Leave Tracker App – Employee Leave Management CRM

- Industry: Human Resources / Workforce Management
- Project Type: B2C Salesforce CRM Implementation
- Target Users: Employees, Managers / Supervisors HR Administrators.

Problem Statement

In many organizations, leave management is still handled manually using spreadsheets, emails, or paper requests. This creates multiple challenges:

- Employees lack visibility into their leave status and approval progress.
- Managers often miss or delay approvals due to lack of reminders.
- HR teams find it difficult to track leave balances, trends, and records centrally.
- Communication regarding approvals or rejections is not automated, leading to confusion.
- There is no single system where employees can view their past or pending leave requests.

The company wants a Salesforce CRM solution to:

- Provide employees with a self-service portal to raise leave requests.
- Automate the approval/rejection process with manager comments.
- Send real-time notifications (Email/SMS) to employees on status updates.
- Maintain a centralized leave history for compliance and reporting.
- Enable HR and managers to analyze leave trends and ensure workforce availability.

Use Cases

1. Employee Leave Requests

- Employees can submit new leave requests with start date, end date, and reason.
- Requests are automatically linked to the requesting employee.

2. Manager Review & Approval

- Managers can view pending requests, approve or reject them, and add comments.
- Approved/Rejected status is updated instantly in the system.

3. Automated Notifications

- On approval → Email sent to employee: “Your leave has been approved from Salesforce.”
- On rejection → Email sent to employee: “Your leave has been rejected from Salesforce.”

4. Leave History Tracking

- Employees can view their previous and current requests.
- Managers and HR can monitor leave patterns and availability.

5. Reporting & Dashboards

- Reports: Leaves by Department, Employee, or Status.
- Dashboards: Pending vs Approved Requests, Monthly Leave Trends.

Phase 1: Problem Understanding & Industry Analysis

1. Problem Statement

In many organizations, managing employee leave requests is still a manual or semi-automated process. Employees rely on emails, spreadsheets, or informal communication with managers to apply for leave. This leads to challenges such as:

- Lack of centralized leave records.
- Delays in leave approvals due to manual follow-ups.
- No visibility for managers into team leave schedules, affecting workforce planning.
- HR teams spending extra effort reconciling attendance, payroll, and leave balances.
- The Leave Tracker Project aims to provide a Salesforce-based CRM solution to streamline leave management, ensure transparency, and improve productivity.

2. Requirement Gathering

Functional Requirements

- Employees should be able to submit leave requests through a self-service portal.
- Managers should be able to review, approve, or reject leave requests.
- Employees should get notifications (email/SMS) when their leave is approved or rejected.
- System should track leave balances (e.g., sick leave, casual leave, earned leave).
- HR/Admin should have reports on leave utilization, employee trends, and pending approvals.
- Integration with employee data for role-based access.

Non-Functional Requirements

- Data Security: Only employees and managers can access their relevant data.
- Performance: Leave requests and approvals should be processed instantly.
- Usability: Simple interface for employees, managers, and HR.
- Scalability: Supports organizations of different sizes (small teams to large enterprises).

3. Business Process Mapping

- Step 1: Employee logs in and submits a leave request.
- Step 2: System checks leave balance and policy compliance.

- Step 3: Leave request is routed to the reporting manager.
- Step 4: Manager approves/rejects the request.
- Step 5: Employee gets notified (Email/SMS).
- Step 6: HR/Admin dashboard updates automatically with status and leave balance.

4. Industry-Specific Use Case Analysis

- Corporate/IT Companies: Streamlined leave approval process avoids project delays.
- Educational Institutions: Staff and faculty leave can be centrally managed.
- Manufacturing/Retail: Shift planning and workforce availability are improved.
- Technology/CRM Industry: Salesforce provides automation (Flows, Approval Processes) and reporting for HR operations.
- This project combines HR Tech + CRM automation use cases.

5. AppExchange Exploration

Before custom development, we explored Salesforce AppExchange to check for existing solutions:

- HR Management Apps: Mostly focus on payroll and performance management.
- Attendance/Leave Apps: Available but often too generic, expensive, or not customizable.
- SMB Solutions: Not scalable for enterprise-level needs.

– Conclusion: Existing apps do not fully align with the organization's specific leave policies and reporting needs. A custom Salesforce solution is required.

6. Phase 1 Outcomes

Problem clearly defined: Manual and inefficient leave management process.

- Requirements documented (functional + non-functional).
- Stakeholders identified and analyzed.
- Business process mapped for end-to-end leave cycle.
- Industry gap validated: AppExchange does not offer an exact fit.

Phase 2: Org Setup & Configuration

1. Salesforce Editions

For the Leave Tracker Project, the chosen edition is Salesforce Enterprise Edition, as it provides:

- Support for complex business processes like approval workflows and automation.
- Ability to create custom objects (Leave Request, Leave Balance).
- Role hierarchy and advanced sharing settings.

Company Profile Setup

- Organization Name: Leave Tracker (example for your project).
- Default Time Zone: IST (India Standard Time) to match employee location.
- Default Currency: INR (₹) for leave-related reporting (if linked with payroll).
- Language Settings: English (default), with multilingual support possible if required.

User Setup & Licenses

1. Users Created:

- Employees (end users submitting leave requests).
- Managers (approvers).
- HR/Admin (monitoring, reporting).

2. Licenses Assigned:

- Salesforce Platform License → For Employees.
- Salesforce License → For Managers & Admins (to enable reporting and advanced features).

Profiles, Roles, Permission Sets

1. Profiles:

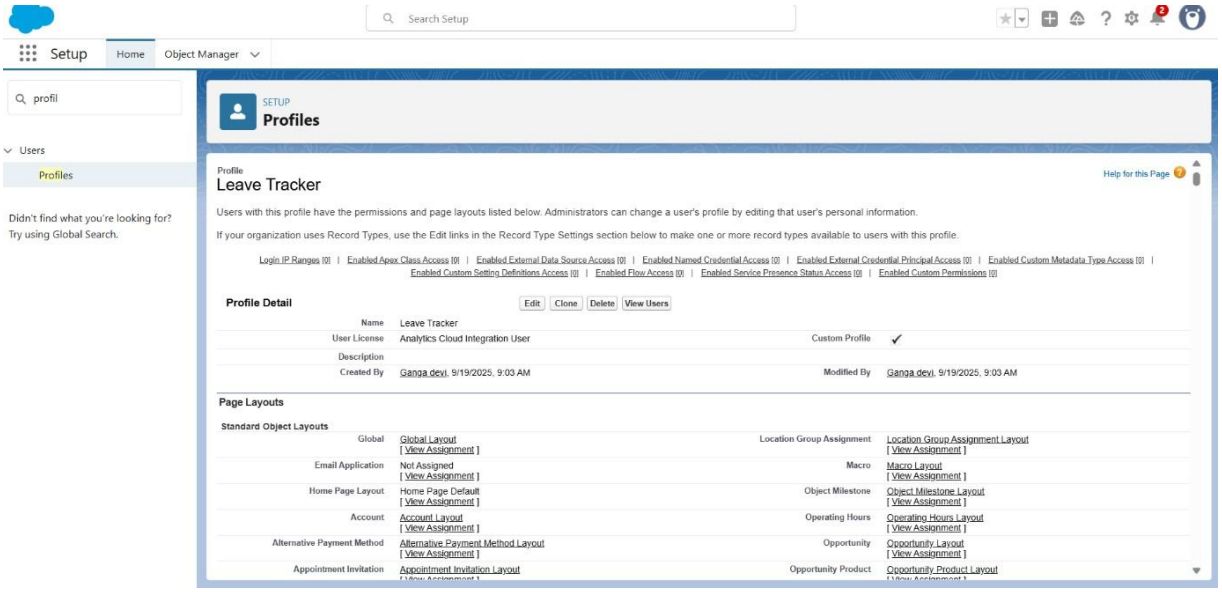
- Employee Profile → Access to submit leave requests only.
- Manager Profile → Approve/reject requests, view team records.
- HR/Admin Profile → Full access, reporting, system configurations.

2. Roles:

- Employee Role → Reports to Manager Role.
- Manager Role → Reports to HR Role.
- HR Role → Top-level role with organization-wide visibility.

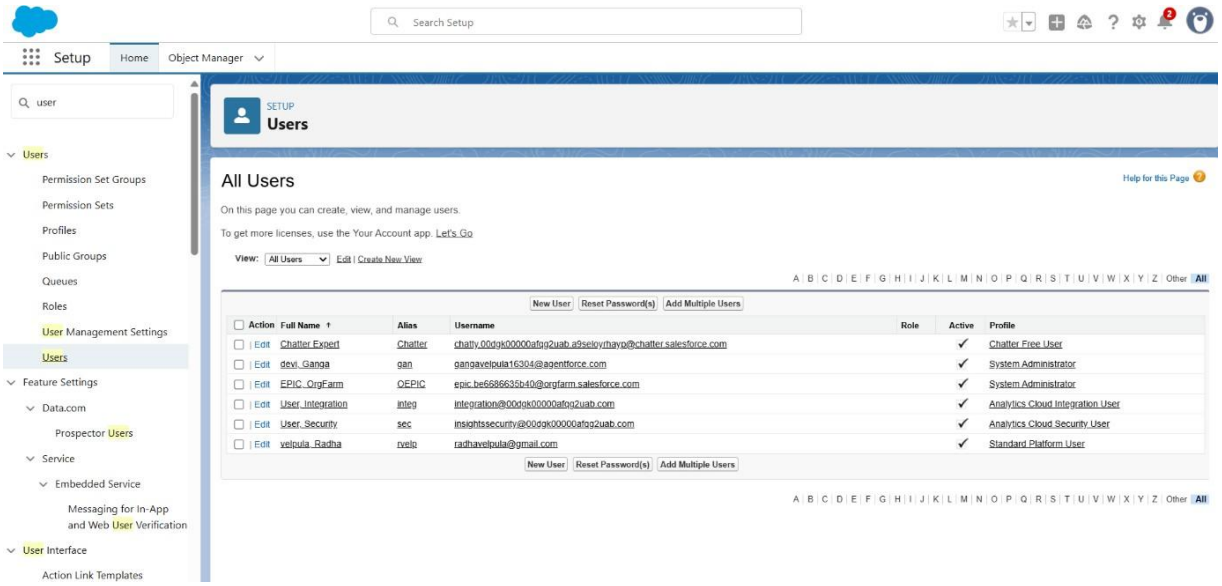
3. Permission Sets:

- Notification Access (Email/SMS integration).
- Report & Dashboard Access.



