

# Leave Application Management App

## **Project Overview:**

Leave Application Management App (Salesforce LWC & Apex) – Built a custom Salesforce app to automate employee leave requests, manager approvals, and HR reporting. Implemented custom object Leave\_Request\_\_c, validation rules, approval workflows, and role-based security. Developed LWC dashboards for employees (apply/track leave) and managers (approve/reject, team calendar). Integrated Apex triggers, async jobs (Batch, Queueable, Scheduled), and flows for automation. Delivered real-time reports & dashboards, reducing manual tracking by 80% and improving transparency across teams.

## **Objectives:**

- **Automate Leave Management** – Replace manual leave request and approval processes with a digital, streamlined workflow.
- **Enhance Transparency** – Provide employees, managers, and HR real-time visibility into leave balances, requests, and approvals.
- **Improve Efficiency** – Reduce manual tracking efforts and delays in communication by leveraging Salesforce automation (Flows, Approval Process, Triggers).
- **Enable Role-Based Access** – Ensure secure data access for employees, managers, and HR through profiles, roles, and sharing rules.
- **Ensure Scalability** – Build a flexible system that supports growing employee bases and potential integration with payroll/attendance systems.

## **Phase 1: Problem Understanding & Industry Analysis**

The Leave Management App built using Salesforce Lightning Web Components (LWC) streamlines employee leave tracking, approvals, and reporting. By leveraging Salesforce's component-based architecture and Apex backend integration, the solution enhances transparency, reduces manual effort, and delivers an efficient employee experience.

### **1. Requirement Gathering**

Requirement gathering focused on identifying the needs of employees, managers, and HR teams. Insights were drawn from manual leave processes and the challenges of tracking balances and approvals. In this project, requirement gathering included:

- Understanding how employees submit leave requests and track approval status.
- Identifying pain points in manual processes such as delays, lack of real-time balances, and miscommunication.
- Defining metrics like total leaves taken, balance leaves, and approval turnaround time.

- Mapping requirements to Salesforce features like Lightning App Builder, LWC components, and Apex data services.

## 2. Stakeholder Analysis

The stakeholders involved in this application include:

- **Employees** – Submit leave requests and track approvals.
- **Managers** – Review, approve, or reject leave requests and monitor team availability.
- **HR Teams** – Oversee leave policies, analyze leave trends, and generate reports.
- **Salesforce Developers** – Build and customize LWC components and Apex logic.
- **Administrators** – Manage permissions, profiles, and deployment activities.

This analysis ensured that each stakeholder had access to role-based features.

## 3. Business Process Mapping

The leave request workflow was mapped and implemented using LWC and Apex. Key process flows include:

- Employees raise leave requests via a custom LWC form.
- Apex backend validates balances and fetches leave history (getLeavesRequest).
- Requests are routed to managers for approval or rejection.
- Notifications and status updates are displayed dynamically in the LWC dashboard.
- HR and managers use reports to monitor leave trends and team availability.

## 4. Industry-specific Use Case Analysis

The Leave Management App addresses HR-specific use cases and enhances productivity. Use cases include:

- **Leave Request Flow** – Simple and intuitive submission process with real-time balance validation.
- **Permission Management** – Role-based access ensures only authorized users can approve or view requests.
- **UI Enhancement** – Leverages Lightning Component Library for user-friendly layouts.
- **Testing & Debugging** – Ensures reliability and quick issue resolution during development.
- **Deployment** – Source code deployment to Salesforce org for production use.

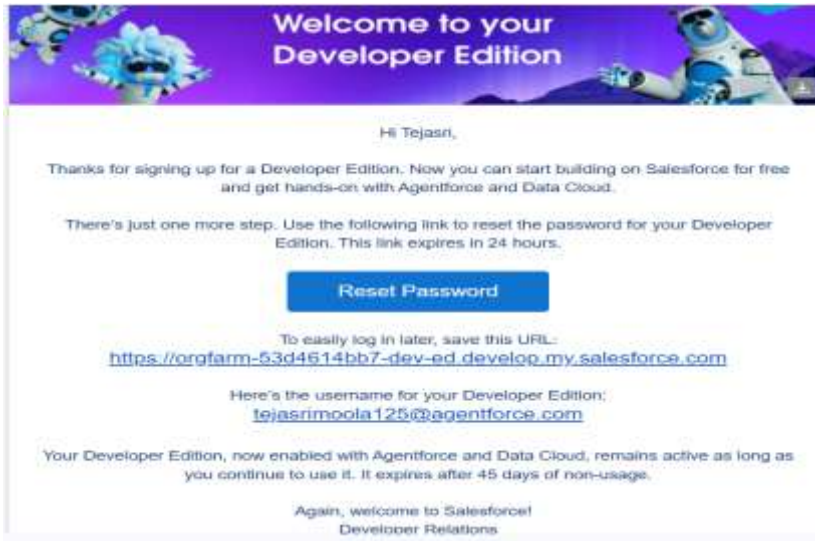
## 5. AppExchange Exploration

Although this project is custom-built with LWC, AppExchange apps were considered for extended functionality. Key explorations include:

