University Management System Using CRM

Phase 1

Problem Understanding & Industry Analysis

1. Requirement Gathering

👉 *Goal*: Understand what each stakeholder needs from the system.

- **Students**: Seamless admission process, easy access to class schedules, results, career guidance.
- **Faculty**: Automated exam scheduling, grading, progress tracking, less manual paperwork.
- **Parents/Guardians**: Transparent dashboards to monitor student performance & attendance.
- **University Admins**: Streamlined student lifecycle (inquiry → admission → graduation), reduced dropout rates.
- **Counselors**: Efficient assignment of students, career recommendation tools, progress monitoring.

2. Stakeholder Analysis

👉 *Goal*: Identify roles and responsibilities in the ecosystem.

- **Primary Stakeholders:**
    - Students → End users of the system.
    - Parents → Need dashboards for visibility.

    - Faculty → Need automation for exams/results.
    - Counselors → Assigned to guide students.

- **Secondary Stakeholders:**
    - University Management → Decision makers.
    - IT/Salesforce Admins → Maintain system.
    - Career Partners/Employers → Connect for placements.

3. Business Process Mapping

👉 *Goal*: Understand how things are done **today** vs. how Salesforce can improve it.

Current Process (Manual/Traditional):

- Students inquire via email/phone → delayed response.
- Counselors assigned manually → mismatch in guidance.
- Class schedules & exams tracked on spreadsheets.
- Parents rarely updated unless a problem occurs.

Proposed Process (Salesforce Enabled):

- Inquiries captured automatically in Salesforce → counselor auto-assigned based on interest.
- Courses, exams, and results managed in Salesforce.
- Parent dashboards updated in real time.
- Automated career guidance suggestions powered by AI.

4. Industry-Specific Use Case Analysis

👉 *Goal*: Benchmark against **best practices in EdTech**.

- **Dropout Problem**:
  - Studies show students drop out due to lack of personalized support.
  - Solution → Salesforce dashboards + AI-driven early intervention.
- **Parental Engagement**:
  - In top universities, parents have portals to monitor student performance.
  - Solution → Parent dashboards.
- **Career Guidance**:
  - Many colleges fail to connect students to jobs.
  - Solution → Career portal integrated with Salesforce Communities.

5. AppExchange Exploration

👉 *Goal*: Identify existing Salesforce apps to reduce development effort.

- **Potential Apps:**
  - Admissions Management Apps.
  - Student Success Hub (by Salesforce).
  - Learning Management System (LMS) Connectors.
  - Parent Communication Apps.

Phase 2: Org Setup & Configuration (Updated)

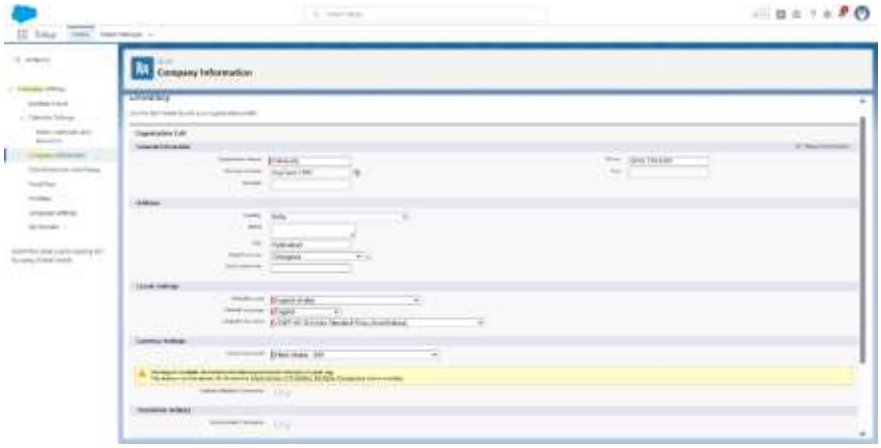
This updated phase establishes the Salesforce foundation for the University 360° Student Lifecycle Management project with the modification that students and parents do not have system access. Only Faculty and Placement Cell users are allowed to store and manage student records, analyze academic performance, and generate reports for placement purposes.

1. Salesforce Edition Selection

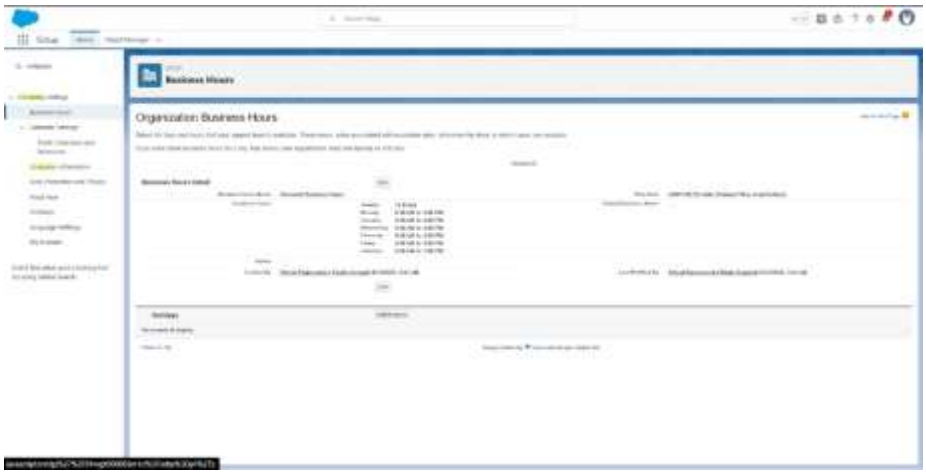
- Recommended: Salesforce Education Cloud (built on Sales & Service Cloud).
- Students and parents will not have user licenses.
- Only Faculty, Placement Cell, and Admins require full/Platform Salesforce licenses.

2. Company Profile Setup

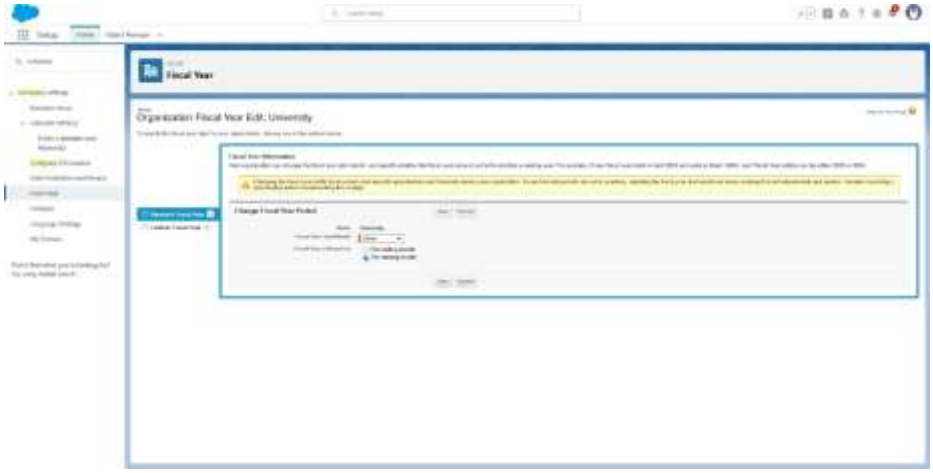
- Define University Org Profile (name, address, contact).



- Configure Business Hours (class timings, counselor/placement availability).

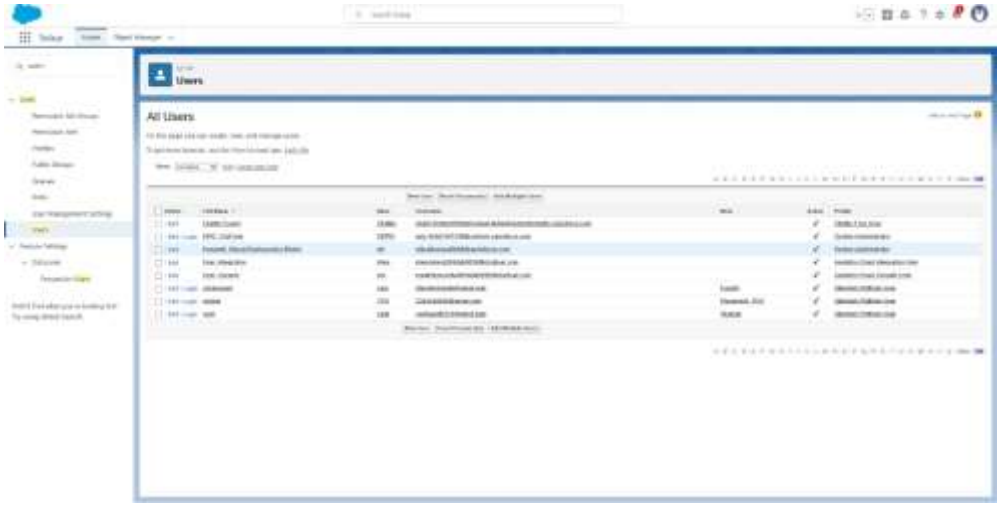


- Add Holidays (semester breaks, exam days).
- Align Fiscal Year with academic year (e.g., June–May).



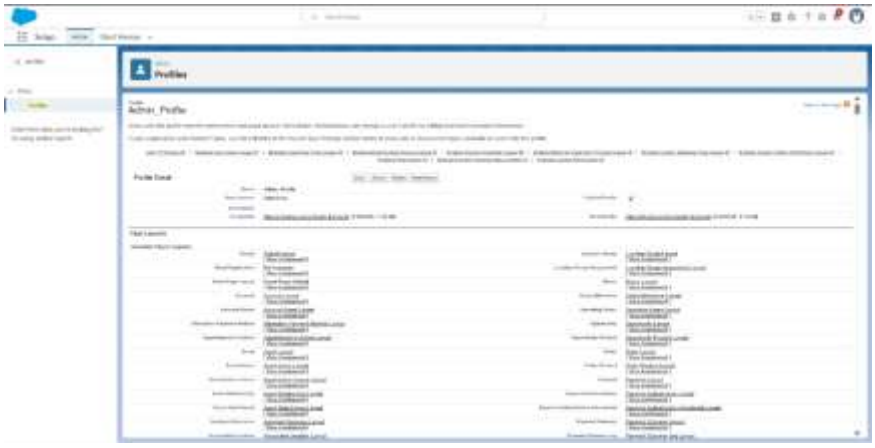
3. User Setup & Licenses

- Users to create:
  - Faculty (Standard Salesforce users).
  - Placement Cell Officers (Standard Salesforce users).
  - Admins & Management (System admins).
- ❌ Students and Parents are not system users. Their data exists only as records.
- Estimate number of faculty and placement cell licenses required.

