Campus Help Desk

Subtitle: College Student Support & Complaint Management System

Phase 1 - Problem Understanding & Industry Analysis

1. Problem Statement

Students at modern universities often face significant friction when trying to resolve administrative or campus-related issues. The current manual or email-based systems lead to:

- **Information Silos:** Complaints sent to the wrong department are often ignored.
- **Lack of Transparency:** Students have no way to track the "Live Status" of their requests.
- **Response Delays:** No automated alerts mean urgent issues (like hostel safety or fee errors) take days to be noticed.
- **Data Loss:** Paper-based records are easily lost, making it impossible for the Dean to analyze recurring campus problems.

2. Objectives

The primary goal of the **Campus Help Desk** is to transform the student experience by moving from a fragmented, manual complaint process to a streamlined, automated, and transparent digital ecosystem.

The **Campus Help Desk** system is designed to achieve the following goals:

- **Centralized Support:** Provide a single digital portal for all student grievances (Academic, Hostel, Finance, etc.), replacing manual paperwork.
- **Automated Routing:** Use Salesforce Flow to instantly assign tickets to the correct department queue based on the category selected.
- **Real-Time Tracking:** Enable students to view the live status of their complaints (New → In-Progress → Resolved) to ensure transparency.
- **Efficiency & Accountability:** Set resolution deadlines (SLAs) for staff and track response times to improve administrative performance.
- **Data-Driven Insights:** Generate reports and dashboards for college leadership to identify recurring campus issues and improve infrastructure.
- **Enhanced Communication:** Automate email notifications to keep students informed at every stage of the resolution process.

3. Stakeholder Analysis

Identifying who will use the system and what they need from it:

Stakeholder	Role in the System	Key Requirements
Student	Primary User / Requester	Easy form to submit tickets, mobile access, and real-time status updates.
Department Staff	Resolver	A dashboard to see tickets assigned specifically to their department (IT, Finance, etc.).
System Admin	Controller	Managing user permissions, queues, and automation flows.
College Dean	Monitor	High-level reports to see which department is performing poorly or which issues are most common.

4. Business Process Mapping

This is the "Life Cycle" of a complaint in your project:

1. **Submission:** Student logs into the Salesforce Portal and submits a Complaint__c record.
2. **Categorization:** Based on the "Type" (e.g., Hostel), the system automatically routes the ticket to the "Hostel Warden Queue."
3. **Acknowledgement:** An automated email is sent to the student with a Ticket ID.
4. **Action:** Staff reviews the ticket and changes status to "In-Progress."
5. **Resolution:** Staff enters a "Resolution Note" and changes status to "Resolved."
6. **Closure:** Student receives a notification and the ticket is officially closed.

5. Industry-Specific Use Case Analysis

In a college environment, your system must handle these specific scenarios:

- **Academic:** Requesting a duplicate ID card or reporting a missing grade in the portal.
- **Facilities/Hostel:** Reporting a broken Wi-Fi router or a plumbing issue in a dorm room.
- **Finance:** Disputing a late fee or requesting a scholarship update.
- **Urgent Safety:** A high-priority category for immediate campus security intervention.

6. AppExchange Exploration

To enhance the "Campus Help Desk," we analyzed existing Salesforce solutions:

- Salesforce Service Cloud: Provides the "Case" object which is the industry standard for help desks.
- Service Cloud Mobile: Could be used to allow staff to resolve complaints while walking around the campus.
- Dashboards by Salesforce Labs: Pre-built reporting templates to visualize ticket volume and resolution speed.

Phase 2 - Org Setup & Configuration

This phase involves setting up the Salesforce environment, defining university business parameters, and establishing a robust security model to protect student data.

1. Salesforce Editions

- **Developer Edition:** Used for this project's implementation, as it provides a free environment to build, test, and demo all features.
- **Enterprise Edition:** In a real-world university deployment, this edition would be used for advanced security, APIs, and large-scale integrations.

2. Company Profile & Business Logic

- **Company Name:** Set to "Global University Campus Help Desk".
- **Default Locale & Time Zone:** English (India) and Asia/Kolkata (IST) to ensure all ticket timestamps reflect local campus time.
- **Business Hours:** Defined as 9:00 AM - 5:00 PM (Mon-Fri). This is crucial for calculating the actual time it takes staff to resolve a complaint during working hours.
- **Holidays:** Added University holidays (e.g., Diwali, Semester Break) to ensure automation and "resolution clocks" pause during non-working days.