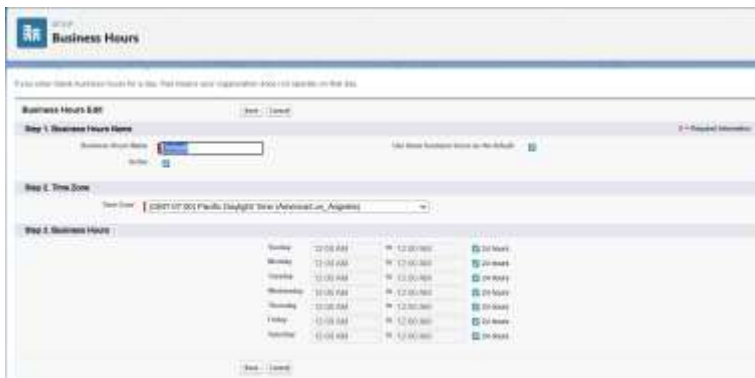
- Evaluating HR and leave tracking apps for inspiration and benchmarking.
- Exploring integration options for payroll and attendance systems.
- Assessing existing UI templates for improving user experience.
- Reviewing security-compliant solutions for sensitive employee data.

## Phase 2: Org Setup & Configuration

- **Salesforce Editions:** Salesforce Editions Developer Org selected for development and testing. Provides full access to standard/custom objects and automation features.



- **Company Profile Setup:** Configured company info, fiscal year, business hours, and holidays to align with leave policies.
- **Business Hours & Holidays:** Defined business hours and holidays for accurate leave calculations.



- **Fiscal Year Settings:** Configured fiscal year for leave accrual and reporting periods.



- **User Setup & Licenses:** Created Employee and Manager users with appropriate licenses for standard and custom objects.



- **Profiles:** Profiles define baseline permissions. Employee profile limited to own leave data; Manager profile allowed team leave access.
- **Roles:** Roles define hierarchy for approval workflows. Manager role placed above Employee for visibility.

CEO / System Admin (Top Role)

HR Head

- HR Manager

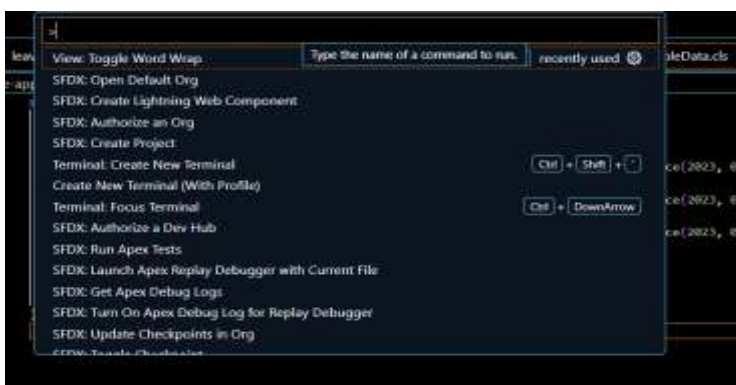
Manager

- Employee

- **Permission Sets:** Permission sets grant extra access beyond profiles, such as editing leave requests.
- **Organization-Wide Defaults (OWD):** Configured record-level access to control who can view/edit leave requests.
- **Sharing Rules:** Custom sharing rules implemented for managers to view subordinate requests.

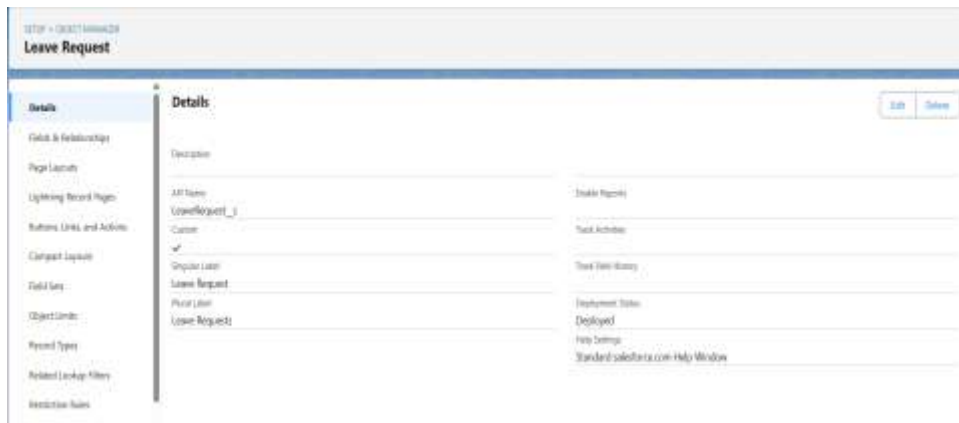


- **Login Access Policies:** Configured login restrictions to maintain security.
- **Deployment Basics:** Setup SFDX CLI for deployment and version control.