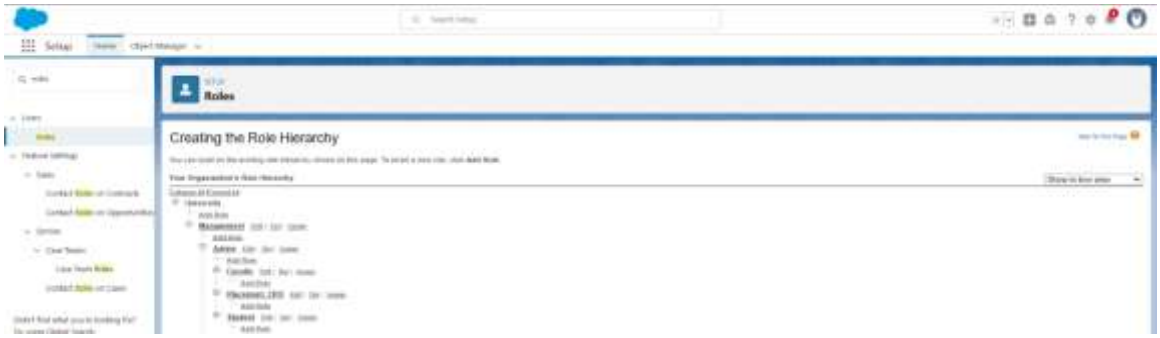


4. Profiles, Roles, and Permission Sets

- Profiles:
  - Faculty Profile – Create/update student records, exams, results.
  - Placement Cell Profile – View student performance, access reports, dashboards.
  - Admin Profile – Full system access.



- Roles:
  - Admin > Placement Cell > Faculty.
- Permission Sets (Optional):
  - Placement Cell Analytics Access – Provides extra permissions for reports, dashboards, and analytics.



5. Org-Wide Defaults (OWD) & Sharing Rules

- OWD Recommendations:
  - Student Data – Private (restricted to faculty and placement cell).
  - Course/Exam Info – Faculty can update, placement cell can view.
  - Placement Reports – Accessible only by placement cell and admins.
- Example Sharing Rules:
  - Placement Cell can access all student records for analysis.
  - Faculty access is limited to their own students.

6. Login & Security Policies

- Restrict logins to university network/IP ranges.
- Two-Factor Authentication (2FA) for placement cell and admins.
- Strong password policies (reset every 90 days).
- Login Hours – Faculty restricted to academic hours, placement cell as per requirement.

7. Dev Org & Sandbox Setup

- Developer Org for prototyping.
- Sandbox Types:
  - Developer Sandbox – For initial testing.

- Partial Copy Sandbox – For student data testing.
- Full Sandbox – For UAT (User Acceptance Testing).

- Use Change Sets or SFDX for deployment to production.

8. Steps to Configure Placement Cell Permission Set

1. Go to Setup → Permission Sets → New.
2. Create a Permission Set called 'Placement Cell Analytics Access'.
3. Under Object Settings → grant Read access to Student, Course, Exam, and Result objects.
4. Under System Permissions → enable Run Reports, Create and Customize Reports, Create and Customize Dashboards, and View Dashboards in Public Folders.
5. Assign the Permission Set to Placement Cell users.
6. Placement Cell can now analyze data without altering faculty workflows.

Outcome of Updated Phase 2

At the end of this updated Phase 2, the Salesforce University Org is configured such that:
- Students and parents are not system users.
- Faculty can create and update student records.
- Placement Cell can analyze performance and generate reports/dashboards.
- Admins retain complete access and control.

This secure and streamlined configuration ensures proper academic record management and placement analysis.

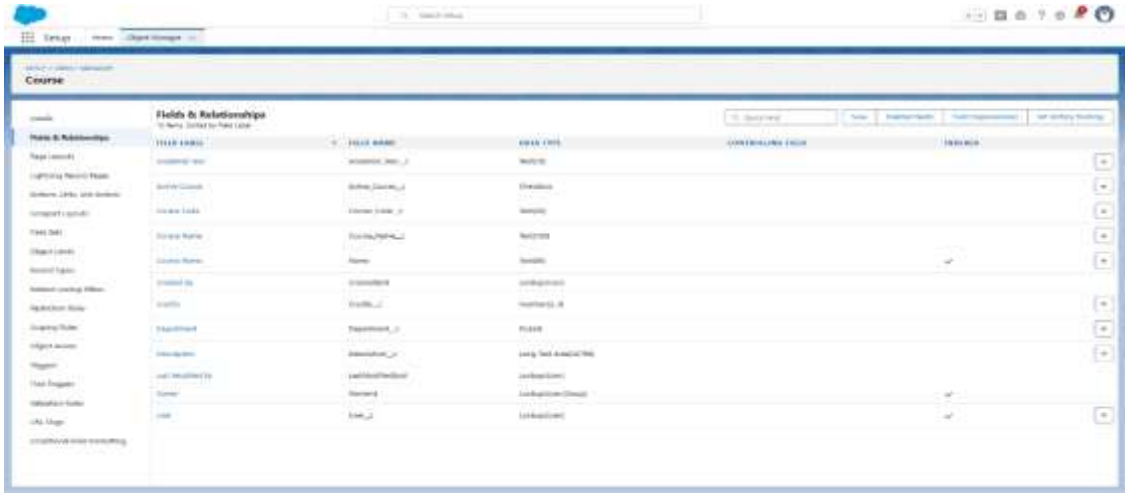
Phase 3: Data Modeling & Relationships

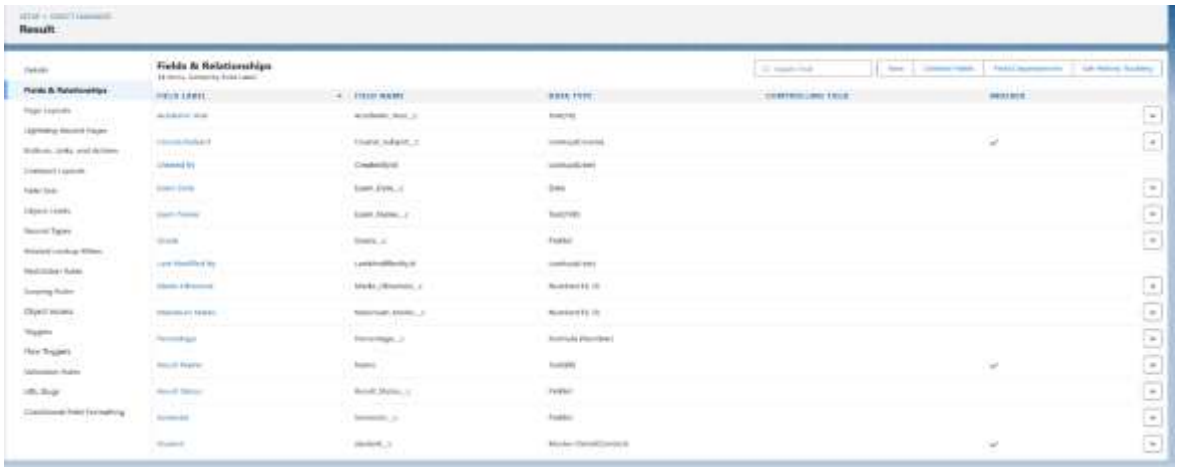
In this updated phase, the Salesforce data model is designed for the University 360° Student Lifecycle Management project using Salesforce best practices. Students and Parents are represented as Contacts, not as a separate custom Student object. This approach aligns with Salesforce Education Cloud standards. Faculty and Placement Cell are Users, while additional academic and performance details are managed using custom objects.

1. Objects to Use

Standard Objects:
- Contact → For Students and Parents.
- Account → For Departments/Programs (e.g., CSE, ECE, MBA).
- User → For Faculty, Placement Officers, and Admins.

Custom Objects:
1. Course → Stores course details.
2. Enrollment → Junction object linking Student (Contact) ↔ Course.
3. Exam → Stores exam details.
4. Result → Exam results linked to Exam + Student.
5. Attendance → Tracks student presence/absence per course.
6. Performance Analysis → Used by placement cell for evaluations and readiness analysis.





2. Key Fields for Each Object

Student (Contact extended): Roll Number, Department (Account lookup), Year of Study, Status (Active/Graduated/Dropped).
Parent (Contact extended): Relation, Mobile Number, Linked Student (Relationship field).
Course: Course Code, Name, Credits, Semester, Faculty Assigned (Lookup to User).
Enrollment: Student (Lookup to Contact), Course (Lookup to Course), Academic Year, Enrollment Status.
Exam: Exam Name, Type (Midterm/Final/External), Date, Linked Course.
Result: Student (Lookup to Contact), Exam (Lookup to Exam), Marks Obtained, Grade, Status.
Attendance: Student (Lookup to Contact), Course (Lookup to Course), Date, Status (Present/Absent).
Performance Analysis: Student (Lookup to Contact), Evaluated By (Placement Officer – User), Skill Score, Placement Readiness, Notes.

3. Relationships

- Student (Contact) ↔ Course → Many-to-many via Enrollment.
- Course ↔ Exam → One-to-many.
- Student (Contact) ↔ Result ↔ Exam → Results per student.
- Student (Contact) ↔ Attendance ↔ Course → Track attendance.
- Student (Contact) ↔ Performance Analysis → Multiple evaluations over time.
- Student (Contact) ↔ Parent (Contact) → Linked via Relationship field.
- Faculty (User) ↔ Course → Faculty assigned to courses.
- Placement Officer (User) ↔ Performance Analysis → Placement cell evaluations.