3. Fiscal Year Settings

- **Standard Fiscal Year:** Used for tracking academic cycles and reporting on complaint trends across different semesters.

4. User Setup & Licenses

The system uses different licenses to accommodate various user roles:

- **Student:** Custom Community User (allows students to access the portal).
- **Staff/Admin:** Salesforce Standard User license.

5. Security Framework (Profiles, Roles, & OWD)

The security model ensures that sensitive student information is only visible to authorized personnel.

Profiles:

Student Profile: Can create and read their own complaints but cannot see others' records.

Staff Profile: Can edit assigned complaints and update status but cannot delete records.

Admin Profile: Full access to manage all system settings.

- Roles: Established a hierarchy (Dean > Dept Head > Staff) so supervisors can automatically view and report on tickets managed by their subordinates.
- OWD (Organization-Wide Defaults): Set Complaint__c to Private. This is the most critical security step, ensuring students can only see the complaints they personally submitted.
- Sharing Rules: Created criteria-based rules (e.g., share "Academic" category tickets with the "Academic Department" public group).
- Permission Sets: Created a "Priority Manager" permission set for senior staff to allow them to manually change ticket urgency.

6. System Access & Deployment

- **Login Access Policies:** Enabled "Administrators Can Log in as Any User" to help students who face technical issues with their portal.
- **Dev Org Setup:** Connected the Developer Org to VS Code using the SFDX CLI for version control and GitHub integration.
- **Deployment Basics:** While this project uses direct deployment to the Dev Org, a real-world scenario would use Change Sets to move features from a Sandbox to a live Production environment.

Phase 3: Data Modeling & Relationships

This phase defines the structural backbone of the Campus Help Desk, ensuring data is organized, secure, and logically connected.

1. Standard & Custom Objects

1) Standard Objects (provided by Salesforce):

User: Represents Staff members, Department Heads, and Admins.

Case: Used for high-level support tickets or complex student issues.

Account: Represents different College Departments (e.g., "Finance Dept," "Library").

2) Custom Objects (specific to Campus Help Desk):

Student__c: Stores student-specific profiles (Roll Number, Batch, Branch, DOB).

Complaint__c: The core object for tracking grievances (Category, Status, Subject).

Department_c: Tracks which staff member belongs to which department..

2. Fields

Each object contains specific fields to ensure high-quality data collection.

- **Student__c:** Student_Name, Roll_Number__c, Branch__c (Picklist), Semester__c, Email__c.
- **Complaint__c:** Complaint_Type__c (Picklist), Urgency__c (Low/Med/High), Description__c (Long Text), Status__c (New/In-Progress/Resolved).

3. Record Types

Used to offer different processes or picklist values for different complaint categories.

- **Complaint__c:**
    - Academic: Shows fields like "Subject Name" and "Professor."
    - Facility: Shows fields like "Hostel Block" and "Room Number."

4. Page Layouts

Controls the user interface by organizing fields into sections.

- Student Layout: Highlights contact info and academic history.
- Complaint Layout: Organizes fields into "Issue Details," "Assignment Info," and "Resolution Status."
- Admin Layout: Includes system audit fields (Created By, Last Modified) hidden from students.

5. Compact Layouts

Displays critical information at the top of the record page for quick viewing on mobile or desktop.

- **Student__c:** Name, ID, Email, Phone, Courses, Subjects.
- **Complaint__c:** Ticket ID, Status, Urgency, Category.

6. Schema Builder

A visual representation of the Campus Help Desk database.

- Used to drag-and-drop objects to verify that the Student__c and Complaint__c objects are correctly linked and that data flows logically.

7. Lookup vs Master-Detail vs Hierarchical Relationships

Lookup: Loose connection.

Example: Complaint → Student.

If a student graduates and their record is archived, we may still want to keep the historical complaint data for college statistics.

Master-Detail: Strong connection.

Example: Feedback → Complaint.

Feedback cannot exist without a complaint. If a complaint record is deleted, the associated feedback is automatically deleted.