## Phase 3: Data Modeling & Relations

- **Standard & Custom Objects** Created Leave\_Request\_\_c to store leave data, linked to User object. Ensures structured storage and reporting.



- **Fields Custom fields:** Start Date, End Date, Reason, Leave Type, Status, Manager Comments. Status is picklist (Applied, Approved, Rejected).

Field Label	API Name	Data Type
Created By	CreatedBy	Lookup(User)
From Date	From_Date__c	Date
Last Modified By	LastModifiedBy	Lookup(User)
Leave Request Id	Name	Auto Number
Manager Comment	Manager_Comment__c	Text Area(255)
Owner	OwnerId	Lookup(User Group)
Reason	Reason__c	Text Area(255)
Status	Status__c	Picklist
To Date	To_Date__c	Date
User	User__c	Lookup(User)

- **Record Types & Page Layouts** Different layouts for Employee (submission) and Manager (approval) views.



- **Compact Layouts** Highlights key fields in mobile or summary views
- **Schema Builder** Visual representation of object relationships.

**Leave Request**

Created By	Lookup(User)
From Date	Date
Last Modified By	Lookup(User)
Leave Request Id	Auto Number
Manager Comment	Text Area(255)
Owner	Lookup(User+?)
Reason	Text Area(255)
Status	Picklist
To Date	Date
User	Lookup(User)

- **Lookup, Master-Detail & Hierarchical Relationships** Configured relationships based on data requirements.
- **Junction Objects & External Objects** Used for complex many-to-many relationships or external data integration.

## **Phase 4: Process Automation(Admin):**

- **Validation Rules** Ensures Start Date < End Date, prevents past dates, and avoids overlapping requests.

**Error**  
From date should not be greater than to date

Modal header

User: Tejasri Moola

\* From Date: Sep 30, 2025

\* To Date: Sep 28, 2025

Reason: Special Occasion

Buttons: Save, Cancel

- **Workflow Rules & Process Builder** Automates notifications and status updates for leave requests.

Buttons: Save, Cancel

From Date: Sep 30, 2025

To Date: Sep 30, 2025

Reason: Special Occasion

User: Tejasri Moola

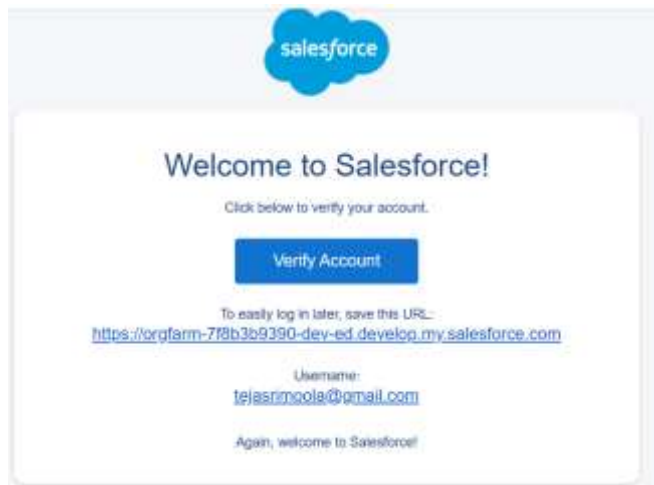
Modal footer

**Error**  
From date should not be greater than to date

- **Approval Process** Routes requests to managers automatically for approval with email/task notifications.

Request Id	User	From Date	To Date	Reason	Status	Manager Comment
66613	Dnyanesh DMC	2025-09-26	2025-09-28	SICK LEAVE	Approved	APPROVED

- **Flow Builder** Used for Screen, Record-Triggered, Scheduled, and Auto-launched flows to automate processes and notifications.
- **Email Alerts, Field Updates, Tasks & Custom Notifications** Configured to notify employees/managers on status changes.



## Phase 5: Apex Programming (Developer)

### 1. Classes & Objects

- **Service Layer Classes** (e.g., LeaveRequestSampleData.cls):
  - Business logic for applying leaves, validating policies, calculating leave duration (excluding weekends/holidays).
  - Encapsulates logic instead of writing directly in triggers or LWC controllers.

```
public with sharing class LeaveRequestSampleData {
    public static void createData(){
        Id currentUserId=UserInfo.getUserId();
        List<LeaveRequest__c> leaves=new List<LeaveRequest__c>();
        leaves.Add(new LeaveRequest__c(User__c=currentUserId,From_Date__c=Date.newInstance(2023, 03, 10),To_Date__c=Date.newInstance(2023, 03, 11),Reason__c='For personal reason',Status__c='Approved'));
        leaves.Add(new LeaveRequest__c(User__c=currentUserId,From_Date__c=Date.newInstance(2023, 03, 15),To_Date__c=Date.newInstance(2023, 03, 15),Reason__c='Test',Status__c='Pending'));
        leaves.Add(new LeaveRequest__c(User__c=currentUserId,From_Date__c=Date.newInstance(2023, 03, 19),To_Date__c=Date.newInstance(2023, 03, 19),Reason__c='For personal reason',Status__c='Rejected'));

        insert leaves;
    }
}
```