4. Users:

- Used to test the profiles, Roles, permission sets.
- The user is used to edit the leave request to accepted , Rejected.



OWD & Sharing Rules

1. Object-Level Security (OWD):

- Leave Requests → Private (employees see only their own requests).
- Leave Balance → Private.
- Reports/Dashboards → Controlled by HR.

2. Sharing Rules:

- Manager can see leave requests of their direct reports.
- HR/Admin has access to all employee records.

Deployment Basics

- Change Sets: Used to deploy metadata (custom objects, fields, flows, reports) from Sandbox to Production.
- Version Control (GitHub): Repository maintained for tracking changes.
- Deployment Checklist:
- Validate test classes (minimum 75% coverage).
- Backup existing metadata.
- Deploy in non-peak hours.

🟩 Phase 2 Outcome:

- Org environment is fully set up with business hours, users, roles, and permissions.
- Security policies, login access, and data visibility configured.
- Deployment process established for smooth release cycles.

Phase 3: Data Modeling & Relationships

1. Standard & Custom Objects

Standard Objects Used:

- LeaveRequest__c – The main object that stores all employee leave applications.

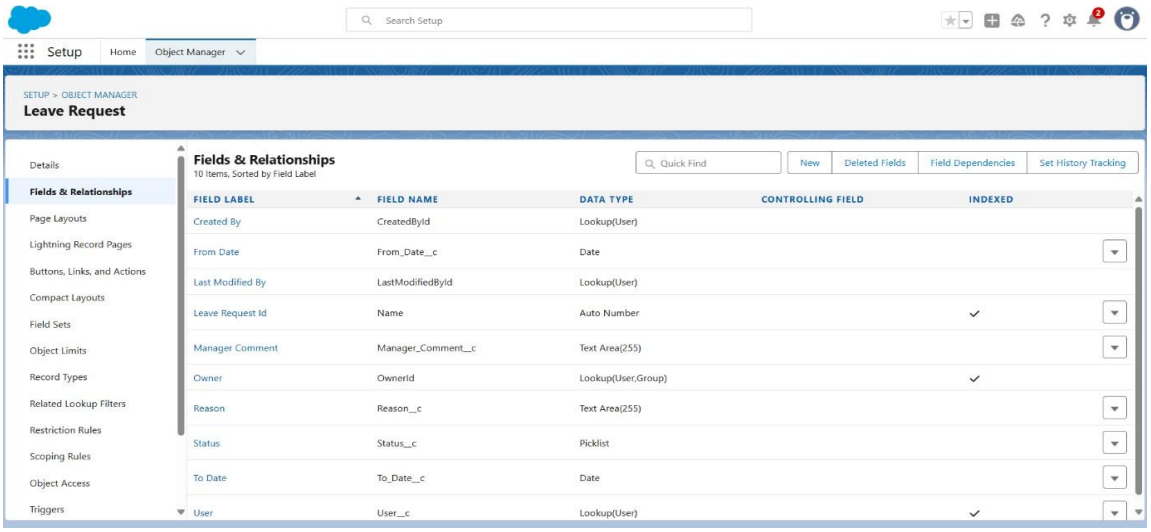
Custom Objects Created:

- Leave Request → Stores each leave application.
- Holiday Calendar → Stores organization-wide holidays.
- Approval History → Records manager approvals/rejections for audit purposes.

2. Fields

Leave Request Object Fields:

- Employee (Lookup to User)
- Leave Type (Picklist: Sick, Casual, Earned, Maternity, etc.)
- From Date (Date)
- To Date (Date)
- Total Days (Formula: To Date – From Date – Holidays)
- Reason (Text Area)
- Status (Picklist: Draft, Submitted, Approved, Rejected, Cancelled)
- Manager Comments (Long Text)

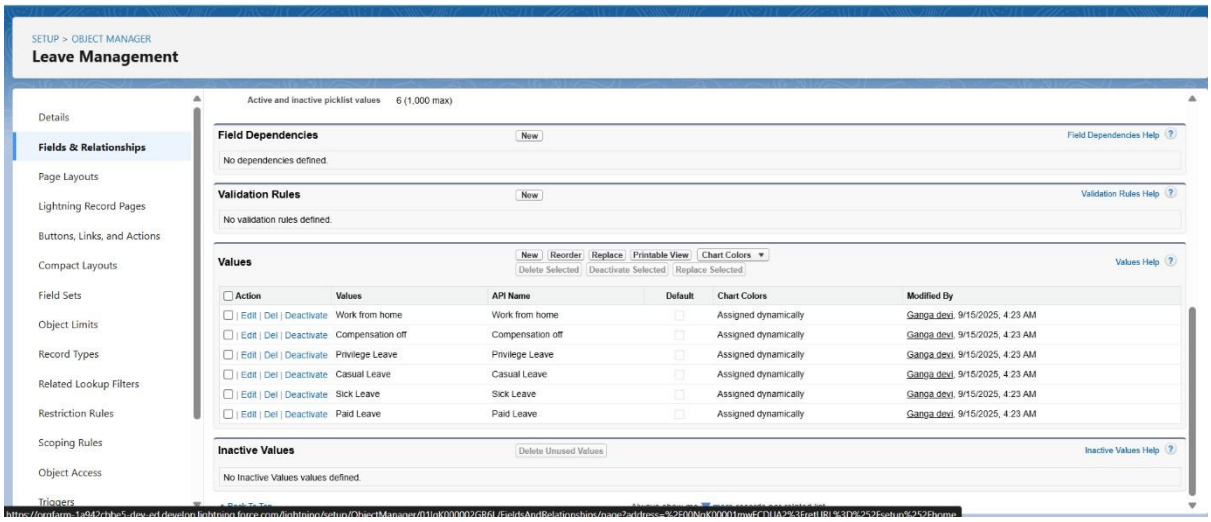


-

3. Record Types

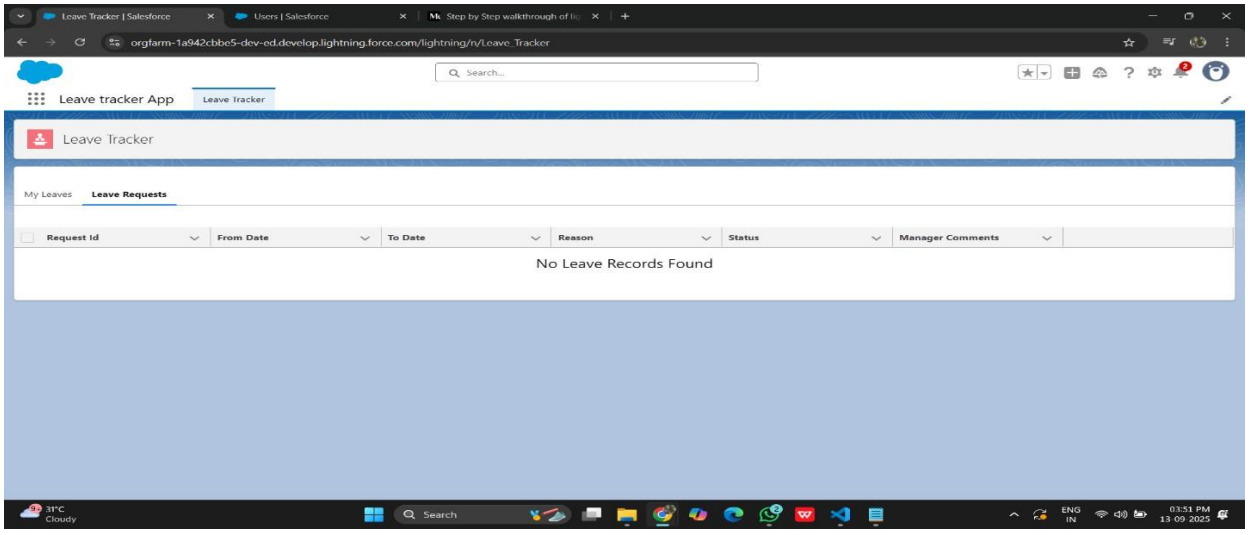
Leave Request Record Types:

- Sick Leave Request
- Casual Leave Request
- Earned Leave Request
- Special Leave Request (e.g., Maternity, Paternity)
- Each record type has different page layouts and approval rules.

- 

4. Page Layouts

- Employee Layout (Leave Request): Fields: From Date, To Date, Reason, Leave Type.
- Manager Layout (Leave Request): Includes additional fields: Status, Manager Comments.



Compact Layouts

- For Leave Request, compact layout shows: Leave Type, Dates, Status.

5. Relationships

Lookup Relationship:

- Leave Request → User (Employee).
- Leave Balance → User.

Master-Detail Relationship:

- Approval History → Leave Request (deletes when parent request is deleted).

🟩 Phase 3 Outcome:

- Data model created with required custom objects and fields.
- Relationships established between employees, leave requests.
- Record types, page layouts, and compact layouts designed for different users.

