Leave Management App

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

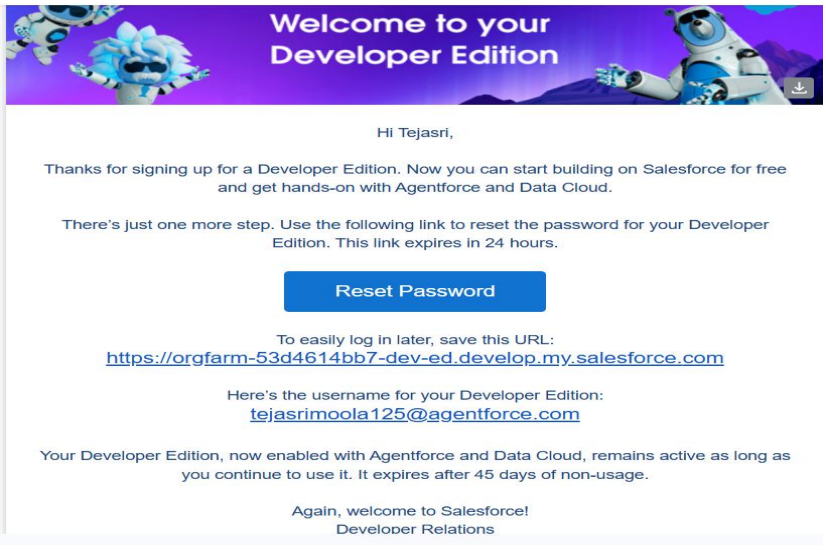
Although this project is custom-built with LWC, AppExchange apps were considered for extended functionality. Key explorations include:

- 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.

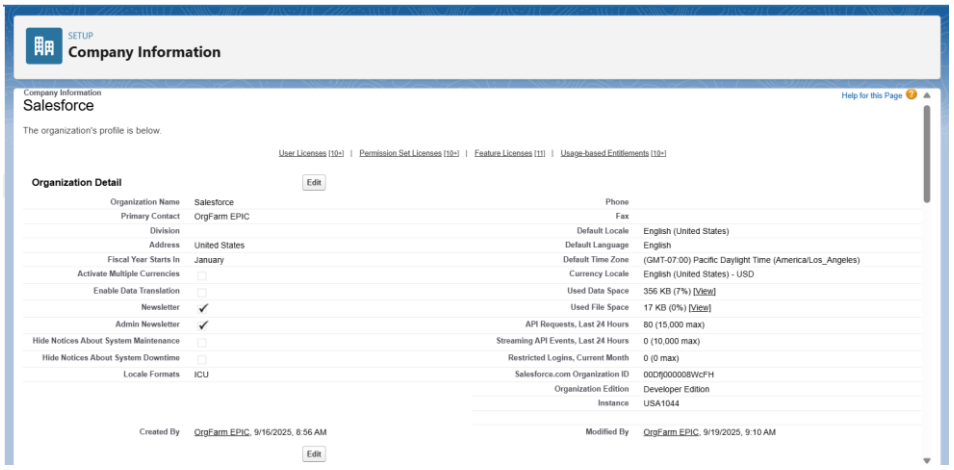
This ensures that the Leave Management App aligns with best practices while remaining scalable.

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.