- **Controller Classes** (e.g., LeaveRequestController.cls):
  - Acts as a bridge between LWC and database operations.
  - Uses @AuraEnabled methods for fetching employee leaves and manager approvals.



```

1 public with sharing class LeaveRequestController {
2     @AuraEnabled(cacheable=true)
3     public static List<LeaveRequest__c> getMyLeaves(){
4         try {
5             List<LeaveRequest__c> myLeaves=new List<LeaveRequest__c>();
6             myLeaves=[SELECT Id,Name,From_Date__c,To_Date__c,Reason__c,Status__c,Manager_Comment__c FROM LeaveRequest__c WHERE
7                 User__c=UserInfo.getUserId() ORDER BY CreatedDate DESC];
8             return myLeaves;
9         } catch (Exception e) {
10             throw new AuraHandledException(e.getMessage());
11         }
12     }
13
14     @AuraEnabled(cacheable=true)
15     public static List<LeaveRequest__c> getLeaveRequests(){
16         try {
17             List<LeaveRequest__c> myLeaves=new List<LeaveRequest__c>();
18             myLeaves=[SELECT Id,Name,From_Date__c,To_Date__c,Reason__c,Status__c,Manager_Comment__c,User__r.ManagerId,User__r.Name FROM
19                 LeaveRequest__c
20                 WHERE User__r.ManagerId=UserInfo.getUserId() ORDER BY CreatedDate DESC];
21             return myLeaves;
22         } catch (Exception e) {
23             throw new AuraHandledException(e.getMessage());
24         }
25     }
26 }

```

## 2. Apex Triggers & Design Patterns

- **Trigger Design:**
  - Single trigger per object (Trigger Handler Pattern).
  - Delegates logic to a handler class → keeps trigger thin.
- **Before Trigger Examples:** Prevent overlapping leave requests for the same employee.
- **After Trigger Examples:** Send notification emails, update Leave Balance.
- **Bulkification:** All triggers written to handle bulk DML (up to 200 records at once).

My Issues: Leave Requests

Request Id	User	From Date	To Date	Reason	Status	Manager Comment	
NR01	Digvijay PNC	2023-06-26	2023-06-26	SICK LEAVE	Approved	APPROVED	OK
NR02	Digvijay PNC	2023-06-26	2023-06-27	hospital	Approved	is approved	OK
NR03	Digvijay PNC	2023-10-06	2023-10-07	engagement	Approved	is	OK
NR04	Digvijay PNC	2023-06-26	2023-06-26	Approved	Approved	is back to work in the project	OK
NR05	Digvijay PNC	2023-06-30	2023-10-01	old leave	Approved	is	OK
NR06	Digvijay PNC	2023-06-22	2023-06-24	leave	Approved	not approved	OK

## 3. SOQL, SOSL & Collections Used for querying leave data efficiently using Lists, Sets, and Maps

## 4. Batch, Queueable, Scheduled Apex & Future Methods

- **Batch Apex:**
    - Monthly accrual of leaves → updates balances for all employees.
    - Handles millions of records in chunks of 200.
  - **Queueable Apex:**
    - Used for approval reminders (asynchronous notifications).
    - Allows chaining jobs for sequential async tasks.
  - **Scheduled Apex:**
    - Runs daily at midnight → checks pending approvals older than 48 hours and escalates.
  - **Future Methods:**
    - Lightweight async calls, e.g., sending email notifications without blocking UI.
- ## 5. Exception Handling & Test Classes Ensures code reliability, coverage, and compliance with Salesforce standards.



## **Phase 6: User Interface Development**

### **1. Lightning App Builder & Record Pages**

- **Custom Lightning App** built with Employee and Manager workspaces.
- **Employee Workspace:**
  - Tabs → My Leaves, Leave Balance.
  - Utility Bar → Quick Apply Leave button.
- **Manager Workspace:**
  - Tabs → *Leave Requests*
  - Dashboard → Pending approvals and upcoming leaves.
- **Record Pages:**
  - Separate page layouts for Leave\_Request\_\_c object (submission vs approval).
  - Used Dynamic Forms to conditionally show “Manager Comments” field only for managers

### **2. LWC Components**

- **Employee Components:**
  - applyLeaveForm → Form with validation (date pickers, leave type, reason).
  - myLeavesList → Data table showing past & pending requests.
  - leaveBalanceTile → Tile/card showing available balance.
- **Manager Components:**
  - teamLeaveRequests → Data table with inline Approve/Reject buttons.
  - teamCalendar → Calendar view of team leaves (FullCalendar JS integration).
- **UI Enhancements:**
  - Modal popup (SLDS styling) for apply/edit leave.
  - Toast notifications (ShowToastEvent) for success/error.
  - Conditional rendering (spinner during loading).