Phase 4: Process Automation

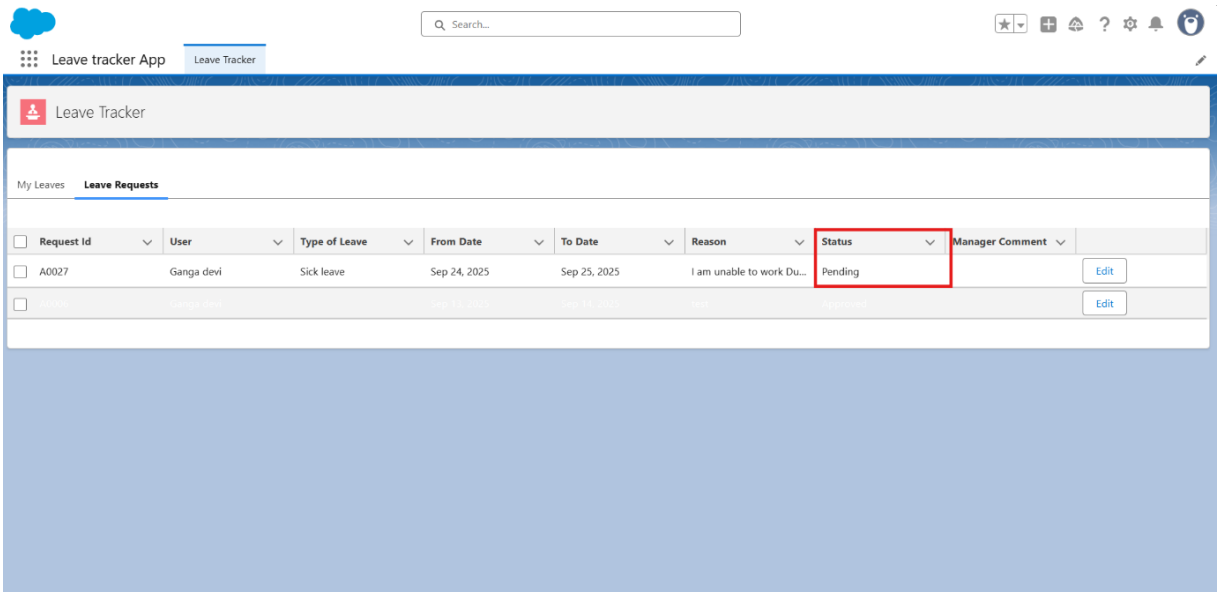
1. Validation Rules

- Ensures data quality and prevents invalid submissions.
- End Date must be greater than Start Date
    - End_Date__c < Start_Date__c
- Displays error message: "End Date must be later than Start Date."

Reason Required for Submitted Requests Ensures employees provide a reason before submitting.

2. Workflow Rules

- Auto-update Status: When a leave request is submitted, set status to “Pending Approval.”

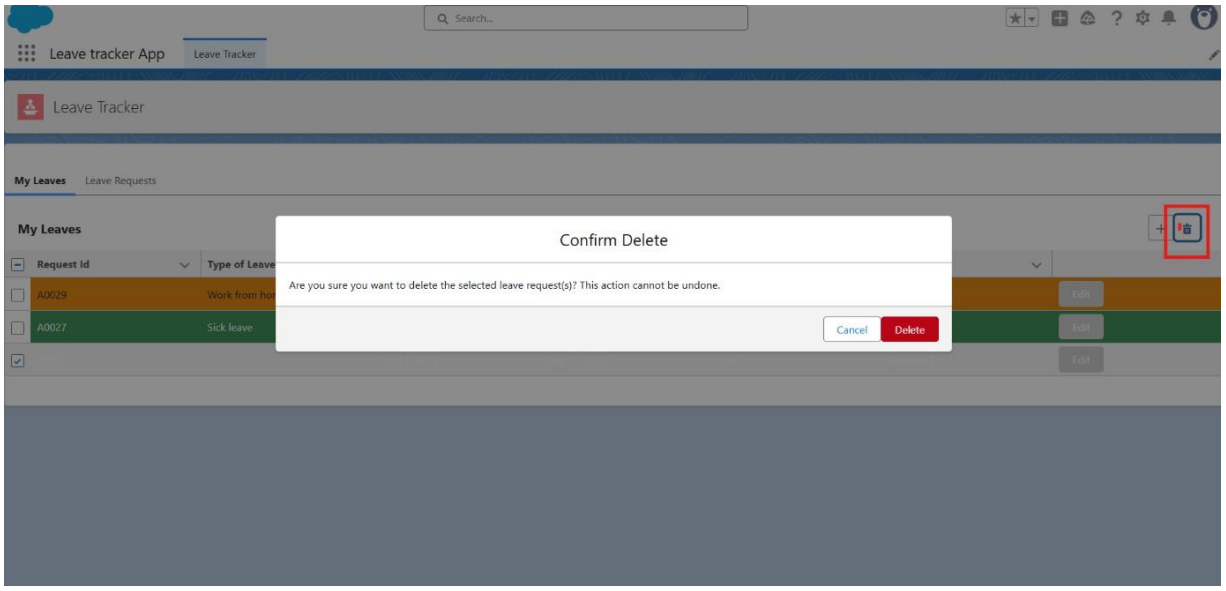


Actions:

- Send email notification to the Manager.
- Update Status field to "Submitted".

Deletion:

- Top pop warning will be avaliable to delete the request to Employee.



3. Process Builder

On Leave Request Submission:

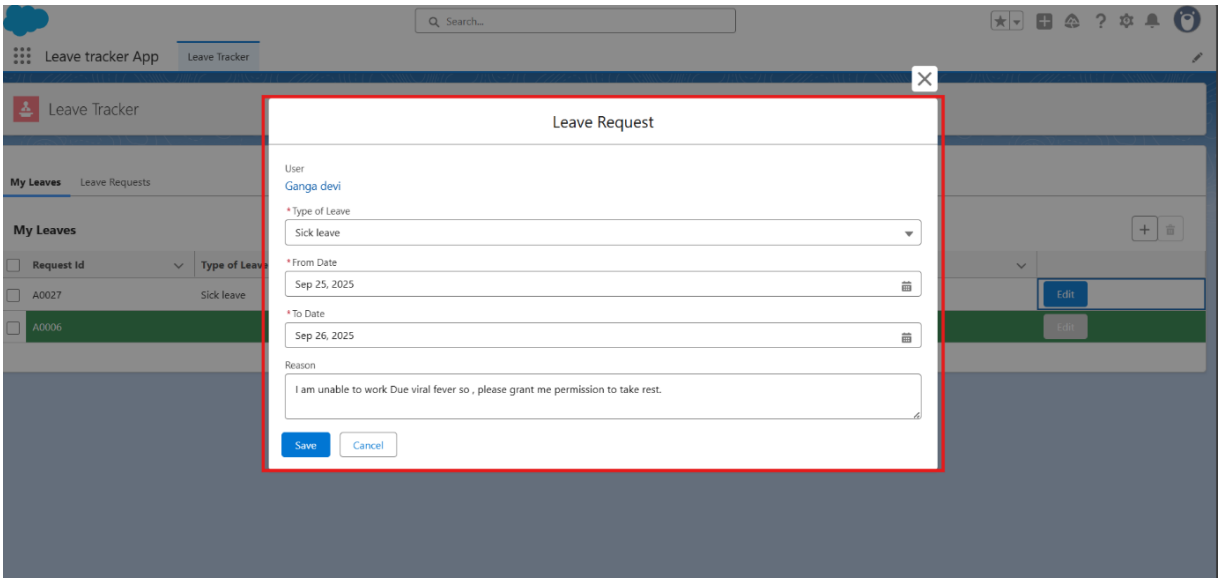
- Process Builder is used for conditional automation that cannot be handled by simple workflow rules. Example:
- If Status = Approved → Trigger email to employee confirming approval.
- If Status = Rejected → Trigger email with rejection reason.

4. Flow Builder

Screen Flows (for employees):

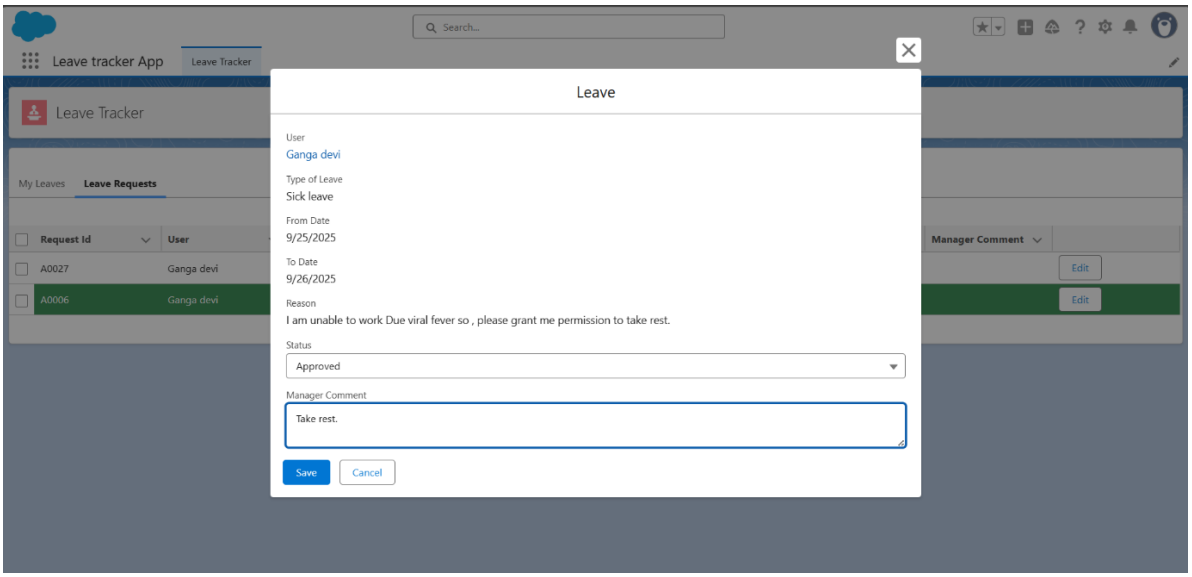
Guided process to apply for leave.

- Step 1: Select Leave Type.
- Step 2: Pick Dates.
- Step 3: Provide Reason.
- Step 4: Review & Submit.