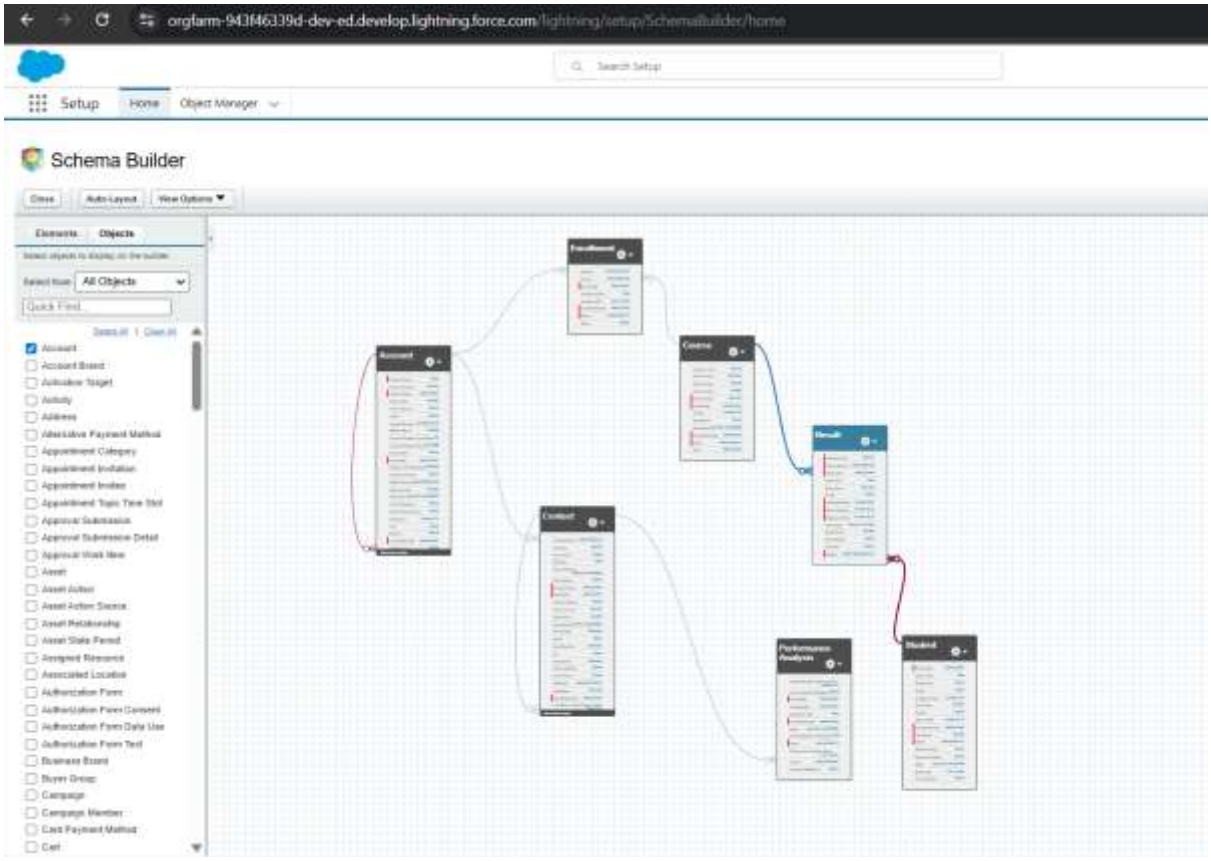
4. Record Types & Page Layouts

Record Types:
- Student: UG vs PG.
- Exam: Internal vs External.
- Performance Analysis: Semester Review vs Placement Review.

Page Layouts:
- Faculty Layout → Courses taught, attendance, results entry.
- Placement Cell Layout → Performance analysis, reports, dashboards.
- Admin Layout → Full access to student lifecycle.

5. Schema Builder View

Student (Contact) —< Enrollment >— Course —< Exam >— Result
Student (Contact) —< Attendance >— Course
Student (Contact) —< Performance Analysis
Student (Contact) — Parent (Contact) (Relationship)
Faculty (User) —< Course (Assigned)
Placement Officer (User) —< Performance Analysis



6. External Objects

If the university uses an external LMS or ERP system, Salesforce Connect can be used to link external data. For example, exam scores or course materials stored externally can be accessed directly within Salesforce without duplication.

Outcome of Updated Phase 3

At the end of this updated Phase 3, the Salesforce University Org will have:
- Students and Parents stored as Contact records, avoiding duplication.
- Departments represented as Accounts.
- Faculty and Placement Cell as Users.
- Academic and performance details stored in custom objects.
- Well-defined relationships supporting student lifecycle management and placement readiness analysis.

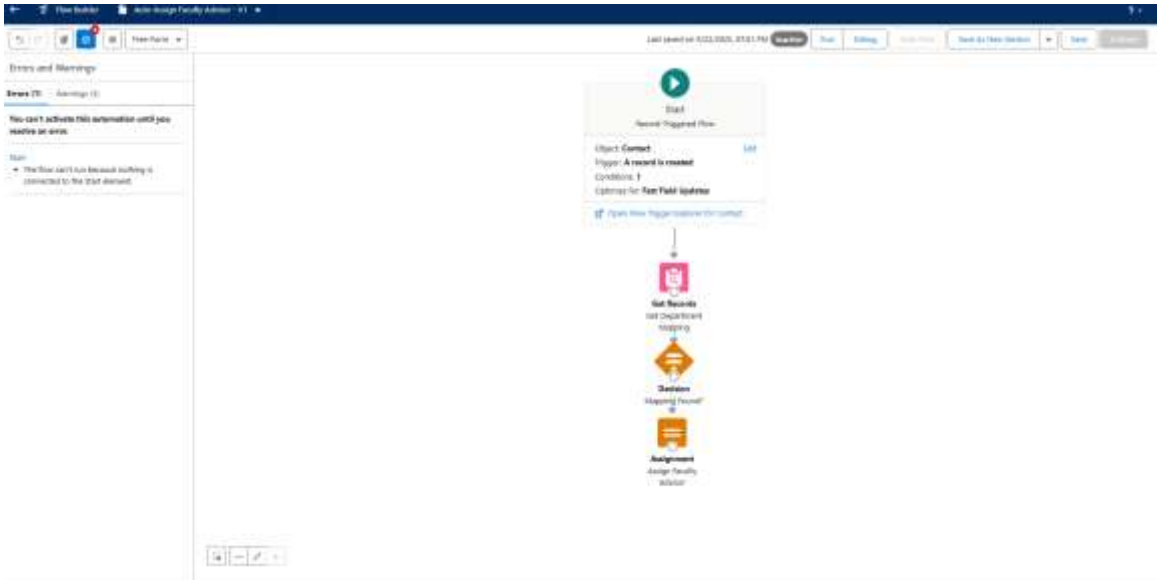
This approach ensures alignment with Salesforce Education Cloud best practices.

Phase 4: Process Automation (Admin)

In this phase, Salesforce automation tools are used to streamline student lifecycle processes. Since students and parents are records only, and faculty and placement cell are active users, automations are focused on faculty assignments, course enrollments, exam scheduling, result publishing, and validation rules to ensure data integrity.

1. Core Automation Requirements

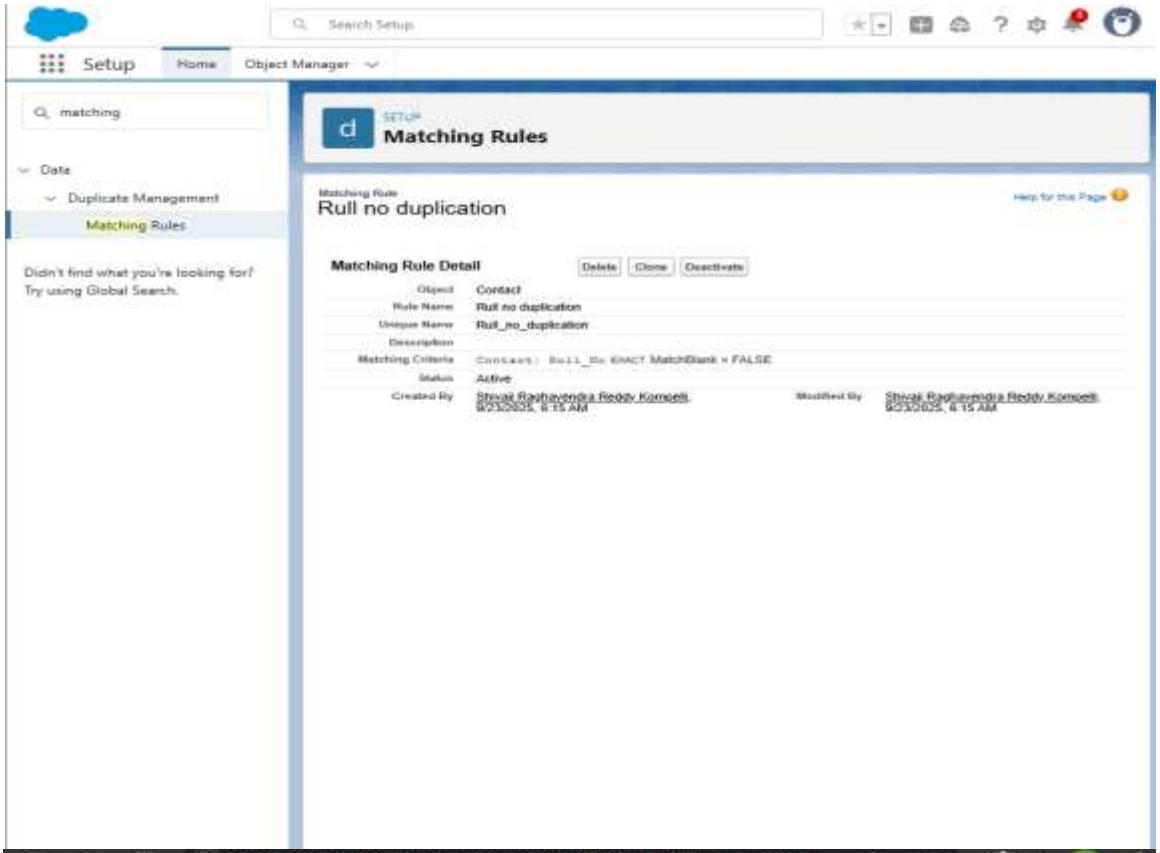
1. Faculty Assignment:
   - When a new Student (Contact) is created, auto-assign a Faculty Advisor based on Department.
   - Tool: Record-Triggered Flow.