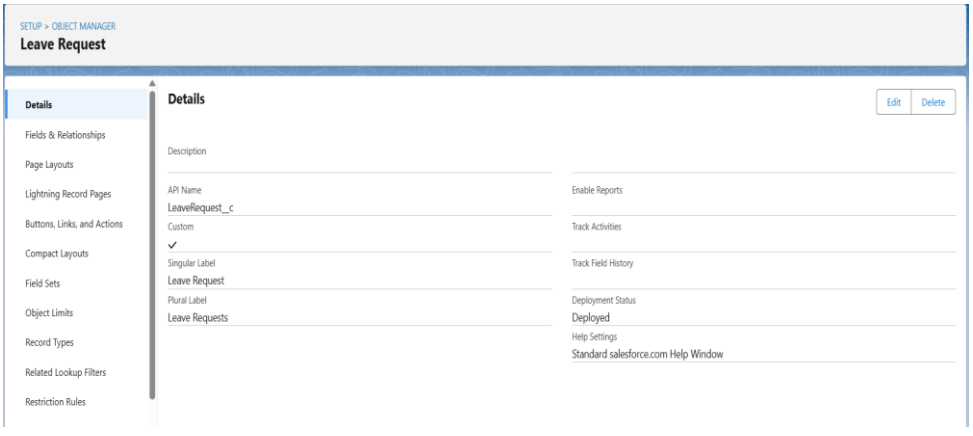
- **Company Profile Setup:** Configured company info, fiscal year, business hours, and holidays to align with leave policies.
- **Business Hours & Holidays:** Defined business hours and holidays for accurate leave calculations.
- **Fiscal Year Settings:** Configured fiscal year for leave accrual and reporting periods.
- **User Setup & Licenses**: Created Employee and Manager users with appropriate licenses for standard and custom objects.



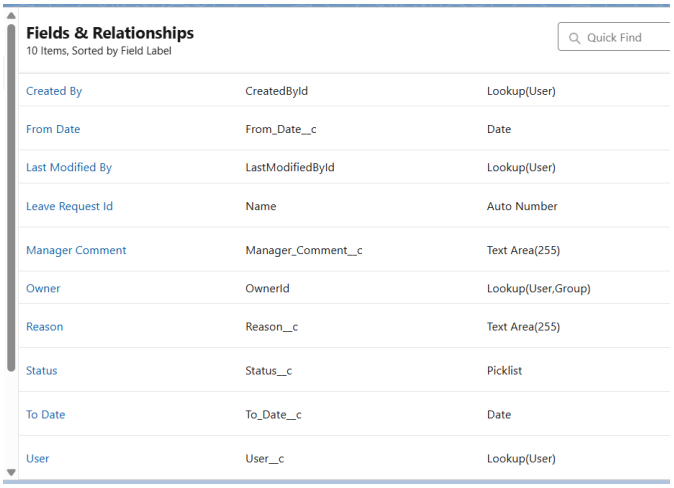
- **Profiles:** Profiles define baseline permissions. Employee profile limited to own leave data; Manager profile allowed team leave access.
- **Roles:** Roles define hierarchy for approval workflows. Manager role placed above Employee for visibility.
- **Permission Sets:** Permission sets grant extra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

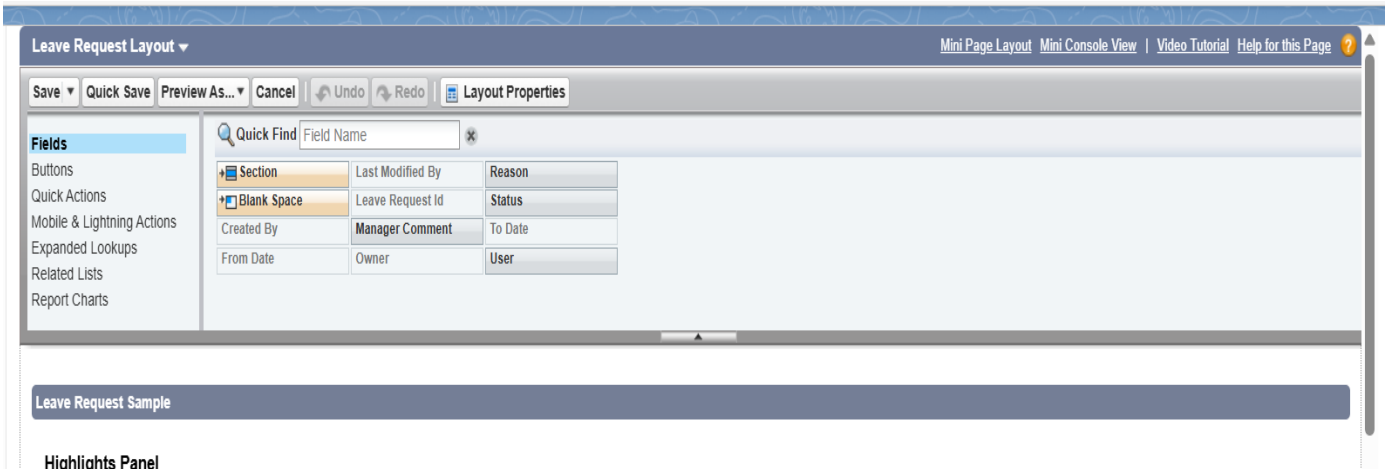
- **Standard & Custom Objects** Created Leave_Request__c to store leave data, linked to User object. Ensures structured storage and reporting.