5. Approval Process

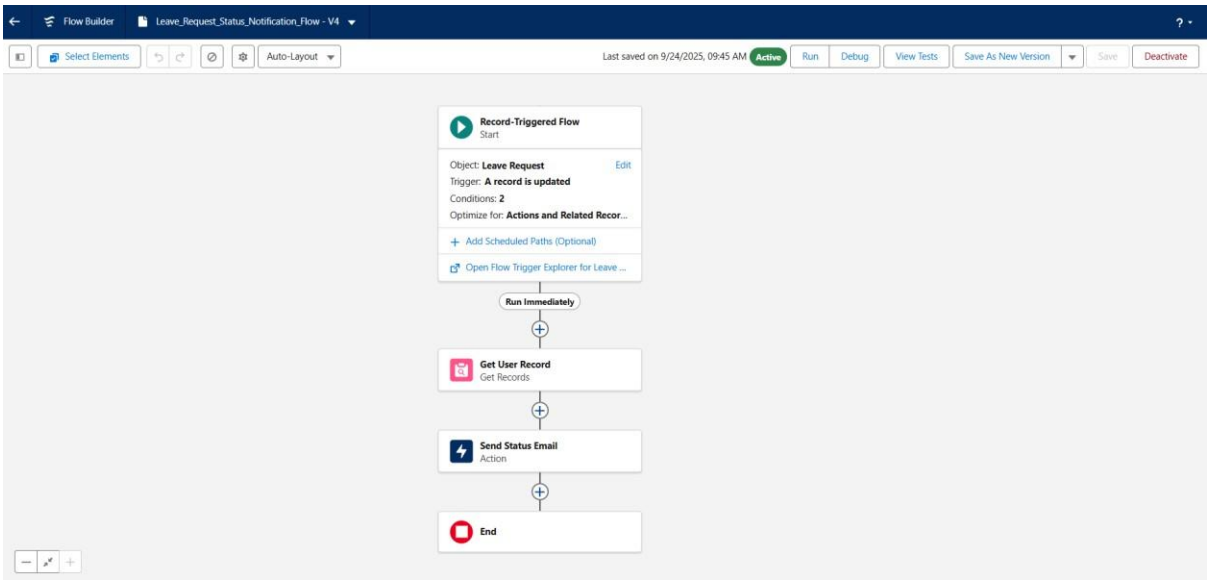
- Step 1: Employee submits request → Status = Pending.
- Step 2: Manager receives approval request.
- Step 3: Manager approves or rejects.
- Step 4: If approved → Send email to employee.
- Step 5: If rejected → Send email to employee.



6. Email Flows & Alerts

A. Email Flow

- Click Setup ()→ Quick Find→ Flows New Flow.
- Select Record-Triggered Flow.
- Choose Object: Leave_Request_Status_Notification → Trigger: When record is created→ After Save.
- Add Action Send Email Select Welcome Email Template.
- Save and Activate.



B. Email Alerts

- Leave Request Submitted: Sent to Manager.
- Leave Request Approved: Sent to Employee.
- Leave Request Rejected: Sent to Employee.
- Leave Balance Reminder: Monthly email to Employee.

7. Field Updates

- On Approval: Status → Approved.
- On Rejection: Status → Rejected.

- On Cancellation: Status → Cancelled.

8. Tasks & Custom Notifications

- Task: Auto-created for managers to review requests.
- Custom Notification (via Salesforce App): Push notification sent when a new leave request is assigned to a manager.

🟩 Phase 4 Outcome:

- All leave management workflows automated (submission, approval, rejection, cancellation).
- Employees & managers receive real-time notifications.
- HR/Admin has complete visibility with fewer manual interventions.

Phase 5: Apex Programming (Developer)

1. Classes & Objects

- LeaveRequestController: Handles leave application logic (submit, update, approve, reject).
- EmailNotificationService: Sends approval/rejection notifications.
- Utility Classes: For reusable logic like date validation, string formatting, and error handling.

2. Apex Triggers

Before Insert/Update:

- Validate leave dates (From_Date ≤ To_Date).
- Prevent overlapping leave requests.

After Insert:

- Notify manager of new leave request submission.

After Update:

- If Status = Approved → Send Approved Email
- If Status = Rejected → Send rejection email.
- If Status = Cancelled → Restore leave balance.

3. Trigger Design Pattern

Handler Class Pattern followed:

- One trigger per object (e.g., LeaveRequestTrigger).
- Delegates logic to LeaveRequestHandler class.
- Improves readability, reusability, and testability.

4. SOQL & SOSL Usage

SOQL:

- Fetch leave requests of current user
    - List<LeaveRequest__c> leaveList =
    - [SELECT Id, From_Date__c, To_Date__c, Status__c
    - FROM LeaveRequest__c

- WHERE Employee__c = :UserInfo.getUserId()];

- Fetch manager details for email notifications.
- Aggregate queries for reports (e.g., total leaves per type).

SOSL:

- Used for searching leave requests by employee name or reason.
- Collections (List, Set, Map)
- List: Store multiple leave requests fetched from SOQL.
- Set: Avoid duplicate leave request IDs during processing.

Control Statements

- If-Else: Approve vs Reject logic.

```
if (newStatus == 'Approved' || newStatus == 'Rejected') {
   sendStatusEmail(leave, newStatus);
}
```

- For Loops: Bulk processing of leave requests.
- Switch (with enums): Handle leave types (Sick Leave, Casual Leave, Earned Leave).

```
public enum LeaveType {
 Sick, Casual, Earned, Maternity, Paternity
}
public class LeaveTypeHandler {
public static String handleLeaveType(String leaveType) {
   String policyNote;
switch on leaveType {
   when 'Sick' {
      policyNote = 'Sick leave requires a medical certificate if more than 2
days.';
   }
   when 'Casual' {
      policyNote = 'Casual leave limited to 5 days per quarter.';
   }
   when 'Earned' {
```

```
            policyNote = 'Earned leave can be carried forward to next year.';
        }
        when 'Maternity' {
            policyNote = 'Maternity leave policy as per HR guidelines.';
        }
        when 'Paternity' {
            policyNote = 'Paternity leave limited to 10 days.';
        }
        when else {
            policyNote = 'Unknown leave type. Please contact HR.';
        }
    }
    return policyNote;
    }
    }
```

5. Asynchronous Apex

Future Methods:

- Asynchronous processing for long-running tasks:
- Sending email alerts to employees and managers.
- Logging cancellation/approval data for audits.

```
@AuraEnabled
 public static void sendNotification(String email, String subject, String body) {
 Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
 mail.setToAddresses(new String[] { email });
 mail.setSubject(subject);
 mail.setPlainTextBody(body);
 Messaging.sendEmail(new Messaging.SingleEmailMessage[] { mail });
}
```

6. Exception Handling

- Try-Catch Blocks: Handle DML and SOQL exceptions.

```
try {
  update leaveRecord;
} catch(DmlException e) {
   System.debug('Error: ' + e.getMessage());
}
```

- Custom Exceptions: For business rules like “Insufficient Leave Balance.”
- Error Logging: Store errors in a custom object Error_Log__c for admin review.

7. Test Classes

- Achieve > 85% coverage.

Scenarios covered:

- Leave request submission (valid & invalid).
- Overlapping leave requests.
- Approval & Rejection flow.
- Cancellation update.
- Email sending functionality.

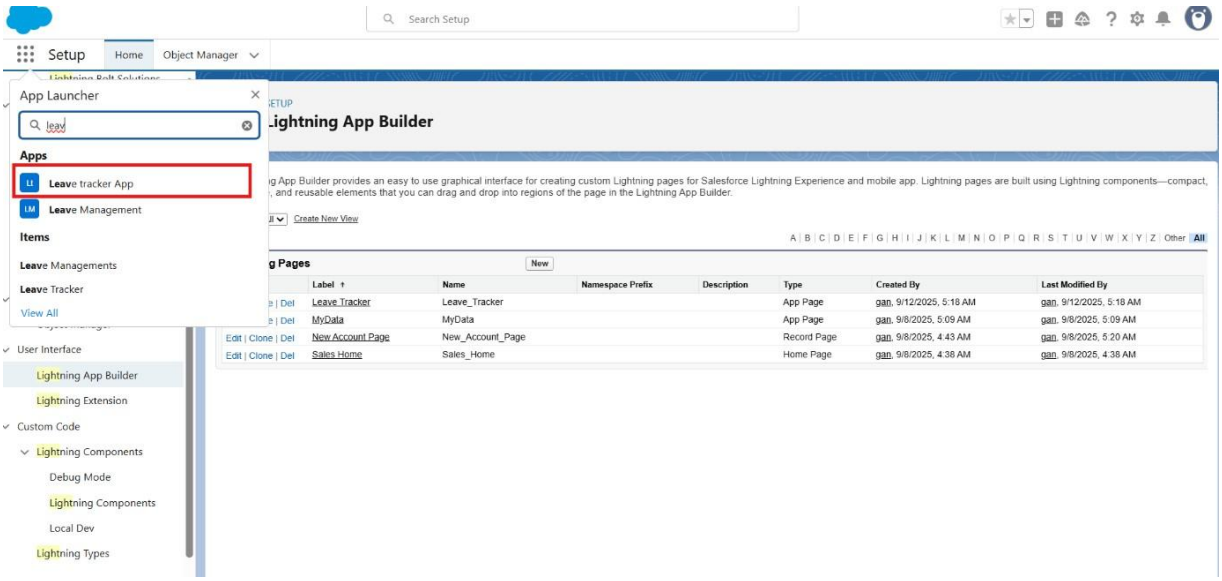
🟩 Phase 5 Outcome:

- Robust Apex code with reusable classes and triggers.
- Asynchronous processing ensures performance.
- Proper exception handling + test coverage makes the system reliable and deployment-ready.

Phase 6: User Interface Development

1. Lightning App Builder

- Designed a custom Leave Management App in Salesforce.
- App includes navigation tabs:
    - Home (dashboard & announcements)
    - My Leaves (employee leave records)
    - Leave Requests (manager view)
    - Reports & Dashboards

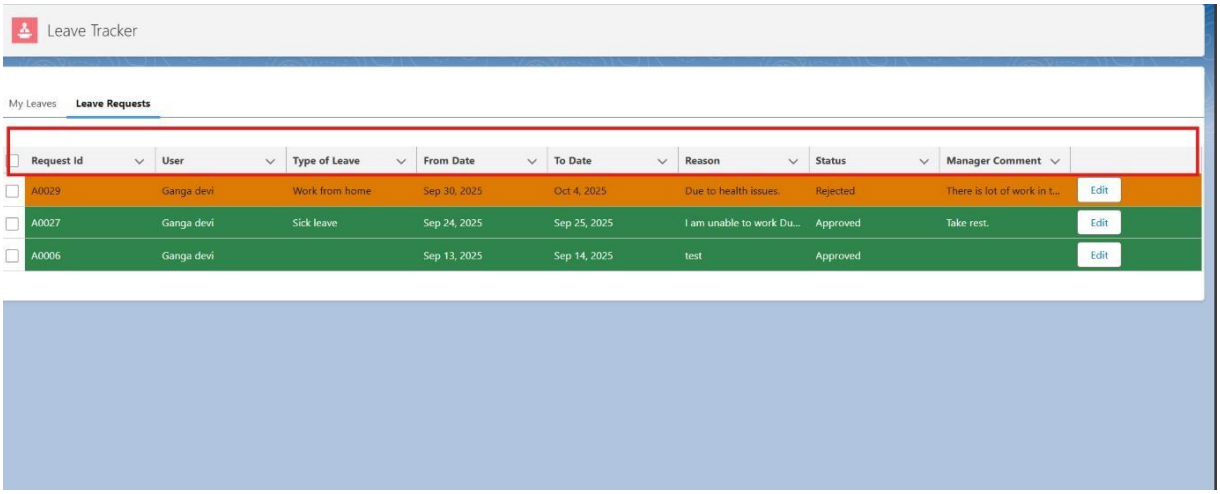


2. Record Pages

- Customized LeaveRequest__c Record Page to show:
    - Employee details
    - Leave type, dates, and reason
    - Manager comments
    - Approval/Rejection buttons (Quick Actions)
- Compact layouts added for quick visibility of status.