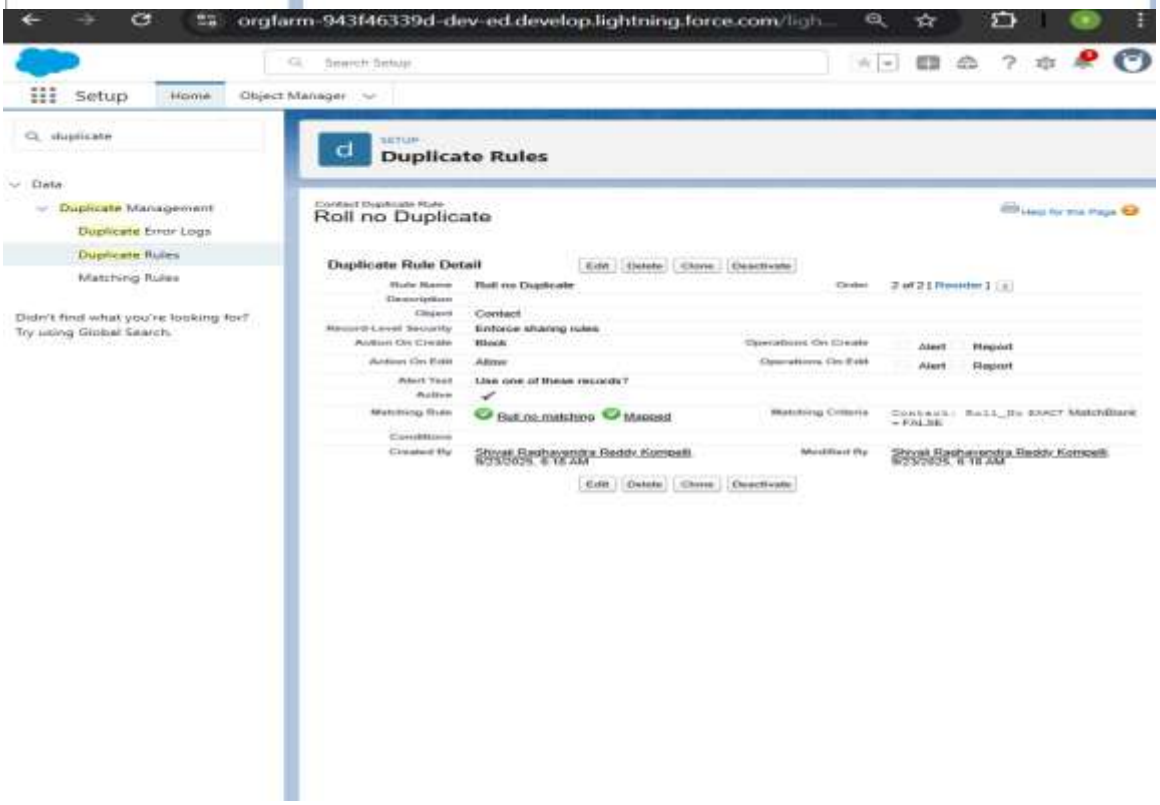


2. Data Validation:
   - Prevent duplicate Roll Numbers.
   - Ensure marks do not exceed maximum marks.
   - Tool: Validation Rules.

2. Automation Tools

- Validation Rules → Enforce data quality (e.g., duplicate roll numbers).

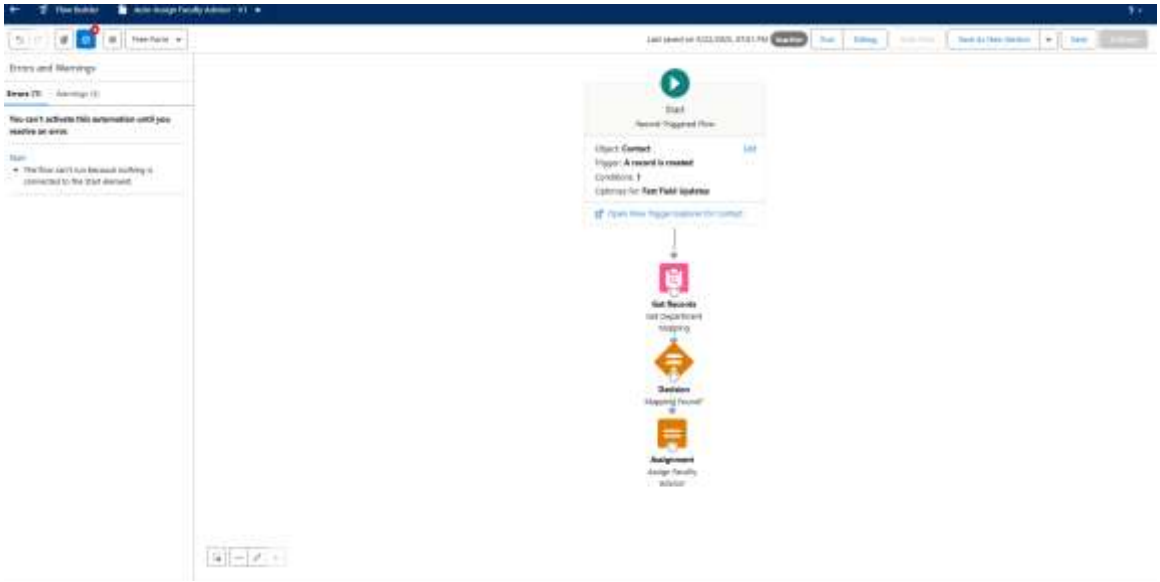




- Flows → Recommended tool for automation:
  - Record-Triggered Flow → Assign faculty, notify placement cell.

3. Example Automations

Automation 1: Auto-Assign Faculty Advisor
- Trigger: Student (Contact) created.
- Logic: If Department = CSE → Assign Faculty Advisor = varaprasad
- Action: Update Student record with Advisor field.



Automation 3: Validation – Prevent Duplicate Roll No
- Rule: If another Contact exists with the same Roll No → block save.

Automation 4: Validation – Marks Validation
- Rule: Marks Obtained must be ≤ Maximum Marks in the Exam.

4. Security in Automation

- Automations must respect user permissions.
- Only Faculty and Placement Users can trigger or execute actions.
- Use Permission Sets to control who can approve or trigger automation.
- Ensure sensitive data is only shared with Placement Cell and authorized faculty.

Outcome of Phase 4

At the end of Phase 4, the Salesforce University Org will:
- Automatically assign faculty advisors when students are created.
- Notify faculty on student course enrollments.
- Remind faculty of exam scheduling deadlines.
- Notify placement cell when results are published.
- Prevent invalid or duplicate data with validation rules.

Phase 5: Apex Programming (Developer)

In this phase, Apex programming is implemented to handle complex academic logic that cannot be managed with point-and-click tools. Key focus areas include GPA calculation, bulk processing of exam results, and scheduled reporting for the Placement Cell.

1. Purpose & Scope

- Automate academic performance logic beyond Flows.
- Core use cases: GPA calculation, bulk result processing, scheduled reports.
- Access restricted to Faculty and Placement Cell users.

2. Key Apex Use Cases

Use Case	Trigger / Context	Action / Logic
GPA Calculation	Result insert/update	Compute Student GPA and update Contact.
Bulk Exam Result Processing	End-of-semester uploads	Batch Apex processes results in chunks, updates GPA.
Scheduled Academic Report	Periodic (monthly/semester)	Scheduled job compiles reports, notifies Placement Cell.