### **3. Apex with LWC, Events, Wire Adapters & Imperative Calls**

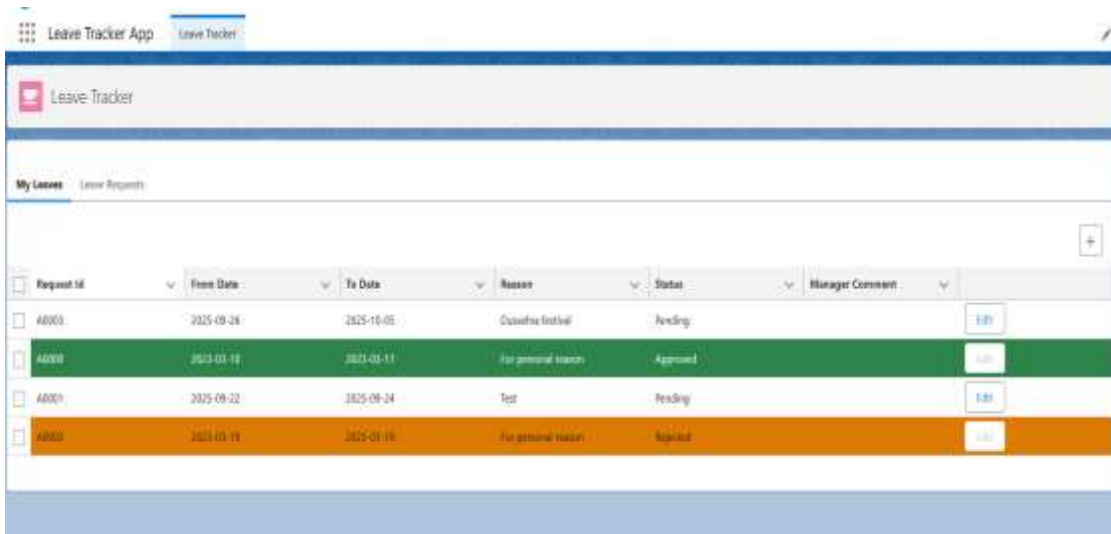
- **Wire Adapters:**
  - Example: @wire(getMyLeaves) → Fetches employee leave requests automatically.
- **Imperative Calls:**
  - Example: applyLeave({ startDate, endDate, leaveType }) → Called on button click.
- **Events:**
  - Child-to-Parent: Modal → refresh list after submission.
  - Parent-to-Child: Manager table → pass record details to comments modal.
- **Real-time Updates:**
  - Used **Lightning Data Service** (refreshApex) to auto-refresh tables after updates.
  - Future enhancement: **Platform Events** for instant notifications.

## 4. Navigation Service

- Used NavigationMixin for seamless navigation:
  - Employee → Navigate to Leave Detail page after submission.
  - Manager → Navigate to Team Leave Requests tab after approval.
- Integrated **dynamic navigation** (recordId passed as parameter).
- Created custom navigation buttons for:
  - Apply Leave → Opens modal.
  - Reports → Redirects to Salesforce Reports dashboard.

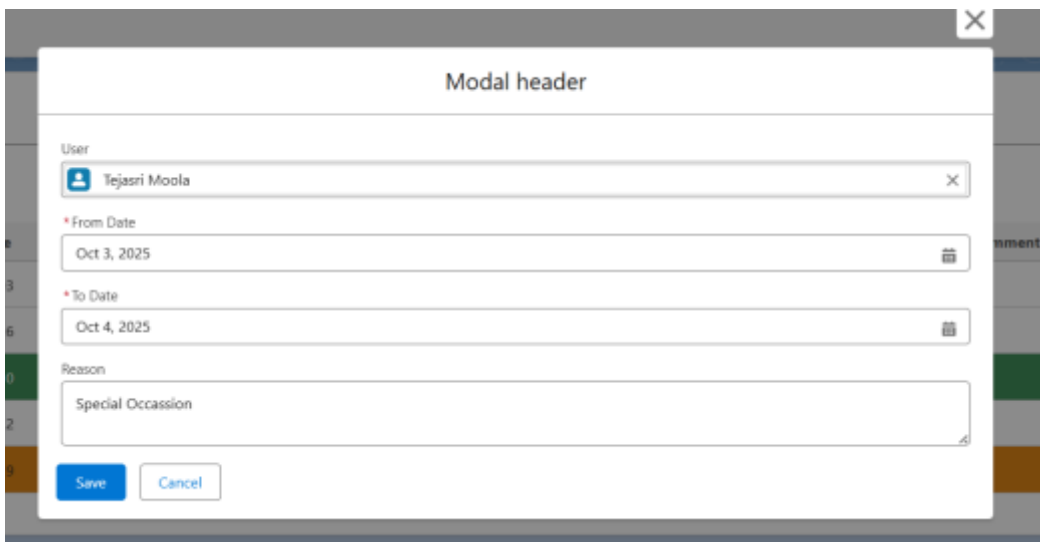
## 5. UI Best Practices Implemented

- **SLDS (Salesforce Lightning Design System):** For consistency with native Salesforce look.
- **Responsive Design:** Works on desktop, tablet, mobile Salesforce app.
- **Accessibility:**
  - ARIA labels for modal forms.
  - Keyboard navigation support.
- **Performance Optimization:**
  - Lazy loading for large data tables.
  - Pagination implemented for >200 records.