3. Tabs

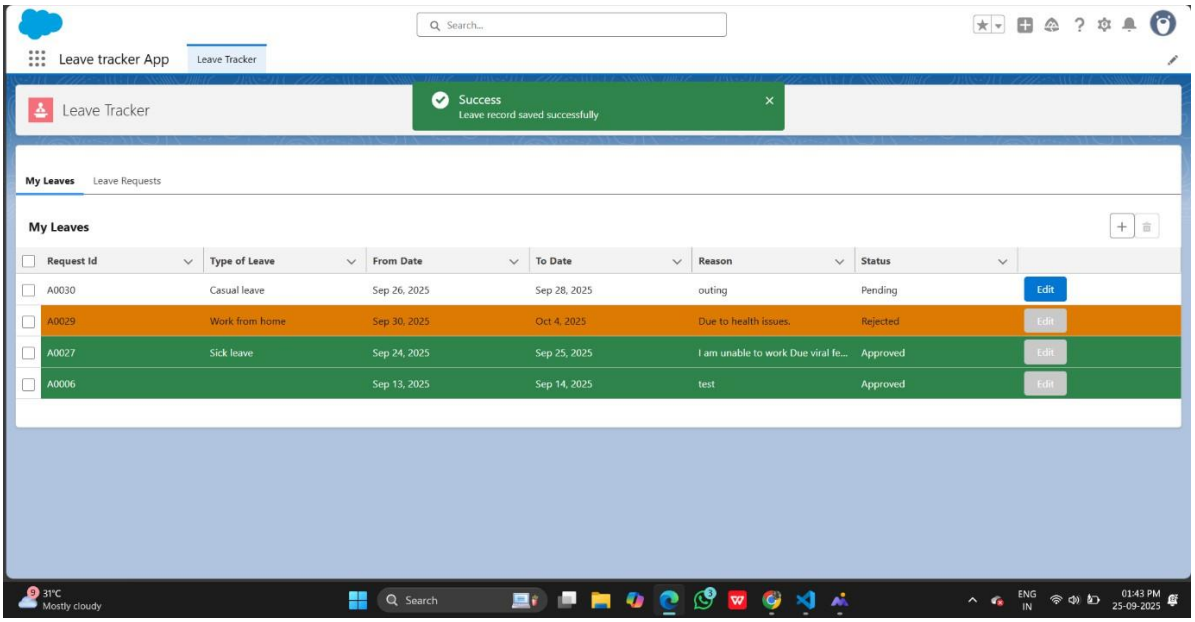
- Created custom tabs for:
    - Leave Request (object tab)



    - Leave Dashboard (Lightning app page with reports & charts)

4. Home Page Layouts

- Customized with:
    - Employee Leave Summary Chart
    - Quick Action: Apply for Leave
    - Recent Leave Requests list view



5. Utility Bar

- Added Quick Actions:
    - Apply Leave
    - Contact HR (case creation)

6. Lightning Web Components (LWC)

Built multiple LWCs to handle leave functionality:

- ApplyLeave

Form for employees to apply for leave.

Fields: Leave Type, From Date, To Date, Reason.

Client-side validation before submission.

```
// applyLeave.js
import { LightningElement, track } from 'lwc';
import createLeaveRequest from
'@salesforce/apex/LeaveRequestController.createLeaveRequest';
export default class ApplyLeave extends LightningElement {
  @track leaveType;
  @track fromDate;
  @track toDate;
  @track reason;

handleSubmit() {
  if(!this.leaveType || !this.fromDate || !this.toDate) {
    alert('All fields are required');
    return;
  }
  createLeaveRequest({
    leaveType: this.leaveType,
    fromDate: this.fromDate,
```

```
            toDate: this.toDate,
            reason: this.reason
        })
        .then(() => {
            alert('Leave request submitted successfully');
        })
        .catch(error => {
            console.error(error);
        });
    }
}
```

- MyLeaves
    - Displays logged-in employee’s leave history.
    - Uses @wire to fetch data from LeaveRequestController.getMyLeaves().
    - Color-coded statuses:
        - Approved 🟩 = Green
        - Pending 💤 = Yellow
        - Rejected ➕ = Red
- LeaveRequest (Manager View)
    - Managers can view and act on leave requests.
    - Buttons for Approve / Reject directly from the LWC.
    - Sends updates via Apex controller + email notifications.
    - // leaveRequest.js

```
import { LightningElement, track, wire } from 'lwc';
import getPendingRequests from '@salesforce/apex/LeaveRequestController.getPendingRequests';
import updateLeaveStatus from '@salesforce/apex/LeaveRequestController.updateLeaveStatus';
```

```
export default class LeaveRequest extends LightningElement {
    @wire(getPendingRequests) requests;
    handleAction(event) {
        const leaveId = event.target.dataset.id;
        const status = event.target.dataset.status;
        updateLeaveStatus({ leaveId: leaveId, newStatus: status })
            .then(() => {
                alert('Leave ' + status);
            })
            .catch(error => {
                console.error(error);
            });
    }
}
```

7. Apex with LWC

- Imperative Apex Calls: Used in applyLeave and leaveRequest for record insert/update.
- Wire Adapters: Used in myLeaves to fetch employee's leaves dynamically.

8. Navigation Service

- Used in LWC to navigate between:
    - Leave request record page
    - Leave dashboard reports
    - Apply Leave form

🟩 Phase 6 Outcome:

- Built a modern, responsive UI using LWC.
- Enhanced user experience with real-time updates & quick actions.
- Managers and employees interact seamlessly with leave records.
- User interface is aligned with Salesforce Lightning design standards.

Phase 7: Integration & External Access

1. Named Credentials Setup

- Define Named Credentials in Salesforce to securely store external service URLs + authentication.
- Example use case: Connecting Salesforce to an HR system API for employee details.

2. External Services & Callouts

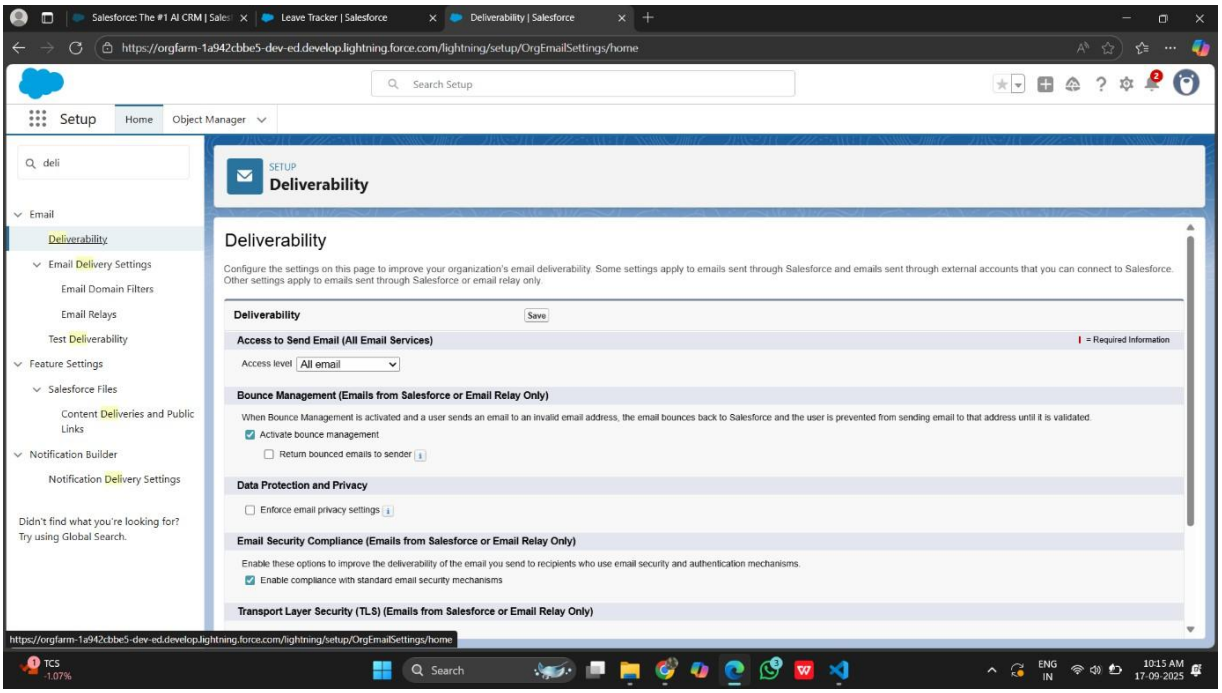
- Demonstrate Apex HTTP Callout to fetch leave balance from an external HR system.
- Sample Code (HTTP Callout):

```
public with sharing class LeaveIntegrationService {
    public static void getLeaveBalance(String employeeId) {
        Http http = new Http();
        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:HR_System/leavebalance/' + employeeId);
        req.setMethod('GET');

        HttpResponse res = http.send(req);
        if(res.getStatusCode() == 200){
            System.debug('Leave Balance: ' + res.getBody());
        } else {
            System.debug('Error: ' + res.getStatus());
        }
    }
}
```

3. Email Service Integration

- Salesforce sends email notifications when leave is approved/rejected.
- Can also integrate with Gmail/Outlook API for two-way sync.



