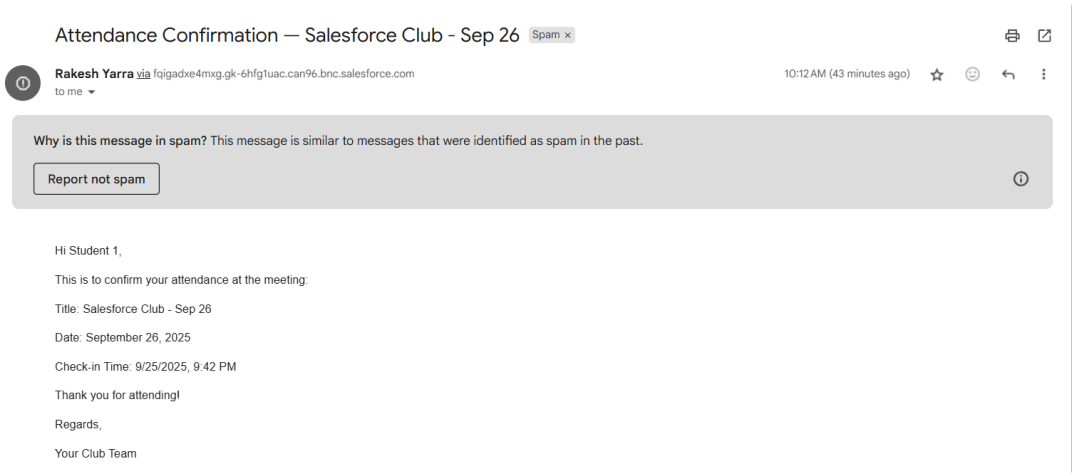
Test Case 2 (Outside Hours: After 7 PM or Before 10 AM)

- Input: Salesperson scans item barcode.

- Expected Result: Error message displayed → “Store transactions can only be recorded between 10 AM and 7 PM.” No record created.



- **Email Notifications**
  Objective: Validate communication to users.
  1. Confirmation Email:
     - Step: Student registers for a Club Pass.
     - **Expected Result:** Email confirmation sent to the student.



  2. Approval Email:
     - Step: Faculty submits student deletion request.
     - Expected Result: Approval email sent to Dean.

Raja csefaculty yarrarakesh39@gmail.com via wue56pfeekeq.gk-6hfg1uac.can96.bnc.salesforce.com
to me

Why is this message in spam? This message is similar to messages that were identified as spam in the past.

Report as not spam

Hello Dean,
A faculty member has requested to delete the student record below.
Please review the details and approve or reject the request in Salesforce.
Reason for Request: Hii Dean Mam,
THis student is not actively Particating in any Club and Clg Classess
Student Details:
- Student Name: Student 1
- Student ID: ECE0001
- Email: yarrarakesh39@gmail.com
- Department: Electronics and Communication Engineering
- Section: A
- Status: Pending Delete
Thank you,
Raja csefaculty

3. Attendance Notification:
   - **Step:** Faculty creates a meeting and scans attendance.
   - **Expected Result:** Students/Faculty receive meeting attendance mail.

Attendance Report for Salesforce Club - Sep 24 Spam ×

Rakesh Yarra via ryxfwb7k4zsqylgs.3vna6cb.gk-6hfg1uac.can96.bnc.salesforce.com
to me

Why is this message in spam? This message is similar to messages that were identified as spam in the past.

Report as not spam

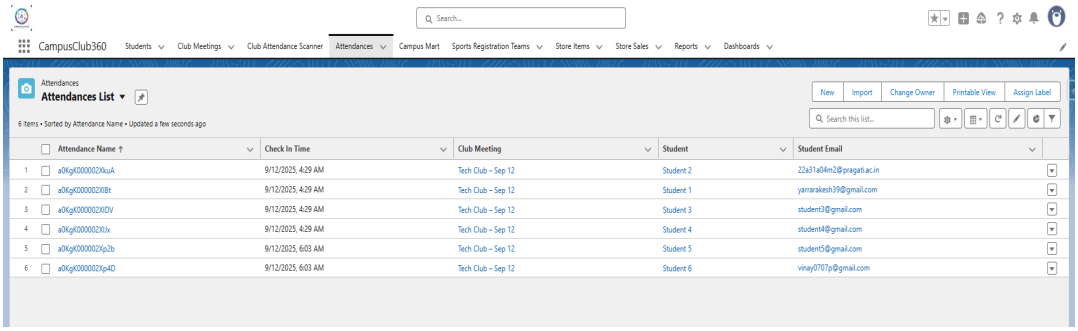
Dear Rakesh Yarra,

Here is the attendance for your meeting: Salesforce Club - Sep 24

Student Name	Email
Student 1	yarrarakesh39@gmail.com
Student 2	22a31a04m2@pragati.ac.in
Student 3	student3@gmail.com
Student 4	student4@gmail.com
Student 5	student5@gmail.com

Regards,
System

- Auto delete and send attendance to Created user.



Phase 10: Final Presentation & Demo

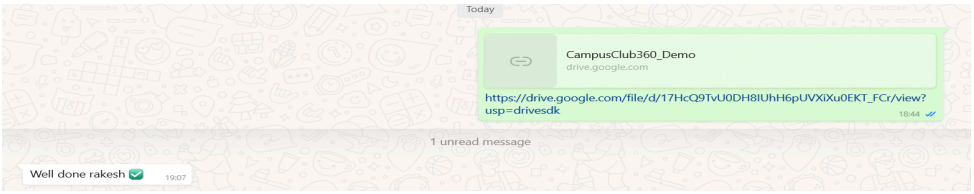
1.Pitch / Introduction:

"CampusClub360 – A unified Salesforce solution for campus clubs, sports, and store.
Our project streamlines student club management, sports registration, attendance tracking, and store sales – all in one Salesforce app."

2. Demo:

Youtube: https://youtu.be/rybcmtzcdDQ?si=vUVsof6XtlRAld23
Drive:
https://drive.google.com/file/d/17HcQ9TvU0DH8IUhH6pUVXiXu0EKT_FCr/view?usp=drivesdk

3. Feedbacks:



4. Portfolio Showcase:

- **GitHub:** https://github.com/YarraRakesh/CampusClub360.git
- **LinkedIn:** https://www.linkedin.com/posts/rakeshyarra_salesforce-smartbridge-tcs-activity-7377747270399848449-POW1?utm_source=social_share_send&utm_medium=member_desktop_web&rcm=ACoAAFDp4OYBZoia2hubIFAGV0wvezZbSq1Vlfc

Conclusion:

- This project automated student attendance tracking using QR code scanning.
- It streamlined sports registration and team management, ensuring fee collection and match records.
- It enhanced Campus Mart store management with barcode scanning, cart management, and sales tracking.
- Faculty and store managers now have better visibility and control, reducing errors and manual workload.

Note: CampusClub360 demonstrates how Salesforce can simplify and digitize student campus management, ensuring transparency, accuracy, and efficiency.

Acknowledgment:

- I sincerely thank Mir Abdul Rehman, Rakesh Bhoomani, Salesforce, SmartBridge, and the TCS Last Mile Program for their guidance, support, and opportunity to successfully complete this project.

References:

- Salesforce Official Documentation
- SmartBridge Learning Resources
- Trailhead Modules

Thank You