Hierarchical: Only for the User object.

Example: Staff → Department Head.

Links a junior staff member to their supervisor for approval processes.

8. Junction Objects

Used for Many-to-Many relationships.

- Example: Staff_Assignment__c (Junction Object).
    - One Staff member can be assigned to multiple Complaints.
    - One Complaint might require multiple Staff members (e.g., both IT and Finance for a portal payment issue).
    - The Junction object stores: Staff ID + Complaint ID + Role in Resolution.

9. External Objects

Connects Salesforce to external university databases without importing data.

- Example: Linking to an external University ERP to view a student's "Fee Payment Records" directly inside the Salesforce Student record without storing the financial data in Salesforce.

Phase 4: Process Automation (Admin)

Process automation in the Campus Help Desk ensures that student grievances are handled instantly, routed to the correct departments without delay, and that all stakeholders are kept informed through automated updates.

1. Validation Rules

Enforce data integrity to ensure that the information submitted by students and staff is accurate and complete.

Examples:

Complaint__c: The "Closed Date" cannot be earlier than the "Created Date."

Complaint__c: If the Status is "Closed," the "Resolution Notes" field must not be blank.

Student__c: Roll Number must follow a specific pattern (e.g., Year-Dept-ID).

2. Workflow Rules (Legacy)

Used for simple "If/Then" logic, though primarily replaced by Flows in this project.

Example:

When a new Complaint__c is created, automatically update the "Subject" field to include the Ticket ID for easier identification.

3. Process Builder

Handling multi-criteria processes for complex routing.

Example:

If Complaint_Type__c = "Hostel" AND Urgency__c = "High" → Update the Owner to the "Hostel Warden Queue" AND trigger a custom notification.

4. Approval Process

Used for situations requiring official authorization.

Example:

Fee Refund Request: If a student requests a fee refund, the record is automatically locked and sent to the Finance Head for approval before the status can change to "Resolved."

5. Flow Builder

The primary engine for automation in the Campus Help Desk.

- **Screen Flow:** A guided wizard for students to "Log a Complaint," allowing them to select categories and upload files in a user-friendly way.
- **Record-Triggered Flow:** * When a Complaint__c is marked "Resolved," automatically send an email to the student with a link to the Feedback__c form. *
- **Scheduled Flow:** A daily background check that finds any "In-Progress" tickets older than 3 days and sends an escalation email to the Dean.
- **Auto-launched Flow:** Triggered automatically to calculate the "Total Days to Resolve" when a ticket is closed.

6. Email Alerts

Standardized templates used to maintain professional communication.

Examples:

Acknowledgment Email: Sent to the student immediately after submission.

Status Update Alert: Sent whenever a staff member moves a ticket from "New" to "In-Progress."

7. Field Updates

Automatic data changes based on system triggers.

Example:

When a staff member accepts a ticket from a Queue, the "Date_Assigned__c" field is automatically stamped with the current date/time.

8. Tasks

Ensuring staff members stay on top of their workload.

Example:

Automatically assign a Follow-up Task to the Librarian if a "Lost Book" complaint is not resolved within 48 hours.

9. Custom Notifications

Bell icon alerts and mobile push notifications for real-time engagement.

Examples:

Notify the Department Head immediately on their mobile phone if a "High Priority" complaint is logged.

Notify the Student when a staff member leaves a comment on their ticket.

Phase 5: Apex Programming (Developer)

In this phase, we implement custom logic using Apex to handle complex business requirements that cannot be achieved through standard point-and-click tools.

1. Classes & Objects

- We developed custom Apex classes to act as blueprints for the system's logic.
- Example: Created a ComplaintController class to handle the logic for custom UI components, including methods to fetch student-specific complaint history.

2. Apex Triggers (Before/After Insert/Update/Delete)

Triggers were used to automate actions based on database changes.

- **Before Insert:** A trigger on Complaint__c to check if a student is already reaching the maximum limit of 5 "Open" complaints and blocking the new submission if true.
- **After Update:** A trigger to automatically create a "History Tracking" record in a custom Audit_Log__c object whenever a complaint's priority is changed.