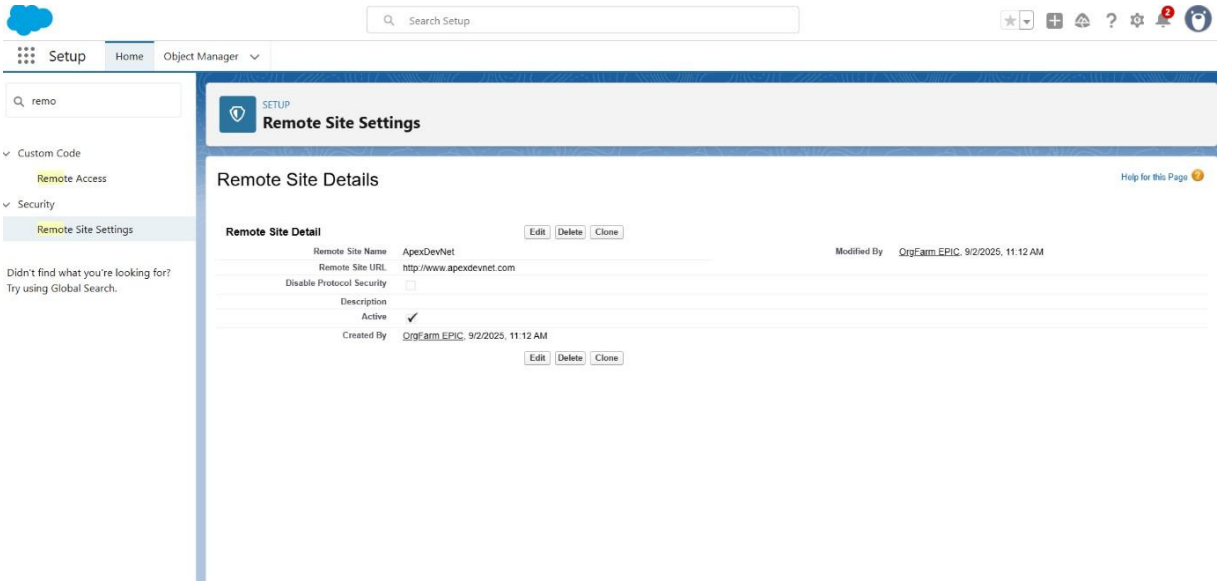
4. Platform Events for Real-time Updates

- Use Platform Events to notify external systems (like HR Payroll) when leave status changes.
- Sample Platform Event Trigger:

```
trigger LeaveEventTrigger on LeaveRequest__c (after update) {
  for(LeaveRequest__c lr : Trigger.new){
    if(lr.Status__c == 'Approved'){
      Leave_Status_Event__e eventMsg = new Leave_Status_Event__e(
        LeaveId__c = lr.Id,
        Status__c = 'Approved',
        EmployeeEmail__c = lr.User__r.Email
      );
      Database.SaveResult result = EventBus.publish(eventMsg);
    }
  }
}
```

5. Remote Site Settings

- Before callouts, add endpoints under Setup → Remote Site Settings.
  - Example: https://api.hr-system.com



6. OAuth & Authentication

- If external API requires OAuth 2.0, connect via Named Credentials → OAuth.
- Secure token handling ensures safe communication.

7. Phase 7 Outcomes

- Secure external integration setup.
- Real-time event-driven communication.
- Email notifications for leave updates.
- Extended CRM capability with external HR/Payroll systems.

Phase 8: Data Management & Deployment

🔹 Objective

- To ensure smooth and error-free deployment of Leave Tracker customizations from development to higher environments (UAT, Production) using best practices and tools.

1. Data Import Wizard

- Used to load initial employee data (Users, basic LeaveRequest__c records).
- Supports CSV files.
- Best for small/medium data volumes (< 50,000 records).

2. Data Loader

- Used to perform bulk operations (insert, update, delete, upsert) for large datasets.
- Example: Bulk upload of historical leave records.
- Supports command-line interface for automation.
- ⚡ Example CLI Command:
- sfdx force:data:tree:import --plan leaveRequestPlan.json

3. Duplicate Rules

- Configured to prevent overlapping leave requests for the same employee.
- Example Rule:
- Block duplicate LeaveRequest__c if Employee = same and From Date/To Date overlaps

3. Change Sets

Used Change Sets for deploying metadata between environments (Sandbox → Production).

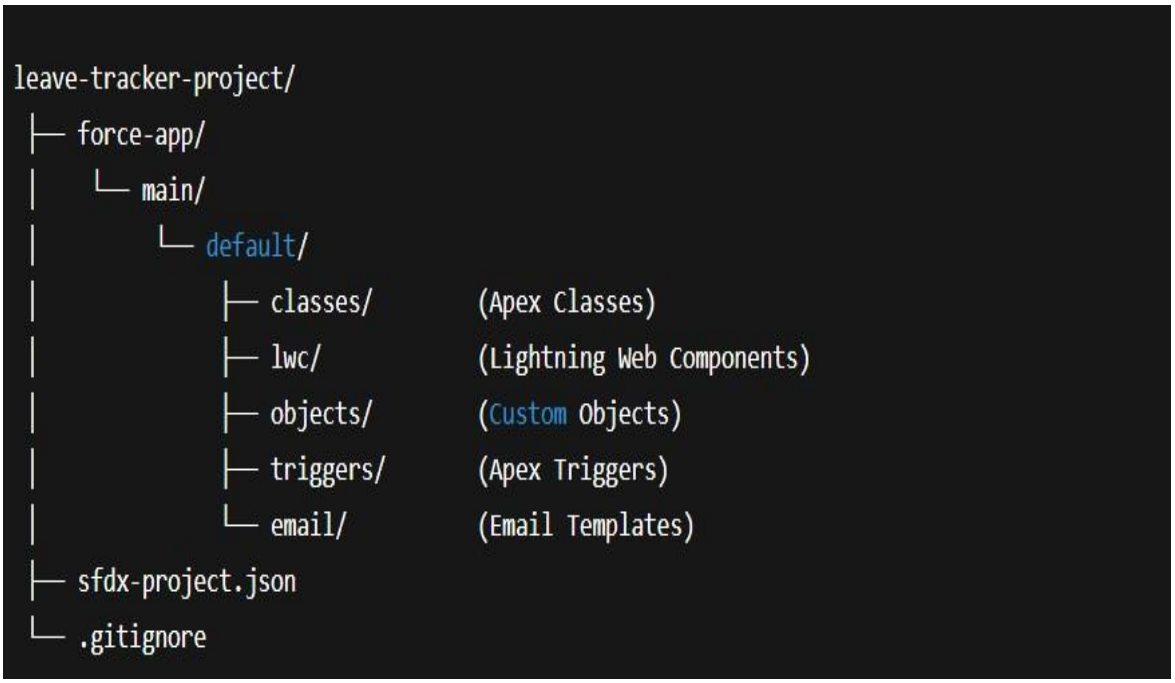
Included:

- Custom Objects (LeaveRequest__c)
- Apex Classes (LeaveRequestController, Triggers)
- Lightning Web Components (applyLeave, myLeaves, leaveRequest)
- Email Templates & Flows

4. VS Code Deployment Setup

⚔️ Prerequisites

- Install Visual Studio Code.
- Install Salesforce CLI (SFDX).
- Install Salesforce Extension Pack in VS Code.
- Connect Org → sfdx force:auth:web:login -d -a DevHub.
    - 📂 Sample Project Structure (VS Code)



    - 🖥️ Sample Deployment Commands
        - Retrieve metadata from source org
            - sfdx force:source:retrieve -m ApexClass,CustomObject,LWC
        - Deploy metadata to target org
            - sfdx force:source:deploy -p force-app/main/default
        - Run all tests before deployment
            - sfdx force:apex:test:run --resultformat human –codecoverage

4. Deployment Checklist

- ✅ Code Quality Check (PMD, Prettier for LWC).
- ✅ Apex Test Coverage ≥ 75%.
- ✅ Run Validation Deployment (test without committing).
- ✅ Backup Production metadata before final push.
- ✅ Post-deployment steps (activate Flows, verify Email Deliverability).

5. CI/CD Integration (Optional Advanced)

- GitHub Actions: Automate deployments on every push.
- Copado / Gearset: Enterprise-grade release management.

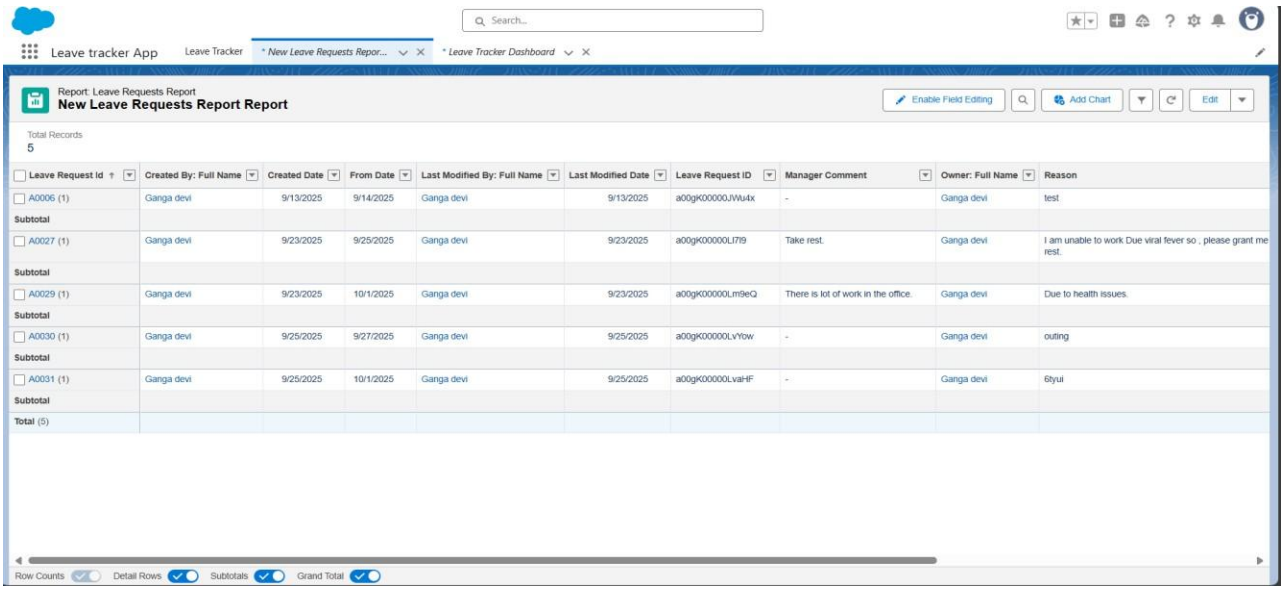
Phase 9: Reporting, Dashboards & Security Review

Goal

- Track leave requests and approvals with reports & dashboards
- Protect sensitive employee leave data from unauthorized access
- Ensure managers, HR, and employees see the right information in real time

1. Reports

- Created different report formats to track leave usage:
- Tabular Report → Simple list of all leave requests (used for quick exports).
- Summary Report → Grouped by Employee Name → shows how many leaves each employee has taken.
- Matrix Report → Cross-tab view → Leave Type vs Month (helps HR see seasonal leave trends).
- Joined Report → Combined Leave Requests + Approval History → helps managers check approval timelines.