3. Classes & Structure

- GPAService.cls → Handles GPA calculations.
- ResultTrigger.trigger → Recalculates GPA when Result changes.
- BulkResultProcessor.cls → Batch Apex for bulk GPA updates.
- AcademicReportScheduler.cls → Scheduled Apex for periodic reports.

```apex
    global void execute(Database.BatchableContext bc, List<Contact> scope) {
        // Build email content for this batch
        String batchBody = '';
        for (Contact c : scope) {
            batchBody += c.Name + ' (Roll No: ' + c.Roll_No__c + ') - GPA: ' + c.GPA__c + '\n';
        }

        // Store results across batches
        if (batchBody != '') {
            results += batchBody;
        }
    }

    // Collect results
    global String results = '';

    // Send email after all batches complete
    global void finish(Database.BatchableContext bc) {
        if (results == '') return;

        Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
        mail.setToAddresses(new String[] { 'placementcell@college.edu' }); // Replace with actual
        mail.setSubject('Weekly Low GPA Alert - Batch Report');
        mail.setPlainTextBody('⚠ Students with GPA below 6:\n\n' + results);

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

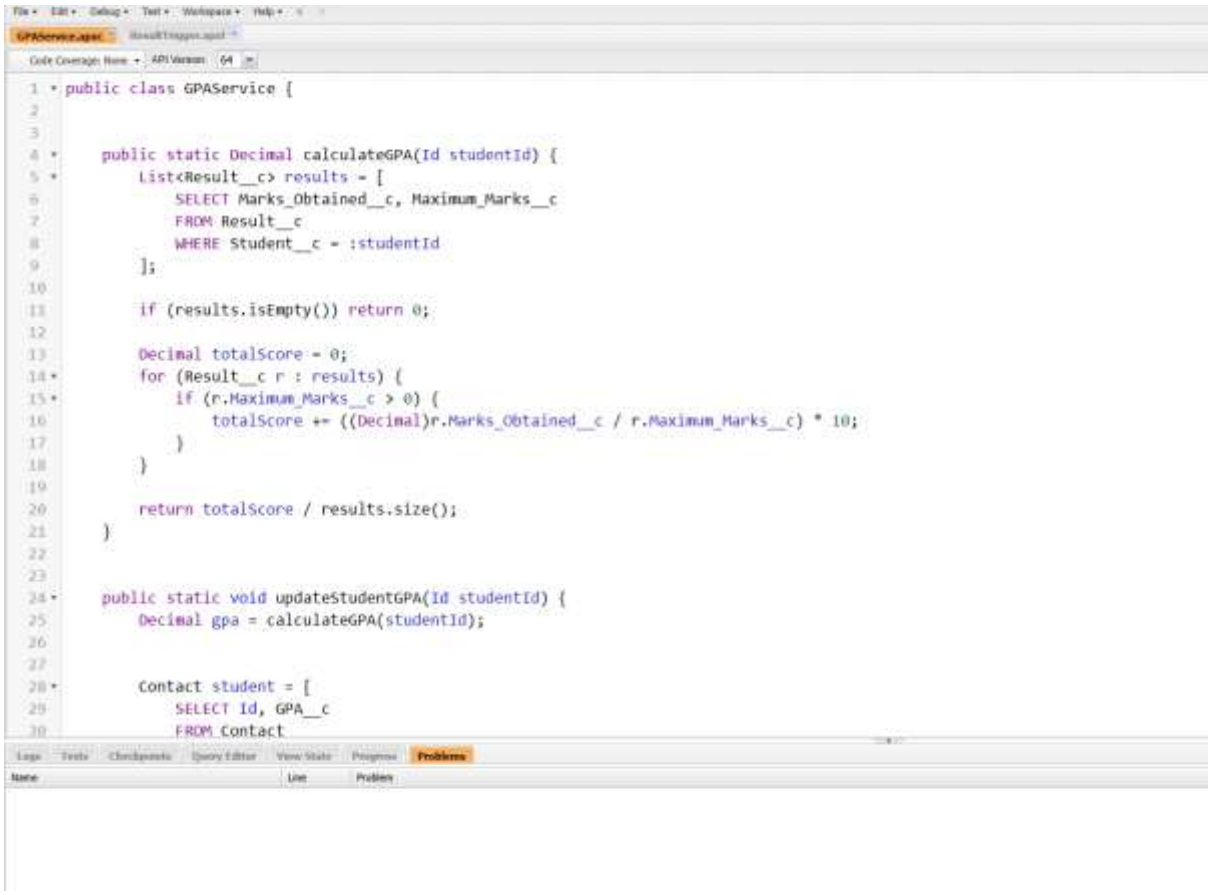
```apex
global class WeeklyLowGPABatchScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        WeeklyLowGPABatch batch = new WeeklyLowGPABatch();
        Database.executeBatch(batch, 200);
    }
}
```

```apex
String cronExp = '0 0 9 ? * MON';
System.schedule('Weekly Low GPA Batch Report', cronExp,
    new WeeklyLowGPABatchScheduler());
```

Example: GPAService Class

Sample outline:

```
public with sharing class GPAService {
   public static Decimal calculateGPA(Id studentId) {
      // Query Results
      // Compute GPA = (Marks / MaxMarks) * 10
      // Return rounded value
   }

   public static Map<Id, Decimal> calculateGPA(List<Id> studentIds) {
      // Bulk calculation logic
   }
}
```



```
public class GPAService {


    public static Decimal calculateGPA(Id studentId) {
        List<Result__c> results = [
            SELECT Marks_Obtained__c, Maximum_Marks__c
            FROM Result__c
            WHERE Student__c = :studentId
        ];

        if (results.isEmpty()) return 0;

        Decimal totalScore = 0;
        for (Result__c r : results) {
            if (r.Maximum_Marks__c > 0) {
                totalScore += ((Decimal)r.Marks_Obtained__c / r.Maximum_Marks__c) * 10;
            }
        }

        return totalScore / results.size();
    }


    public static void updateStudentGPA(Id studentId) {
        Decimal gpa = calculateGPA(studentId);


        Contact student = [
            SELECT Id, GPA__c
            FROM Contact
```

5. Triggers

ResultTrigger:

```
trigger ResultTrigger on Result__c (after insert, after update) {
   // Collect Student Ids
```

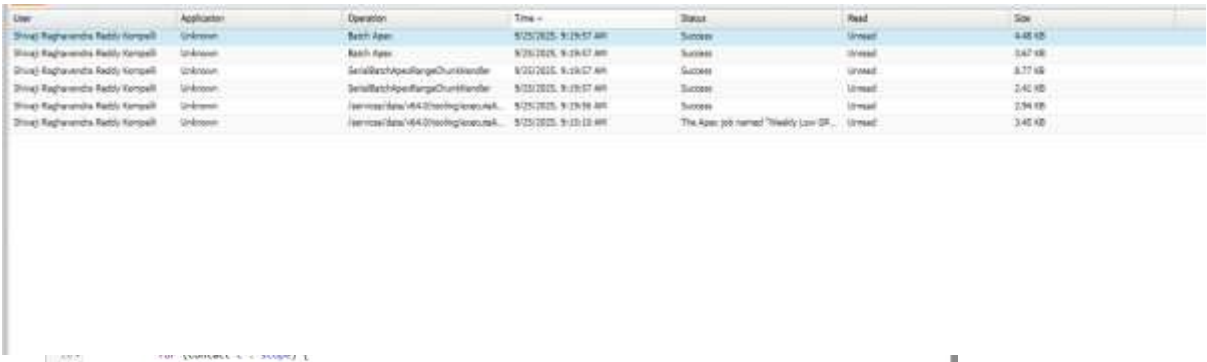
```
    // Call GPAService.calculateGPA
    // Update GPA__c on Contact
}
```

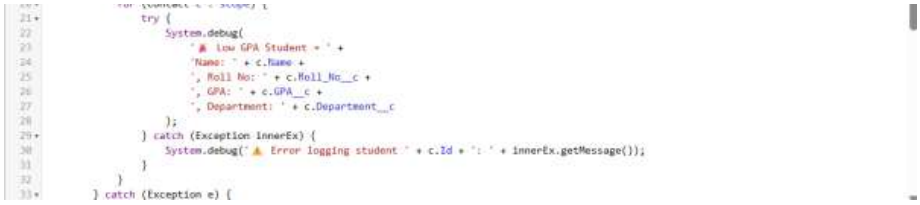
```
trigger ResultTrigger on Result__c (after insert, after update, after delete) {
    Set<Id> studentIds = new Set<Id>();

    if (Trigger.isInsert || Trigger.isUpdate) {
        for (Result__c r : Trigger.new) {
            if (r.Student__c != null) {
                studentIds.add(r.Student__c);
            }
        }
    }
    if (Trigger.isDelete) {
        for (Result__c r : Trigger.old) {
            if (r.Student__c != null) {
                studentIds.add(r.Student__c);
            }
        }
    }

    GPAService.updateMultipleStudents(studentIds);
}
```

6. Batch Apex

BulkResultProcessor implements Database.Batchable<Result__c>:

- start(): Query Results
- execute(): Recalculate GPA per student
- finish(): Optional summary notification





```
            try {
                System.debug(
                    '⚠ Low GPA Student - ' +
                    'Name: ' + c.Name +
                    ', Roll No: ' + c.Roll_No__c +
                    ', GPA: ' + c.GPA__c +
                    ', Department: ' + c.Department__c
                );
            } catch (Exception InnerEx) {
                System.debug('⚠ Error logging student ' + c.Id + ': ' + InnerEx.getMessage());
            }
        }
    } catch (Exception e) {
```

7. Scheduled Apex

AcademicReportScheduler implements Schedulable:

- execute(): Compile department-wise results, GPA averages.
- Email report sent to Placement Cell.

Schedule with CRON, e.g. monthly at 6 AM.

```
    global void execute(Database.BatchableContext bc, List<Contact> scope) {
        // Build email content for this batch
        String batchBody = '';
        for (Contact c : scope) {
            batchBody += c.Name + ' (Roll No: ' + c.Roll_No__c + ') - GPA: ' + c.GPA__c + '\n';
        }

        // Store results across batches
        if (batchBody != '') {
            results += batchBody;
        }
    }

    // Collect results
    global String results = '';

    // Send email after all batches complete
    global void finish(Database.BatchableContext bc) {
        if (results == '') return;

        Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
        mail.setToAddresses(new String[] { 'placementcell@college.edu' }); // Replace with actual
        mail.setSubject('Weekly Low GPA Alert - Batch Report');
        mail.setPlainTextBody('⚠ Students with GPA below 6:\n\n' + results);

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

```
Enter Apex Code:
LowGPABatch job = new LowGPABatch();
Database.executeBatch(job, 100);
```

```
Exception innerEx) {
n.debug('⚠ Error logging student ' + c.Id + ': ' + innerEx.getMessage());
```

8. Test Classes

Test classes are mandatory. Ensure ≥ 75% coverage:
- GPAServiceTest
- ResultTriggerTest
- BulkResultProcessorTest
- AcademicReportSchedulerTest

9. Exception Handling & Best Practices

- Use 'with sharing' to respect permissions.
- Bulkify all logic.
- Avoid SOQL in loops.
- Handle missing data gracefully.

```
LowGPABatch.apex *
Code Coverage: None   API Version: 64
global class LowGPABatch implements Database.Batchable<SObject>, Database.Stateful {


    global Database.QueryLocator start(Database.BatchableContext bc) {
        try {
            return Database.getQueryLocator(
                'SELECT Id, Name, Roll_No__c, GPA__c, Department__c ' +
                'FROM Contact WHERE GPA__c < 6'
            );
        } catch (Exception e) {
            System.debug('⚠ Error in Batch start: ' + e.getMessage());

            return Database.getQueryLocator('SELECT Id FROM Contact WHERE Id = null');
        }
    }


    global void execute(Database.BatchableContext bc, List<Contact> scope) {
        try {
            for (Contact c : scope) {
                try {
                    System.debug(
                        '⚠ Low GPA Student = ' +
                        'Name: ' + c.Name +
                        ', Roll No: ' + c.Roll_No__c +
                        ', GPA: ' + c.GPA__c +
                        ', Department: ' + c.Department__c
                    );
                } catch (Exception InnerEx) {
                    System.debug('⚠ Error logging student ' + c.Id + ': ' + innerEx.getMessage());
                }
            }
        } catch (Exception e) {
```

10. Security & Deployment

- Sensitive fields like GPA visible only to Faculty & Placement Cell.
- Permission sets control access.
- Use Sandbox → Test → Deploy approach.