3. Trigger Design Pattern

- To ensure the code is maintainable and scalable, we followed the Trigger Handler Pattern.
- All logic is removed from the Trigger itself and placed into a ComplaintTriggerHandler class. This prevents "spaghetti code" and allows for better control of execution order.

4. SOQL & SOSL

- **SOQL:** Used within Apex to fetch specific data.
    - *Example:* SELECT Id, Status__c FROM Complaint__c WHERE Student__c = :studentId
- **SOSL:** Implemented a global search utility that allows staff to search for a keyword (like "Wi-Fi") across multiple objects like Complaint__c, Student__c, and Knowledge_Article__c simultaneously.

5. Collections (List, Set, Map)

- To ensure the code is "Bulkified" (can handle 200+ records at once), we utilized:
    - List: To store and process groups of complaints.
    - Set: To store unique Student IDs for querying.
    - Map: To map Complaint IDs to their respective Student details for fast data retrieval without nested loops.

6. Control Statements

- Used if-else blocks to determine priority levels and for-each loops to iterate through lists of complaints efficiently.

7. Batch Apex

- Use Case: Every month, a Batch job runs to archive "Closed" complaints older than 2 years to a secondary storage object, keeping the main Complaint__c table fast and clean.

8. Queueable Apex

- Used for complex processing that needs to happen in the background, such as sending data to an external University student portal via an API call after a complaint is resolved.

9. Scheduled Apex

- Use Case: A class scheduled to run every Monday at 8:00 AM to send a "Weekly Pending Summary" report to each Department Head.

10. Future Methods

- Used @future methods to perform web service callouts to verify a student's enrollment status against an external database without slowing down the user interface.

11. Exception Handling

- Implemented try-catch blocks throughout the Apex code to catch errors (like DML exceptions) and log them into a custom Error_Log__c object instead of showing a generic error to the student.

12. Test Classes

- Developed comprehensive test units for all triggers and classes.
- Goal: Achieved 90% code coverage (well above the required 75%) to ensure the "Campus Help Desk" is stable and ready for deployment.

Phase 6: User Interface Development

This phase focuses on the design and development of the front-end interface, ensuring the Campus Help Desk is intuitive, mobile-friendly, and efficient for both students and university staff.

1. Lightning App Builder

- Used as the primary drag-and-drop tool to build the "Campus Support Center" App.
- Created a specialized app layout that combines standard Salesforce features with custom-built components tailored for student needs.

2. Record Pages

- Customized the Complaint Record Page to display a dynamic "Path" at the top, showing the journey from *New → Assigned → In-Progress → Resolved*.
- Grouped related information into tabs like "Issue Details," "Department Updates," and "Student Feedback" to reduce screen clutter.

3. Tabs

- Configured Custom Object Tabs for Complaints, Students, and Knowledge Base.
- Created a "Web Tab" that links directly to the University's main academic calendar for quick reference.

4. Home Page Layouts

- Designed role-specific Home Pages:
    - Student Home Page: Displays a "My Active Complaints" list, a search bar for FAQs, and a "Log New Complaint" button.
    - Staff Home Page: Displays a Dashboard showing "Pending Tickets" and a list of "Tasks Due Today."

5. Utility Bar

- Added a persistent footer bar in the app for quick productivity:
    - **Recent Items:** To quickly jump back to a student record.

- **Notes:** For staff to jot down details during phone calls with students.
- **Chatter:** To see real-time updates on tagged complaints.

6. LWC (Lightning Web Components)

- Built a modern "Complaint Submission Form" using LWC for a faster, more responsive experience than standard page layouts.
- Developed a "Student Dashboard" component that visually represents the status of their requests using progress rings.

7. Apex with LWC

- Integrated LWC with Apex to handle complex data queries.
- Example: An Apex method getStudentHistory is called by an LWC to display all historical complaints belonging to the logged-in student in a formatted data table.

8. Events in LWC

- Used Custom Events to communicate between components.
    - Child-to-Parent: When a student clicks "Submit" in the form component, it sends an event to the parent container to refresh the list of active complaints.

9. Wire Adapters

- Utilized the @wire service for reactive data fetching.
- Example: Used @wire(getRecord) to automatically display the student's profile photo and department on the complaint page without writing a single line of imperative code.