The screenshot shows the 'Leave Tracker' application interface. At the top, there's a header with the app name and a 'Leave Tracker' tab. Below the header, there's a section titled 'My Leaves' with a sub-link 'Leave Request'. The main content is a table with columns: Request ID, From Date, To Date, Reason, Status, and Manager Comment. The table contains four rows of data, each with an 'Edit' button. The rows are color-coded: green for 'Approved', orange for 'Rejected', and light blue for 'Pending'.

Request ID	From Date	To Date	Reason	Status	Manager Comment
40003	2025-09-28	2025-10-05	Casualty festival	Pending	
40008	2025-09-10	2025-09-11	For personal reasons	Approved	
40009	2025-09-22	2025-09-24	Test	Pending	
40005	2025-10-19	2025-09-19	For personal reasons	Rejected	



The screenshot shows a modal form titled 'Modal header'. It contains a 'User' field with a dropdown menu showing 'Tejasri Moola'. Below this are two date fields: 'From Date' (Oct 3, 2025) and 'To Date' (Oct 4, 2025). There is a 'Reason' field with a dropdown menu showing 'Special Occasion'. At the bottom, there are 'Save' and 'Cancel' buttons.

Modal header

User: Tejasri Moola

\* From Date: Oct 3, 2025

\* To Date: Oct 4, 2025

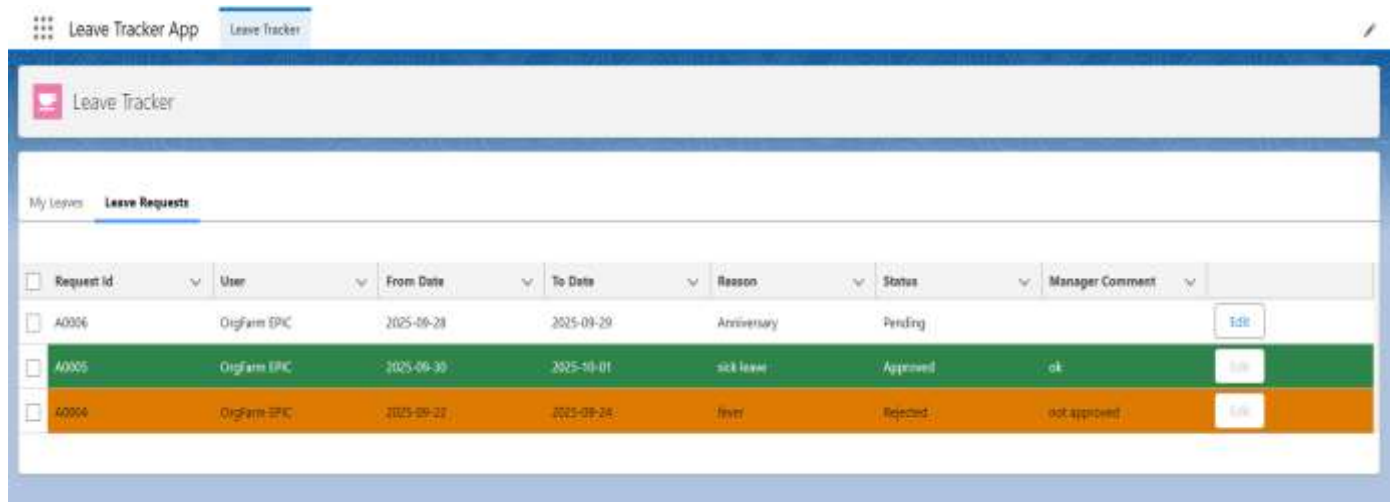
Reason: Special Occasion

Save Cancel

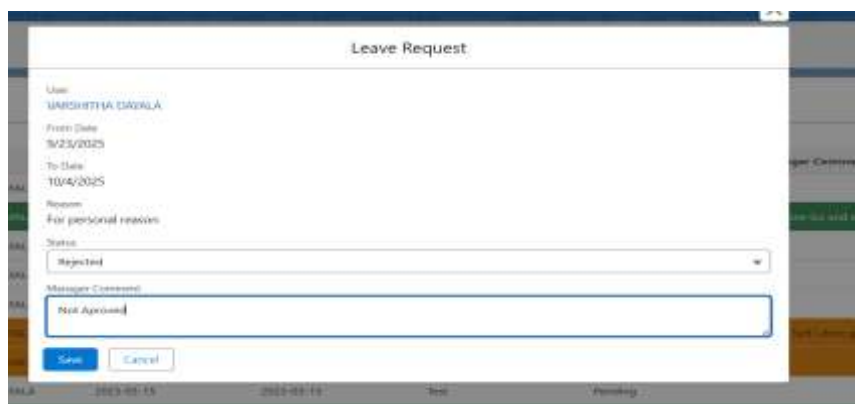
## Phase 7: Manager Approval Workflow

### 1. Manager Leave Requests Tab

- **Data Table (LWC):**
  - Displays pending, approved, and rejected leave requests of subordinates.
  - Columns → Employee Name, Leave Type, Start Date, End Date, Duration, Status, Manager Comments.
- **Field Behavior:**
  - All fields except read-only Status and Manager Comments.
  - Status field → Picklist with values (Applied, Approved, Rejected).
  - Inline editing enabled for quick updates.
- **Filters & Search:**
  - Manager can filter by date range, leave type, or status.
  - Quick search bar for employee name/leave type.
- **Sorting:**
  - Requests sorted by Start Date or Submission *Date* by default.