11. Outcome of Phase 5

- GPA auto-calculated when Results are updated.
- Bulk processing supported with Batch Apex.
- Placement Cell receives periodic reports.
- Scalable, secure, and developer-optimized implementation.

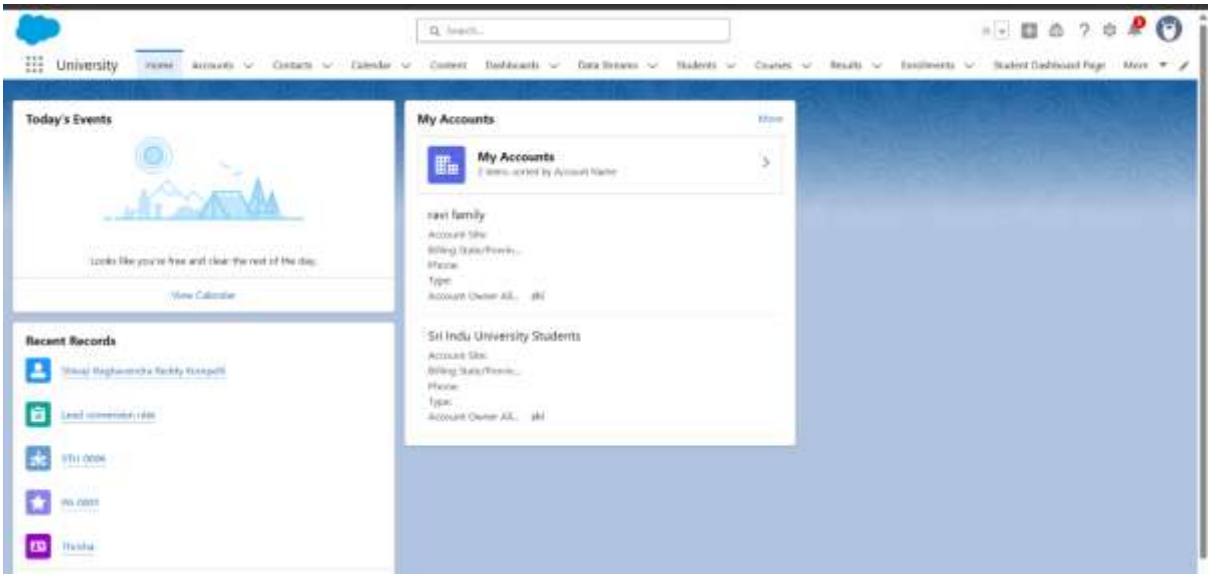
Phase 6 — University Management App: Documentation

Overview

Phase 6 focuses on **User Interface Development** for the University Management App. The goal is to make Salesforce user-friendly for **Faculty** and **Placement Cell**, providing dashboards, record pages, tabs, utility tools, and interactive components.

1) Lightning App Builder

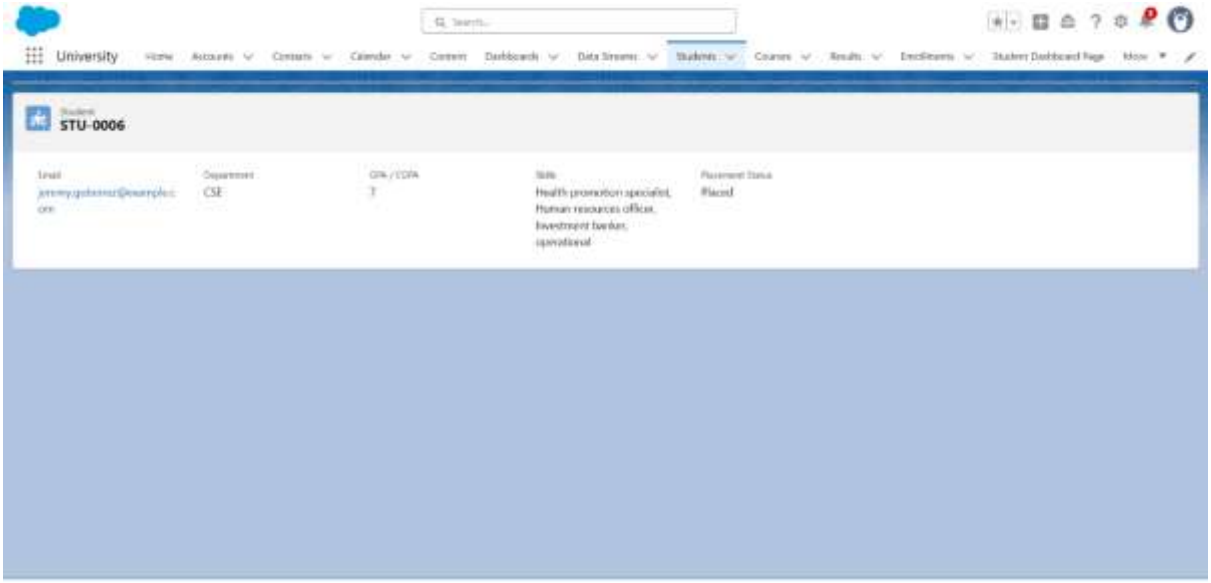
- **App Name:** University Management App
- **Navigation Items:** Students (Contacts), Departments (Accounts), Courses, Enrollments, Exams, Results, Dashboards
- **Purpose:** Central entry point for Faculty & Placement Cell.



2) Record Pages

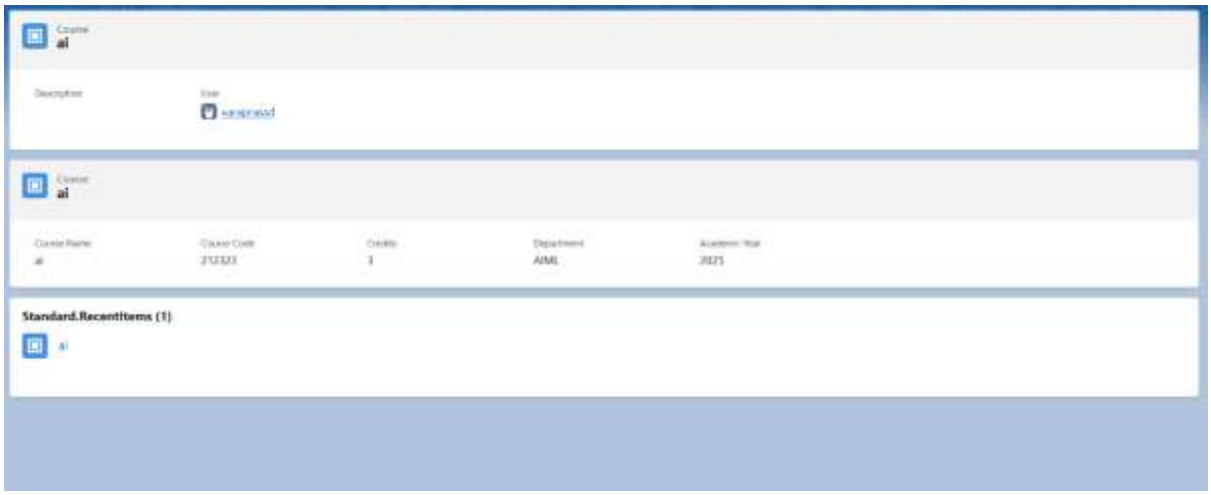
Student (Contact) Page

- **Tabs:**
  - Details: Roll No, Department, GPA (Dynamic Form)
  - Related: Courses, Results
- **Dynamic Forms:** Show GPA only if Results exist



Course Page

- **Tabs:**
    - Course Info
    - Related Exams
    - Enrollments



Exam Page

- **Tabs:**
    - Exam Details
    - Related Results

3) Tabs

- Custom Tabs: Courses, Enrollments, Exams, Results
- Added to University Management App navigation

4) Home Page Layouts

Placement Cell Home

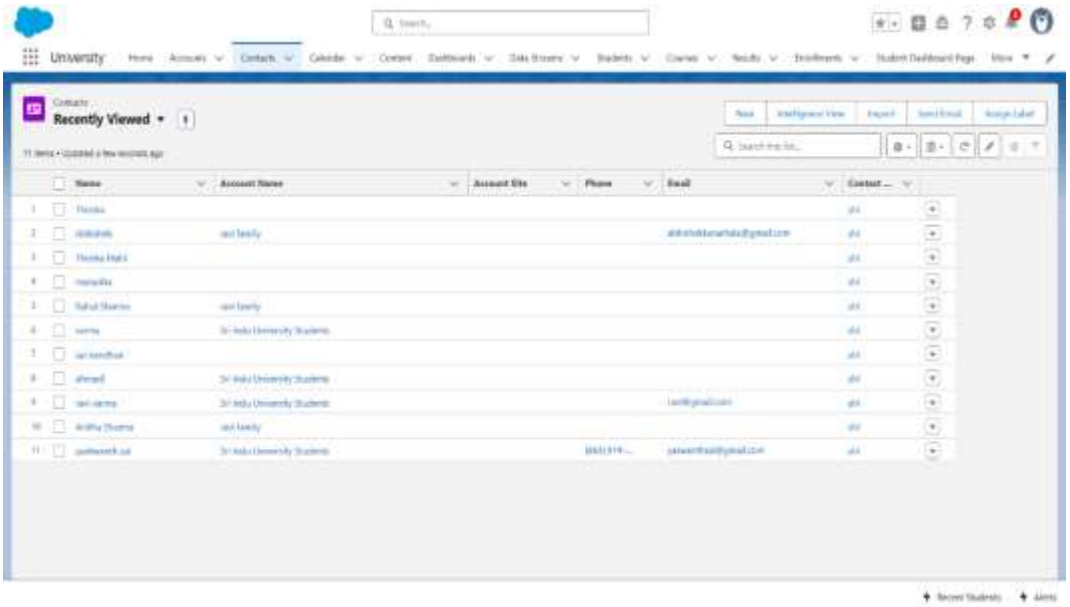
- Dashboard Components: Eligible Students, Top GPA chart, Pass/Fail trends
- Quick Links: Add Exam, Upload Results