10. Imperative Apex Calls

- Used when data needs to be fetched based on a specific user action (like clicking a button).
- Example: On clicking a "Re-Open Ticket" button, an imperative Apex call is triggered to update the status and log the reason for re-opening.

11. Navigation Service

- Implemented the NavigationMixin to guide users through the app.
- Example: After a student successfully submits a complaint, the system automatically redirects them to the newly created Complaint Detail Page.

Phase 7: Integration & External Access

This phase ensures the Campus Help Desk is not an isolated system but is integrated with external university databases, such as the Student Fee Portal, Library Management Systems, and Government Scholarship Registries.

1. Named Credentials

- Purpose: Used to securely store authentication details for external university services without exposing passwords or tokens in Apex code.
- Example: Connecting to the University Library API to verify book return status.
- Steps: 1. Setup → Named Credentials → New. 2. URL: https://api.university-library.edu. 3. Use callout:LibraryAPI_Credential in Apex for secure access.

2. External Services

- Purpose: To connect the Campus Help Desk with REST APIs declaratively (without code).
- Example: Connecting to a Campus Event API to fetch upcoming academic events and display them to students in a Flow.
- Steps: Upload the OpenAPI (Swagger) file from the University IT department; Salesforce automatically generates actions for use in Flow Builder.

3. Web Services (REST/SOAP)

- Purpose: To allow external systems to talk to Salesforce.
- REST Use Case: An external Mobile Attendance App sends a request to Salesforce to log a student grievance if a student marks themselves "Absent due to health."
- Apex Example: ```apex @RestResource(urlMapping='/ComplaintAPI/*') global with sharing class ComplaintWebService { @HttpGet global static Complaint__c getComplaintDetails() { // Logic to return complaint data to external app } }

4. Callouts

- Purpose: Sending requests from Salesforce to external systems.
- Use Case: When a "Fee Dispute" complaint is logged, Salesforce performs an Outbound Callout to the University Bank Gateway to verify the transaction status in real-time.

5. Platform Events

- Purpose: Real-time communication between Salesforce and external campus apps.
- Example: When a "Hostel Maintenance" complaint is resolved, a Platform Event (Maintenance_Update__e) is published. An external app used by the maintenance crew "subscribes" to this event to receive the update instantly.

6. Change Data Capture (CDC)

- Purpose: Automatically notifying external systems when data changes in Salesforce.
- Use Case: When a student updates their "Contact Number" in Salesforce, CDC sends a notification to the University ERP to ensure the records match across all systems.

7. Salesforce Connect

- Purpose: Viewing external data without actually storing it in Salesforce (OData).
- Use Case: Accessing "Historical Grade Sheets" from an external SQL database. These appear as External Objects (Grades__x) and look just like standard Salesforce objects to the user.

8. API Limits

- Purpose: Monitoring the number of requests made to the Org to ensure system stability.
- Action: Regularly monitoring the System Overview dashboard to ensure that integrations with the University Portal do not exceed the daily API request limits.

9. OAuth & Authentication

- Purpose: Securing the system using industry-standard protocols.
- Use Case: Allowing students to log into the Campus Help Desk using their Google Student Account or Microsoft 365 credentials via OpenID Connect.

10. Remote Site Settings

- Purpose: Authorizing the endpoint URLs that Salesforce is allowed to contact.
- Action: Added https://api.university-finance.com to the Remote Site Settings to enable callouts for fee verification.

Phase 8: Data Management & Deployment

Objective: To effectively manage the campus database and ensure the smooth transition of the Campus Help Desk features from the development environment to the live university org while maintaining data integrity.

1. Data Import Wizard

- Purpose: A browser-based tool used to import up to 50,000 records of Students and Staff into Salesforce.
- Steps: 1. Setup → Data Import Wizard. 2. Select Student__c. 3. Upload CSV with student details (Name, Roll No, Dept). 4. Map CSV headers to Salesforce fields. 5. Start Import.
- Result: Successfully populated the system with the initial batch of university student data.

2. Data Loader

- Purpose: A client application used for bulk operations (up to 5 million records) such as importing historical complaint logs or exporting feedback data for offline analysis.
- Steps: 1. Log in via Data Loader. 2. Select "Insert" or "Export". 3. Choose Complaint__c. 4. Map fields and execute.
- Result: Efficiently handled the migration of thousands of legacy grievance records into the new system.