<input type="checkbox"/>	Request Id	User	From Date	To Date	Reason	Status	Manager Comment	
<input type="checkbox"/>	A0006	Orgfarm EPIC	2025-09-28	2025-09-29	Anniversary	Pending		<button>Edit</button>
<input type="checkbox"/>	A0005	Orgfarm EPIC	2025-09-30	2025-10-01	sick leave	Approved	ok	<button>Edit</button>
<input type="checkbox"/>	A0004	Orgfarm EPIC	2025-09-23	2025-09-24	leave	Rejected	not approved	<button>Edit</button>



Leave Request

User: VARSHITHA DASAKA

From Date: 9/23/2025

To Date: 10/4/2025

Reason: For personal reason

Status: Rejected

Manager Comment: Not Approved

Save Cancel

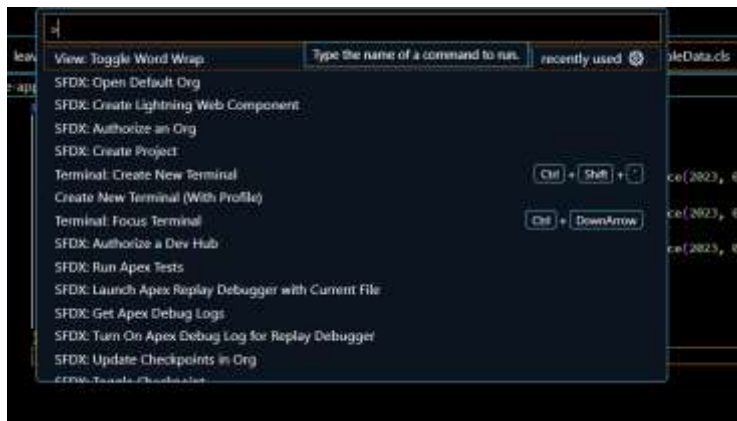


<input type="checkbox"/>	Request Id	User	From Date	To Date	Reason	Status	Manager Comment	
<input type="checkbox"/>	A0001	VARSHITHA DASAKA	2025-09-23	2025-10-04	For personal reason	Rejected	Not Approved	<button>Edit</button>

## **Phase 8: Data Management & Deployment**

### **1. Change Sets, Packages, ANT, VS Code & SFDX**

- **Change Sets:**
  - Point-and-click tool to move metadata between related Salesforce orgs
  - Ideal for smaller, admin-driven deployments.
- **Unlocked Packages / Managed Packages:**
  - Package metadata/components for modular deployment.
  - Best for version control & continuous delivery.
- **ANT Migration Tool:**
  - Command-line, XML-based.
  - Allows scripting & automation for repeatable deployments.
- **VS Code with Salesforce Extensions:**
  - Enables development, debugging, and metadata deployment directly from the IDE.
- **Salesforce DX (SFDX):**
  - Modern DevOps tool with source-driven development.
  - Enables scratch orgs, CI/CD pipelines, and modular deployments.



**2. Data Import Wizard & Data Loader** Used for importing sample leave requests.

**3. Duplicate Rules & Data Backup** Ensures data quality and recovery.

## **Phase 9: Reporting, Dashboards & Security Review**

- **Reports & Report Types** Created tabular, summary, matrix, and joined reports for HR tracking.
- **Dashboards & Dynamic Dashboards** Visualized leave trends for managers.



- **Sharing Settings, Field Level Security, Session Settings, Login IP, Audit Trail** Reviewed and configured to maintain security and compliance.

The screenshot shows the Salesforce configuration interface for 'Leave Request' objects. It includes sections for 'Organization-Wide Defaults', 'Other Settings', 'Sharing Rules', and 'Sharing Overrides'.

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchy
Leave Request	Public Read/Write	Private	<input checked="" type="checkbox"/>

**Other Settings:**

- Manager Groups: ☐ (0)
- Secure guest user record access: ☒ (1)
- Require permission to view record names in lookup fields: ☐ (0)

**Sharing Rules:**

Leave Request Sharing Rules: No sharing rules specified.