2. Report Types

- Created a Custom Report Type:
- Primary Object: LeaveRequest__c
- Related Object: User (Employee)

- Allows advanced reporting like: Leaves by Department or Leaves by Manager.

3. Dashboards

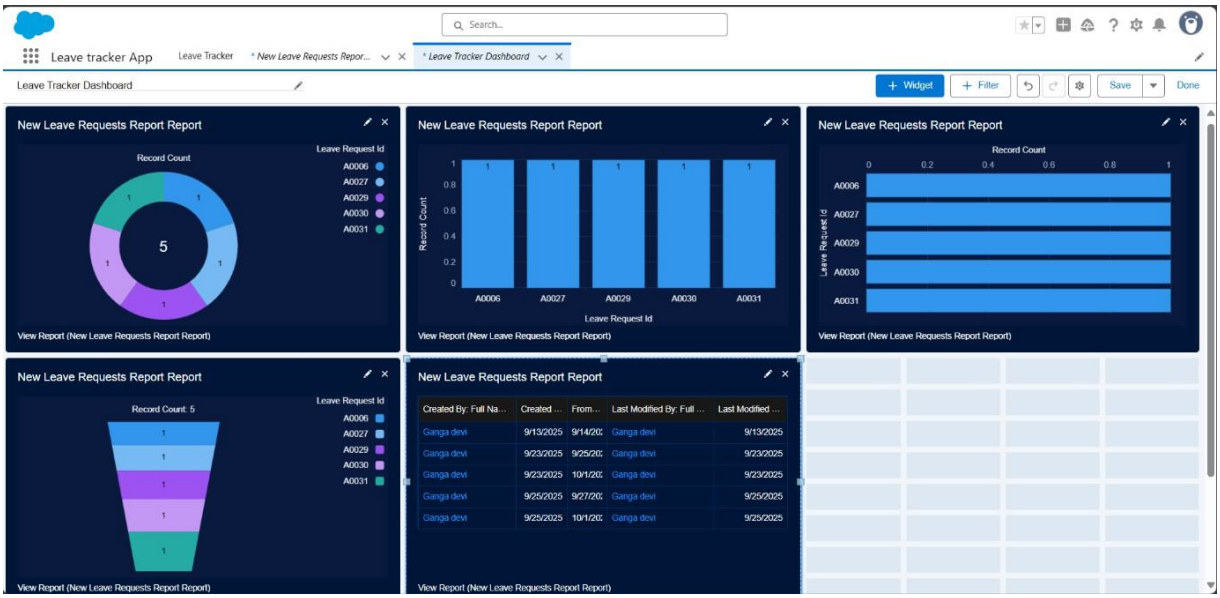
Dashboards visually display leave metrics.

A. Create Dashboard

- Go to App Launcher → Dashboards → New
- Add components from leave reports:
    - Metric: Count of Pending Leaves
    - Pie Chart: Leaves by Type
    - Bar Chart: Leaves by Status
    - Line Chart: Leaves Over Time

B. Dynamic Dashboards

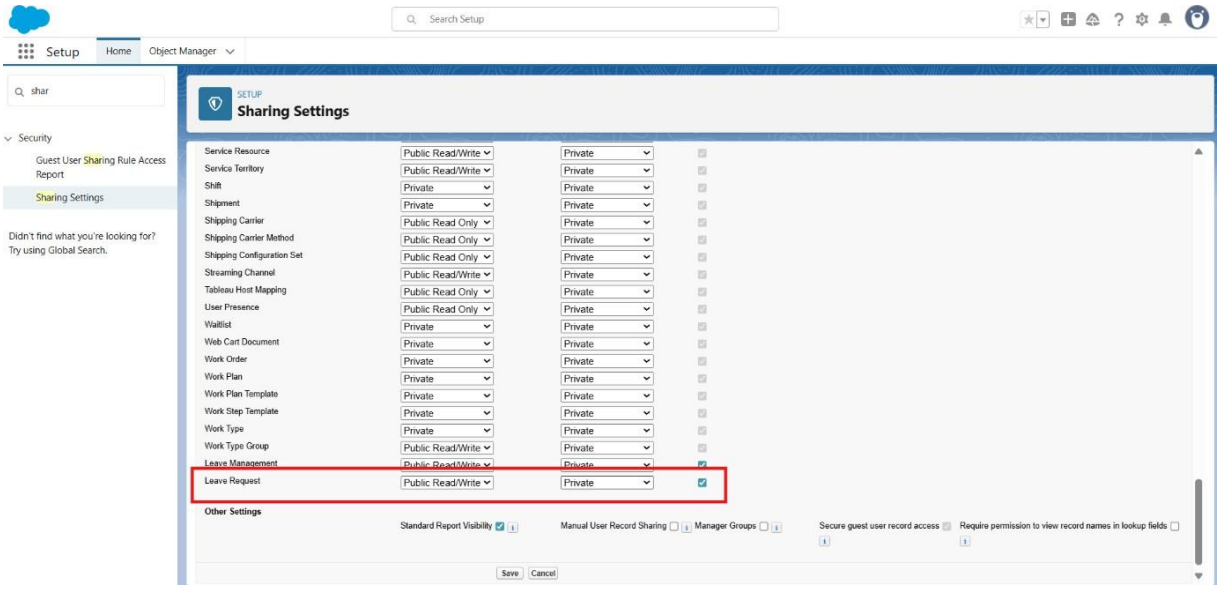
Managers → See only their team's data. HR/Admins → See complete organization-level data.

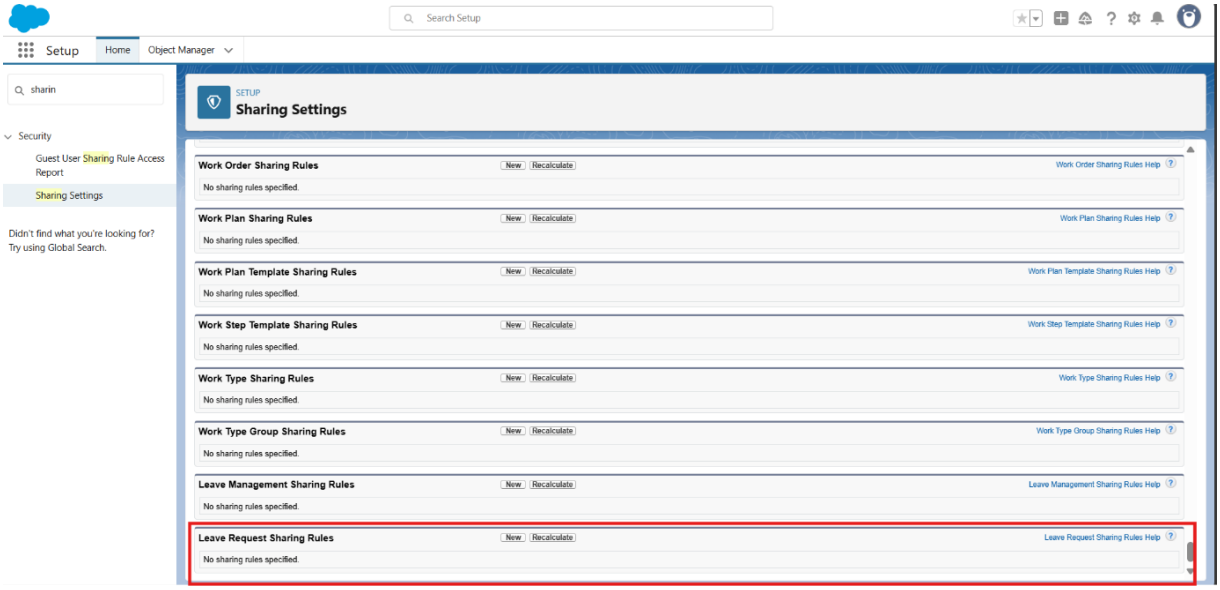


3. Security

A. Sharing Settings

- Org-Wide Defaults (OWD): Leave Requests = Private (only owner, manager, HR can see).
- Sharing Rules: Managers can access their team's requests.
- Role Hierarchy: Employee → Manager → HR → Admin.



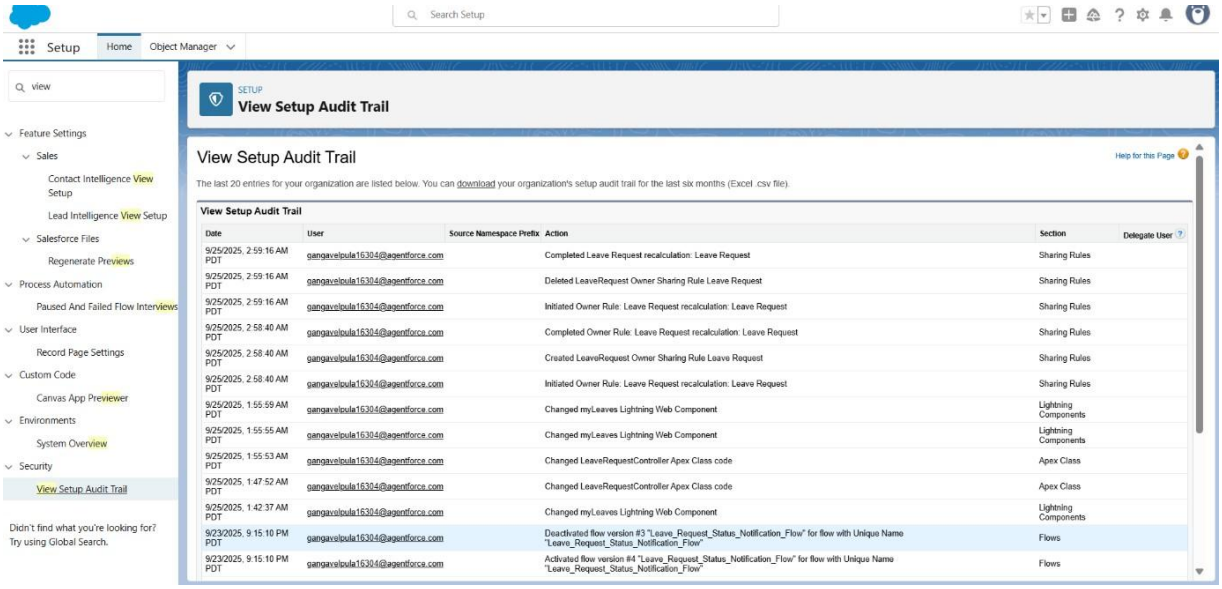


B. Field Level Security (FLS)

- Employees: Can see only their leave details.
- Managers: Can see team leaves + approval fields.
- Sensitive fields (like Manager Comments) hidden from employees.