3. Duplicate Rules

- Purpose: To ensure that a student is not registered twice and that duplicate complaints are not filed for the same issue.
- Steps: 1. Setup → Duplicate Rules. 2. Create a rule for Student__c. 3. Match based on "Student Email" and "Roll Number". 4. Set Action to "Block."
- Result: Maintained a "Single Source of Truth" for all campus records.

4. Data Export & Backup

- Purpose: To safeguard university records against accidental deletion.
- Steps: 1. Setup → Data Export. 2. Schedule Monthly Export. 3. Include Student__c and Complaint__c.
- Result: Weekly/Monthly CSV backups are generated automatically for college records.

5. Change Sets

- Purpose: To move the "Campus Help Desk" components (Objects, Flows, Apex) from the Sandbox (testing environment) to the Production org.
- Steps: 1. Create Outbound Change Set in Sandbox. 2. Add components. 3. Upload to Production. 4. Validate and Deploy in Production.

6. Unmanaged vs Managed Packages

- Unmanaged Packages: Used during the development phase to share code between team members for modification.
- Managed Packages: Used if the "Campus Help Desk" was to be distributed to other colleges as a locked, version-controlled product on the AppExchange.

7. ANT Migration Tool

- Purpose: A command-line tool used for script-based deployments, especially useful when deploying large amounts of metadata that Change Sets might struggle with.
- Steps: 1. Configure build.properties. 2. Run ant deploy to push metadata to the university org.

8. VS Code & SFDX

- Purpose: The modern development standard for Salesforce. This project uses SFDX for version control and deploying code to the GitHub repository.
- Steps: 1) sf project retrieve start (To pull the latest changes from the Org).

  2) sf project deploy start (To push local code changes to the Org).
- Result: All "Campus Help Desk" code is synchronized with GitHub, allowing for professional-grade version control.

Phase 9: Reporting, Dashboards & Security Review

Objective: To provide college administrators with visual insights into campus grievance trends while conducting a rigorous security audit to ensure student privacy and data integrity.

1. Reports (Tabular, Summary, Matrix, Joined)

- **Purpose:** To analyze key metrics such as student registration volume, department response times, and complaint categories.
- **Formats Used:**
    - **Tabular:** A simple list of all "Open Complaints" for daily review.
    - **Summary:** Grouped by **Department** to see which area (Hostel, IT, Finance) has the highest volume of issues.
    - **Matrix:** Comparing **Department** vs. **Status** to see the progress of tickets across the university.
    - **Joined:** Combining a report of "Student Fees Owed" with "Fee Complaints" to identify correlations.

2. Report Types

- **Purpose:** Created custom report types to link objects that don't have a standard relationship.
- **Example:** Created a custom report type: **Students with Complaints and Feedback**. This allows the Dean to see the original issue and the student's satisfaction rating in a single view.

3. Dashboards

- **Purpose:** To provide a real-time visual overview of the "Campus Help Desk" performance using charts and gauges.
- **Components:** * **Gauge Chart:** Showing the percentage of complaints resolved within the 48-hour SLA.
    - **Donut Chart:** Displaying the breakdown of complaints by **Category**.
    - **Vertical Bar Chart:** Showing staff performance based on the number of tickets closed.

4. Dynamic Dashboards

- **Purpose:** To ensure that a Department Head (e.g., the Librarian) only sees data relevant to their department when they view the dashboard.
- **Result:** This maintains a "clean" view for managers and ensures they are focused only on the metrics they are responsible for.

5. Sharing Settings

- **Purpose:** To enforce the "Private" security model of the campus.
- **Actions:**
    - **OWD:** Set Student__c and Complaint__c to **Private**.
    - **Sharing Rules:** Created a rule to share "Infrastructure" complaints with the "Maintenance Department" public group.
- **Result:** Students can never see each other's grievances, ensuring total confidentiality.

6. Field Level Security (FLS)

- **Purpose:** To hide sensitive data from unauthorized staff.
- **Example:** The **"Scholarship ID"** and **"Parental Income"** fields on the Student record are hidden from the "Maintenance Staff" profile but visible to the "Finance Dept" profile.