Faculty Home

- Recent Courses taught
- Notifications: Upcoming Exams
- Reports: Student GPA distribution

5) Utility Bar

- Notes: Quick faculty notes
- Chatter: Internal communication
- Recent Students: Quick lookup of student records



6) Lightning Web Components (LWC)

studentPerformanceCard

- Displays Student GPA, Courses, Placement Eligibility
- Uses **@wire** for fetching data and **imperative Apex** for refresh

examResultsUploader

- Allows faculty to upload multiple exam results via CSV
- Uses imperative Apex to save results in bulk

7) Apex with LWC

Studentdashboardaura.cmp

- getStudentResults(Id studentId) → fetch student results
- saveResults(Id examId, List<ResultInput> inputs) → save multiple results
- ResultInput class for passing studentId, marks, grade

Outcome of Phase 6

- Faculty & Placement Cell get user-friendly Lightning pages
- Custom LWCs make GPA, Results, and Placement Eligibility interactive
- Mix of **Wire Adapters** (fetch) + **Imperative Apex Calls** (save) ensures performance and flexibility
- End users can access everything from the custom app, pages, dashboards, and utility bar without digging into backend

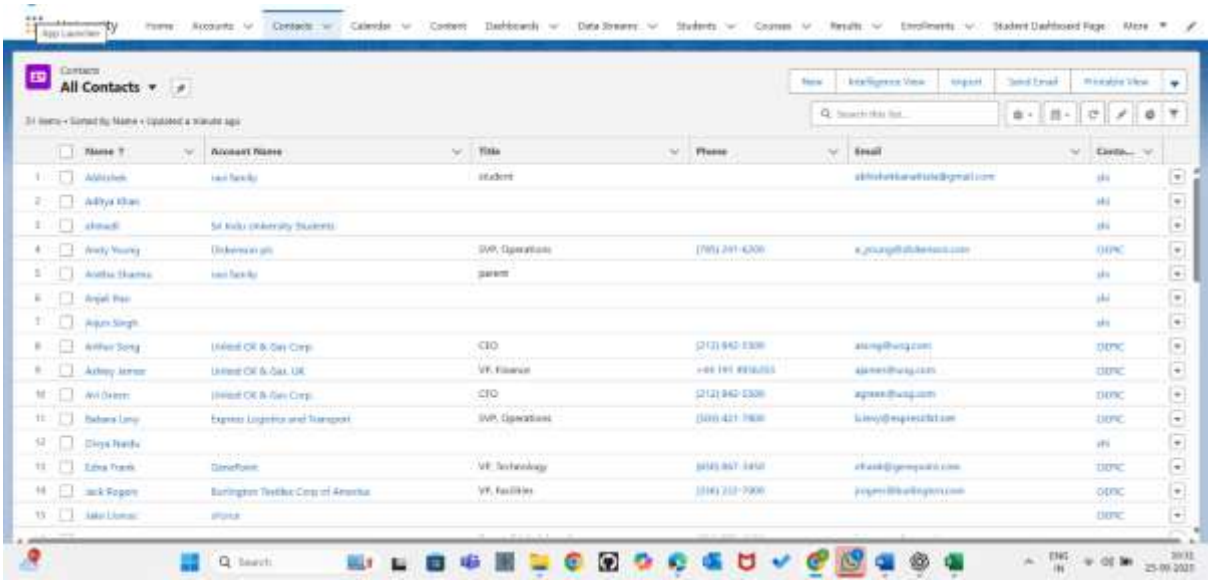
Phase 8 — Data Management & Deployment

Overview

Phase 8 focuses on **managing data efficiently** in Salesforce and deploying changes across environments. This ensures that the University Management App remains consistent, backed up, and properly migrated.

1) Data Import Wizard

- **Purpose:** Import small to medium volumes of records (Contacts, Accounts, Leads, Custom Objects)
- **Steps:**
    1. Setup → Data Import Wizard
    2. Select Object → Map CSV fields to Salesforce fields → Review → Start Import
- **Use case:** Import initial student, course, exam, and result data.

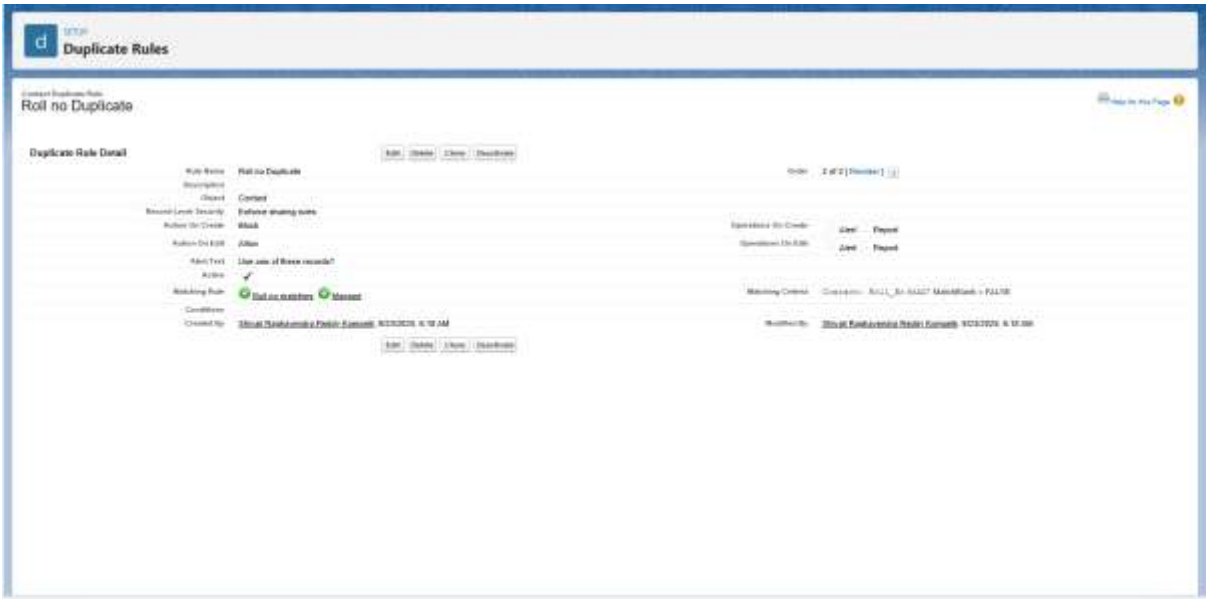


2) Data Loader

- **Purpose:** Bulk import/export of large data sets
- **Steps:**
    1. Install Data Loader
    2. Login → Choose Insert, Update, Upsert, Export, or Delete
    3. Select CSV file → Map fields → Execute
- **Use case:** Upload 1000+ student exam results efficiently.

3) Duplicate Rules

- **Purpose:** Prevent duplicate records (Contacts, Accounts, Students)
- **Setup:**
    1. Setup → Duplicate Management → Matching Rules → Duplicate Rules
    2. Define criteria (e.g., Email + Roll No) → Activate



4) Data Export & Backup

- **Purpose:** Regularly back up Salesforce data
- **Steps:**
    1. Setup → Data Export → Schedule Export
    2. Choose Objects → Export as CSV → Store securely
- **Recommendation:** Weekly backup for Students, Courses, Exams, Results

5) Change Sets

- **Purpose:** Deploy metadata between Salesforce orgs (Sandbox → Production)
- **Steps:**
    1. Create Outbound Change Set in Sandbox
    2. Add Components (Objects, Fields, Pages, LWCs, Apex Classes)
    3. Upload → Deploy in Production → Validate → Deploy

6) Unmanaged vs Managed Packages

- **Unmanaged Package:** For sharing code/configuration; editable after installation
- **Managed Package:** Locked components; used for AppExchange apps
- **Use case:** Package custom University App for reuse or deployment

7) ANT Migration Tool

- **Purpose:** Script-based deployment for large/complex changes
- **Steps:**
    1. Install ANT Migration Tool
    2. Define package.xml → Specify components
    3. Run commands: retrieve/deploy
- **Use case:** Automate deployment of LWCs, Apex, and page layouts across orgs

8) VS Code & SFDX

- **Purpose:** Modern developer workflow with Salesforce DX

- **Steps:**
    1. Install VS Code + Salesforce Extensions Pack
    2. Authenticate with org using SFDX CLI
    3. Retrieve / Push metadata using sfdx force:source:retrieve and sfdx force:source:deploy
- **Use case:** Version control, source-driven development, team collaboration

Outcome of Phase 8

- Efficient import/export and backup of Salesforce data
- Duplicate prevention ensures clean records
- Smooth deployment from Sandbox to Production using Change Sets, ANT, or VS Code/SFDX
- Knowledge of packaging ensures scalability and reuse

.