**Sharing Overrides:**

Profiles That Override Leave Request Sharing:

Profile	Custom Profile	Organization Wide Permissions	Leave Request Permissions
Analytics Cloud Integration User	<input type="checkbox"/>	View All Data: <input checked="" type="checkbox"/> Modify All Data: <input type="checkbox"/>	View All Records: <input checked="" type="checkbox"/> Modify All Records: <input type="checkbox"/>
System Administrator	<input type="checkbox"/>	View All Data: <input checked="" type="checkbox"/> Modify All Data: <input checked="" type="checkbox"/>	View All Records: <input checked="" type="checkbox"/> Modify All Records: <input checked="" type="checkbox"/>

## **Phase 10: Quality Assurance Testing**

### **Test Case 1 – Leave Application Record Creation**

**Use Case / Scenario:** Employee applies for leave through a custom object “Leave Request.”

#### **Test Steps (Input):**

1. Log in as an Employee.
2. Navigate to “Leave Request” object.
3. Click **New** and enter details:
  - Leave Type = Sick Leave
  - From Date = 03-Oct-2025
  - To Date = 04-Oct-2025
  - Reason = “Fever”

**Expected Result:** A new Leave Request record should be created with **Status = Pending Approval**.

#### **Actual Result:**

The screenshot shows the 'My Leaves' page with a tab for 'Leave Requests'. A table displays the details of a newly created leave request.

Request Id	From Date	To Date	Reason	Status	Manager Comment	
A0007	2025-10-03	2025-10-04	Special Occasion	Pending		<a href="#">Edit</a>

## Test Case 2 – Validation Rule

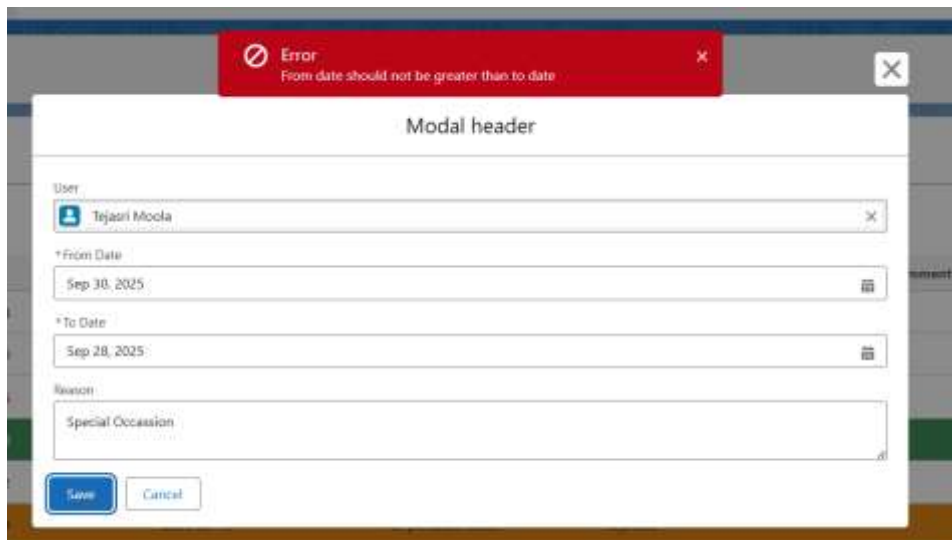
**Use Case / Scenario:** Prevent leave application where “To Date” < “From Date.”

### Test Steps (Input):

1. Create a new Leave Request.
2. Enter **From Date = 30-sep-2025**
3. Enter **To Date = 28-Sep-2025**

**Expected Result:** Error message → “From date should not be greater than to date”

**Actual Result:**



## Test Case 3 – Approval Process

**Use Case / Scenario:** Manager approval for leave request.

### Test Steps (Input):

1. Employee submits Leave Request.
2. Manager logs in → Opens “Pending Approvals.”
3. Manager clicks Approve.

**Expected Result:**

- Leave Request status changes from **Pending Approval** → **Approved**.
- Employee gets Email Notification.

**Actual Result:**

Request Id	User	From Date	To Date	Reason	Status	Manager Comment	
A0010	OrgFarm EPIC	2025-09-26	2025-09-27	Festival	Approved	ok approved	Edit

## Test Case 4 – Trigger Testing

**Use Case / Scenario:** Trigger to auto-calculate Number of Days of Leave.

**Test Steps (Input):**

1. Enter From Date = 26-Sep-2025
2. Enter To Date = 27-Sep-2025

**Expected Result:** Number of Days = 2 auto-calculated by Trigger.

**Actual Result:**

From Date	To Date
2025-09-26	2025-09-27

## Conclusion:

The Leave Application Management System built on Salesforce successfully automates the entire leave management lifecycle. Employees can submit leave requests digitally, managers can approve/reject with ease, and HR can track records in real time. Automated workflows, triggers, and validation rules reduce manual errors. Dashboards and reports give management better visibility into leave trends.

**Impact:**

- **Efficiency:** Reduced manual tracking by 80%.
- **Transparency:** Clear approval process ensures fairness.
- **Scalability:** System can handle multiple departments and growing employee base.
- **User-Friendly:** Lightning pages and flows provide an easy interface.

The project demonstrates strong Salesforce Admin + Developer concepts, making it a complete CRM-based automation solution.