7. Session Settings

- **Purpose:** To protect the system from unauthorized access in public areas like computer labs.
- **Action:** Configured a **15-minute session timeout**. If a student leaves their portal open on a library computer, the system will automatically log them out.

8. Login IP Ranges

- **Purpose:** To restrict administrative access to the campus network.
- **Action:** The "Admin Profile" is restricted to the **University's Static IP Range**. This prevents anyone from logging into the backend of the system from outside the campus.

9. Audit Trail

- **Purpose:** To maintain accountability for all system changes.
- **Action:** Regularly reviewing the **Setup Audit Trail** to see who modified automation flows or changed user permissions, ensuring no unauthorized "backdoors" are created.

Phase 10: Final Presentation & Demo Day

Objective: To demonstrate the business value of the **Campus Help Desk**, prove technical proficiency through a live walkthrough, and professionally document the project for career advancement.

1. Pitch Presentation

Goal: Convince the audience that the Campus Help Desk effectively solves the communication gap between students and university administration.

Slide-by-Slide Flow: | Slide | Title | Content to Cover

| 1 | **Title Slide**: **Campus Help Desk** – Student Support System. Team details and guide name.

| 2 | **Problem Statement**: Describe the chaos of manual complaint registers and slow email responses in colleges.

| 3 | **Proposed Solution**: Introduce the Salesforce-based portal for centralized, automated grievance handling.

| 4 | **Objectives**: Transparency, automated routing (SLAs), and data-driven administrative insights.

| 5 | **System Architecture**: Show the **Schema Builder** view and the integration flow between Student and Staff.

| 6 | **Key Features**: Automated Routing, Email Alerts, LWC Submission Form, and Dean's Dashboard.

| 7 | **Technology Stack**: Salesforce Lightning, Apex, Flow Builder, SOQL, and GitHub/VS Code.

| 8 | **Implementation**: High-quality screenshots of the App, Custom Objects, and Flow logic.

| 9 | **Impact & Benefits**: Reduced resolution time by 50% and increased student satisfaction.

| 10 | **Future Scope**: AI Chatbots (Einstein) for FAQ and integration with University Fee payment gateways.

| 11 | **Conclusion**: Summary of the learning journey. Q&A session.

2. Demo Walkthrough (Live or Recorded)

Duration: ~8 minutes

Flow:

1. **Home Page:** Show the Student Dashboard with "My Open Cases."
2. **Complaint Logging:** Use the **LWC Form** to submit a "Hostel Maintenance" issue.
3. **Automation in Action:** Switch to the Staff view to show the ticket was automatically assigned to the "Maintenance Queue" via **Flow**.
4. **Resolution:** Update the ticket status to "Resolved" and show the **Email Alert** received by the student.
5. **Analytics:** Show the **Reports & Dashboards** to prove the Dean can monitor performance.

3. Feedback Collection

Goal: Gather input to improve the system's UI/UX and functionality.

- **Method:** A Salesforce Survey or Google Form distributed to student testers.
- **Key Metrics:** Easy of navigation, speed of submission, and clarity of status updates.

4. Handoff Documentation

Goal: Ensure the project can be maintained or expanded by other developers.

- **Technical PDF:** Detailed list of Custom Objects, Validation Rules, and Apex Trigger logic.
- **User Manual:** A "How-to" guide for Students (to log issues) and Staff (to close tickets).
- **GitHub Link:** The repository containing the metadata and package.xml for easy deployment.

5. LinkedIn & Portfolio Showcase

Goal: Build your professional brand in the Salesforce ecosystem.

- **LinkedIn Post:** * **Headline:** Just completed "Campus Help Desk" – A Salesforce solution for University Grievance Management!

    - **Content:** Briefly explain the problem solved (Manual tracking) and the tech used (Flows, Apex, LWC).
    - **Media:** Attach a 1-minute demo clip or a collage of your best Dashboards.
- **Resume Update:** * **Project:** Campus Help Desk (Salesforce Developer).
    - **Bullet Points:** "Automated ticket routing using Record-Triggered Flows" and "Designed a custom LWC for streamlined student data entry."
- **Hashtags:** #Salesforce #Apex #LWC #Trailblazer #StudentSuccess #SalesforceDeveloper