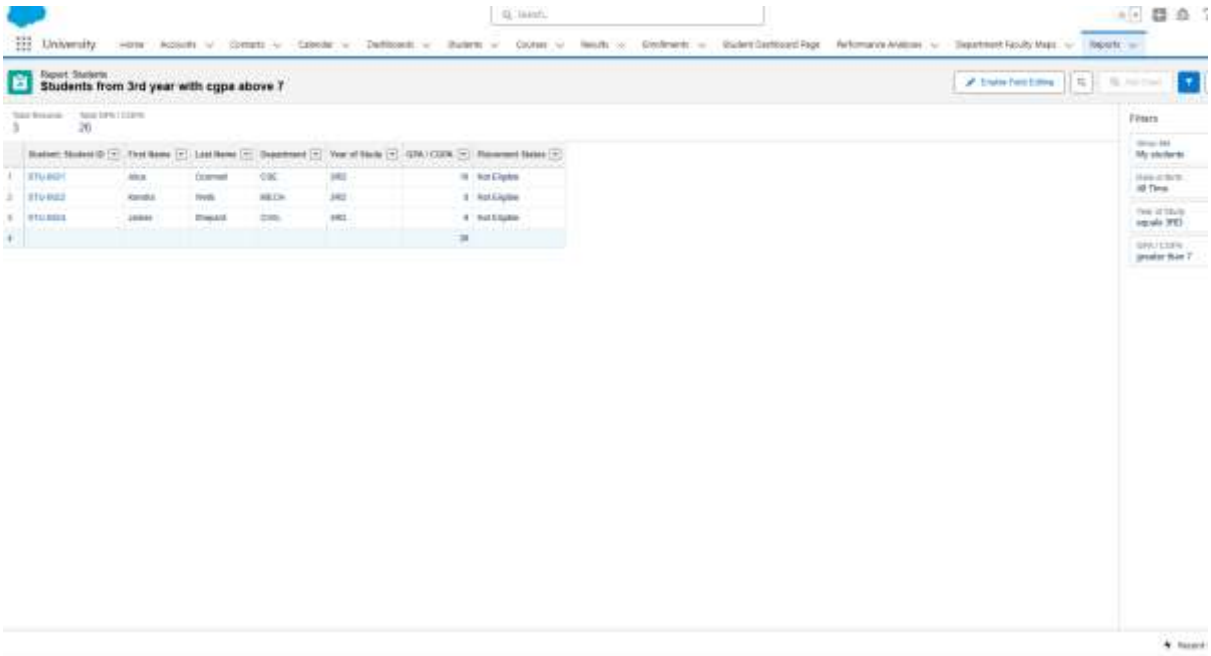
Phase 9: Reporting, Dashboards & Security Review

1. Reports

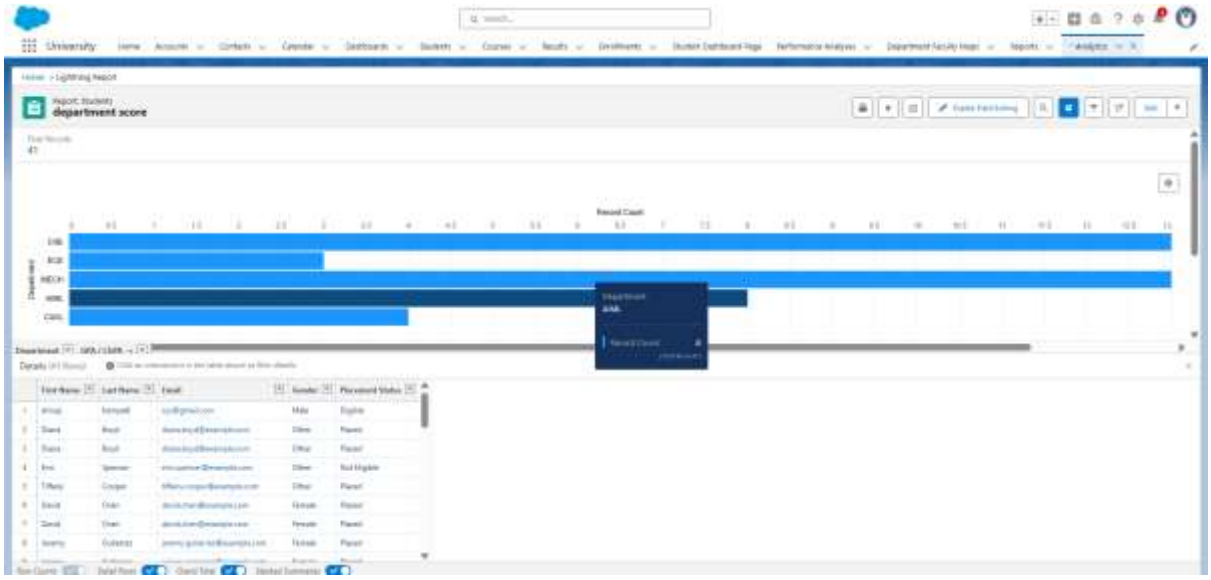
Salesforce reports are used to analyze student and placement data.

Types of Reports Implemented

1. **Tabular Report**
    - Purpose: Simple list of students with GPA, department, and placement status.
    - Example: “All Students List” or “Students with GPA > 8”.



2. **Summary Report**
    - Purpose: Group and summarize records by fields.
    - Example: “Department-wise GPA Summary” – shows average GPA and student count per department.
3. **Matrix Report**
    - Purpose: Group records by both rows and columns.
    - Example: GPA distribution by **Department vs Year of Study**.
4. **Joined Report**
    - Purpose: Combine data from multiple objects.
    - Example: Compare **Student Records vs Placement Records** in one report.



2. Report Types

- **Standard Report Types**: Available for Student and Placement custom objects.
- **Custom Report Types**:
  - Example: “Students with Exam Results” (Student object + Exam Result object).
  - Allowed cross-object analysis for academic and placement performance.

3. Dashboards

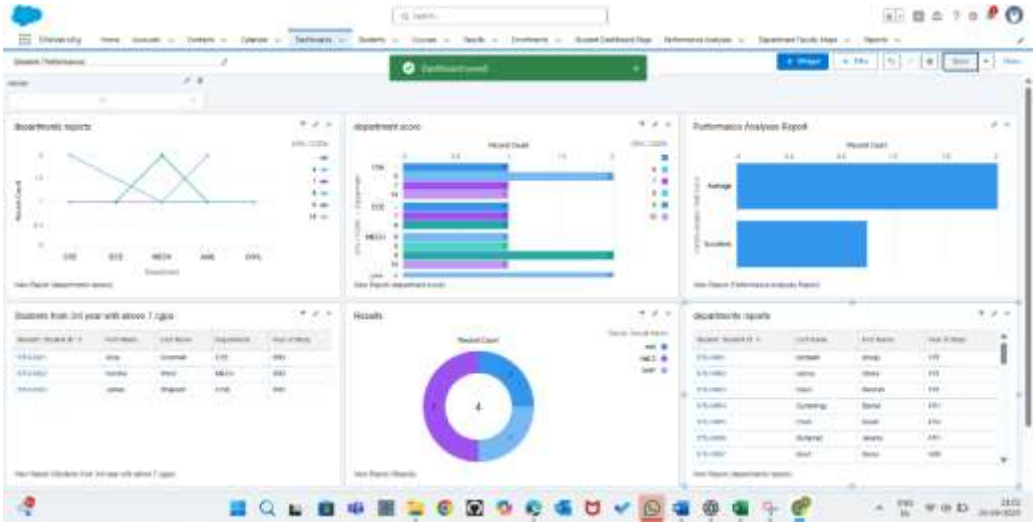
Dashboards were created to visualize reports in real time.

Key Dashboard Components

1. **Average GPA by Department** – Bar Chart (from summary report).
2. **Placement Status Distribution** – Pie Chart (from student report).
3. **Top 10 Students by GPA** – Table.
4. **Overall University Average GPA** – Gauge.

Dynamic Dashboards

- Configured dashboards to run as **“Logged-in User”** so department heads see only their department’s data.

4. 

Sharing Settings

- **Organization-Wide Defaults (OWD):** Student records set to **Private**.
- **Sharing Rules:** Opened access to department heads and placement officers.
- Ensures only authorized users can view/edit sensitive academic data.

5. Field Level Security (FLS)

- Configured field-level visibility for sensitive fields.
    - Example: GPA visible only to Faculty/Admin, hidden for students.
- Placement Status restricted to Placement Team.

6. Session Settings

- Session timeout reduced to **15 minutes** for security.
- Enabled **Require re-login after session timeout**.
- Prevented simultaneous logins from different devices.

7. Login IP Ranges

- Restricted login access for student profiles to **University Campus IP ranges**.
- Admin and Faculty profiles allowed wider access.

8. Audit Trail

- Setup Audit Trail enabled to track changes to:
    - Reports & Dashboards

    - Sharing Rules
    - Field-Level Security
- Ensured compliance and accountability.

Phase 10: Deployment & Go-Live

1. Change Sets & Metadata Deployment

- **Purpose:** Move customizations (Objects, Fields, Apex Classes, LWC, Reports, Dashboards) from **Sandbox → Production**.
- **Steps:**
    1. Create **Outbound Change Set** in Sandbox.
    2. Add components (Objects, Triggers, Classes, Reports, Dashboards).
    3. Upload to Production.
    4. In Production, go to **Inbound Change Set → Deploy**.

2. Pre-Deployment Checklist

- Data backup from Sandbox.
- Ensure all **Profiles & Permission Sets** match production roles.
- Validate Apex code & triggers in **Deployment Preview**.
- Run test classes → must achieve **75%+ code coverage**.

3. Post-Deployment Activities

- Verify data integrity (Student Records, GPA, Placement data).
- Test reports & dashboards with production data.
- Validate **Sharing & FLS rules** are working as expected.
- Ensure **Dynamic Dashboards** reflect user-specific views.

4. User Training & Documentation

- Conduct training for:
    - Faculty (using reports & dashboards).
    - Placement Team (tracking placement progress).
    - Admins (managing student data & security).

- Provide user manuals (screenshots + step-by-step usage).

5. Go-Live Support

- Enable **Login Hours & IP Restrictions** for student logins.
- Monitor **Login History** and **Audit Trail** in production.
- Keep a **Rollback Plan** (restore metadata or data if issues occur).

6. Post Go-Live Monitoring

- Monitor system performance.
- Collect feedback from users.
- Track adoption metrics (number of reports viewed, dashboards used).
- Schedule regular data backup.