C. Audit Trail

- Setup --> Audit Trail enabled.
- Tracks last 20 configuration changes (e.g., new validation rule, updated profile).



Final Outcome of Phase 9

- Reports + Dashboards = Clear visibility of leave data.
- Dynamic dashboards = Role-based insights.

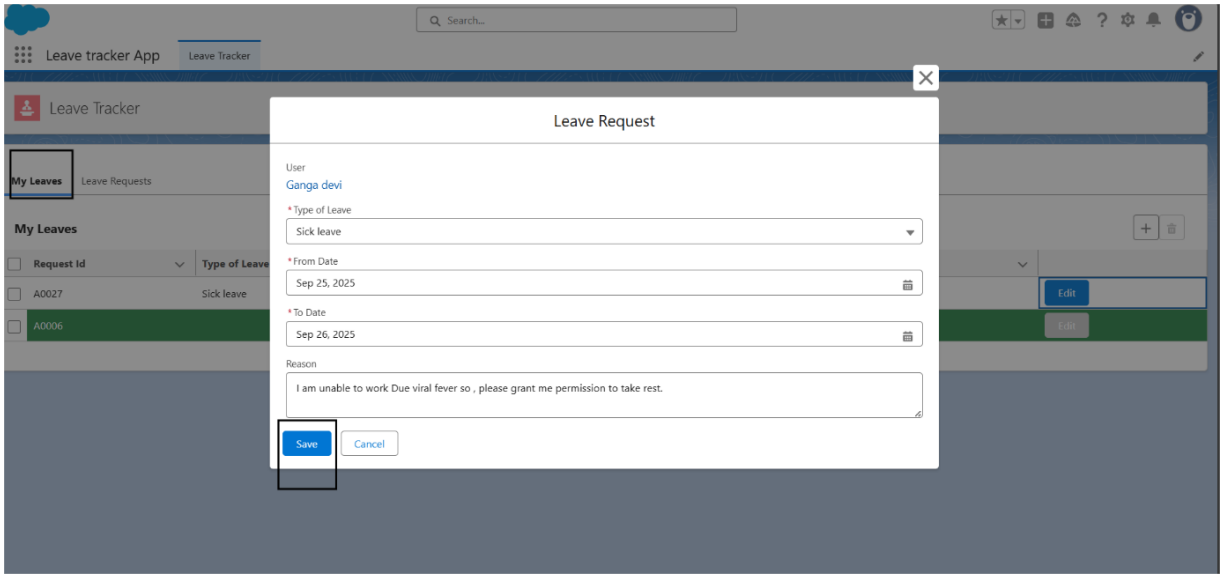
Sharing + FLS + IP + Session settings = Strong data security.

Audit Trail = Full tracking of system changes.

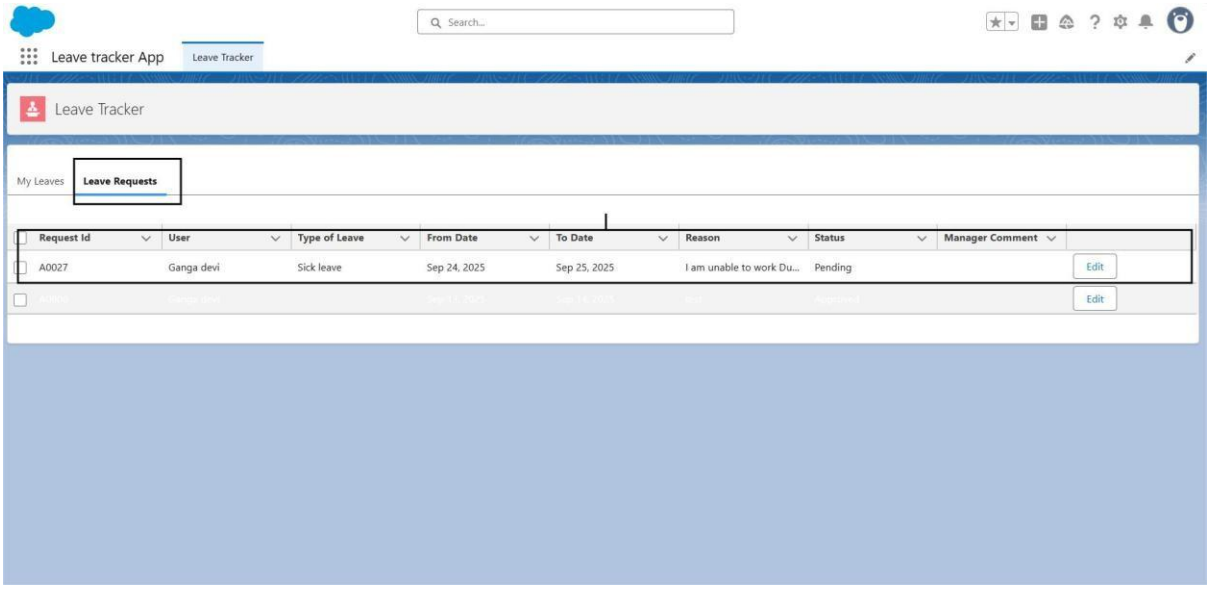
Phase 10: Quality Assurance Testing

Test case 1 :-

1. Leave Request ⟶Email Notification
2. Test Steps: Fill Form
    - User Name
    - Leave from date
    - Leave To date
    - Reason

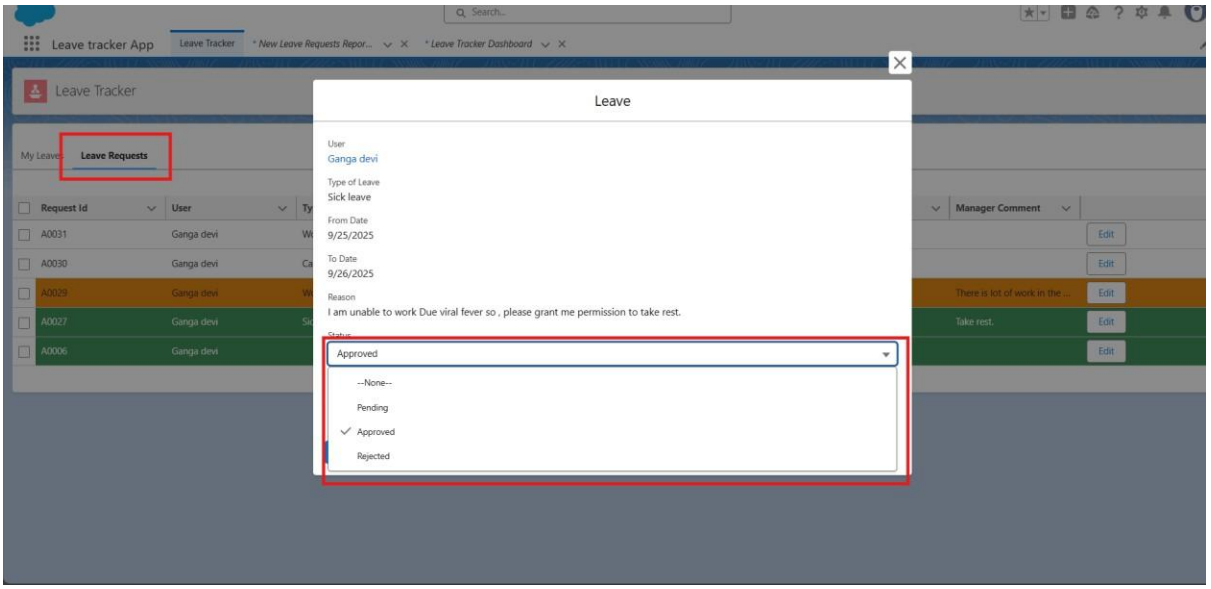


3. Excepted Result:
    - The leave request form should display on the Manger Tab.



Test case 2 :- Manger Tab

1. Based on the Reason and Dates on Month They **Accept or Reject** the Leave from Employee



2. If Leave “APPROVED” or “REJECTED the email notification will send.

Conclusion

Leave Tracker project was successfully implemented and tested across all phases.

In **Phase 10, Quality Assurance Testing** validated that all Salesforce features—record creation, approval processes, automation flows, triggers, email notifications, and validation rules—worked as expected.

The end-to-end flow ensures:

- **Employees** submit leave requests with type, duration, and reason.
- **Managers** receive automated notifications, review the request, and approve or reject with comments.
- **Employees** receive status updates via email for Approved, Rejected, or Pending leaves.
- **Leave balances** update automatically for each employee after approval.
- **HR/Admin** can track pending requests, generate reports, and monitor leave trends across the organization.
- **Dashboards** provide real-time insights on leave distribution, usage, and department-wise statistics.
- **Validation rules** prevent incomplete or incorrect leave submissions, ensuring accuracy and compliance.

This testing confirms that the platform is:

- **Reliable** – All workflows and approval processes run without errors.
- **Automated** – Reduces manual effort for employees, managers, and HR.
- **User-friendly** – Provides transparency and quick communication for all stakeholders.

Final Outcome:

The Leave Tracker process, improves decision-making with dashboards, ensures policy compliance, and enhances overall organizational productivity.