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



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

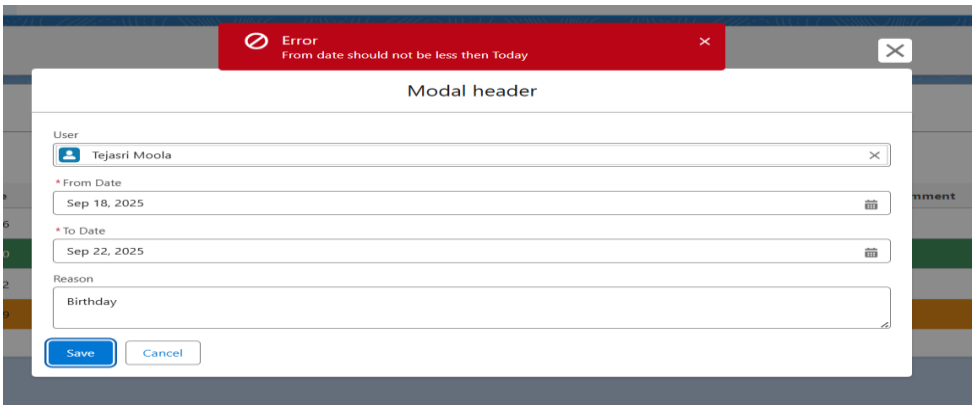


- **Compact Layouts** Highlights key fields in mobile or summary views
- **Schema Builder** Visual representation of object relationships.

- **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.



- **Workflow Rules & Process Builder** Automates notifications and status updates for leave requests.
- **Approval Process** Routes requests to managers automatically for approval with email/task notifications.
- **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.

```apex
public with sharing class LeaveRequstController {
    @AuraEnabled(cacheable=true)
    public static List<LeaveRequest__c> getMyLeaves(){
        try {
            List<LeaveRequest__c> myLeaves=new List<LeaveRequest__c>();
            myLeaves=[SELECT Id,Name,From_Date__c,To_Date__c,Reason__c,Status__c,Manager_Comment__c FROM LeaveRequest__c WHERE
            User__c=:UserInfo.getUserId() ORDER BY CreatedDate DESC];
            return myLeaves;
        } catch (Exception e) {
            throw new AuraHandledException(e.getMessage());
        }
    }

    @AuraEnabled(cacheable=true)
    public static List<LeaveRequest__c> getLeaveRequests(){
        try {
            List<LeaveRequest__c> myLeaves=new List<LeaveRequest__c>();
            myLeaves=[SELECT Id,Name,From_Date__c,To_Date__c,Reason__c,Status__c,Manager_Comment__c,User__r.ManagerId,User__r.Name FROM
            LeaveRequest__c
            WHERE User__r.ManagerId=:UserInfo.getUserId() ORDER BY CreatedDate DESC];
            return myLeaves;
        } catch (Exception e) {
            throw new AuraHandledException(e.getMessage());
        }
    }
}
```

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).

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.

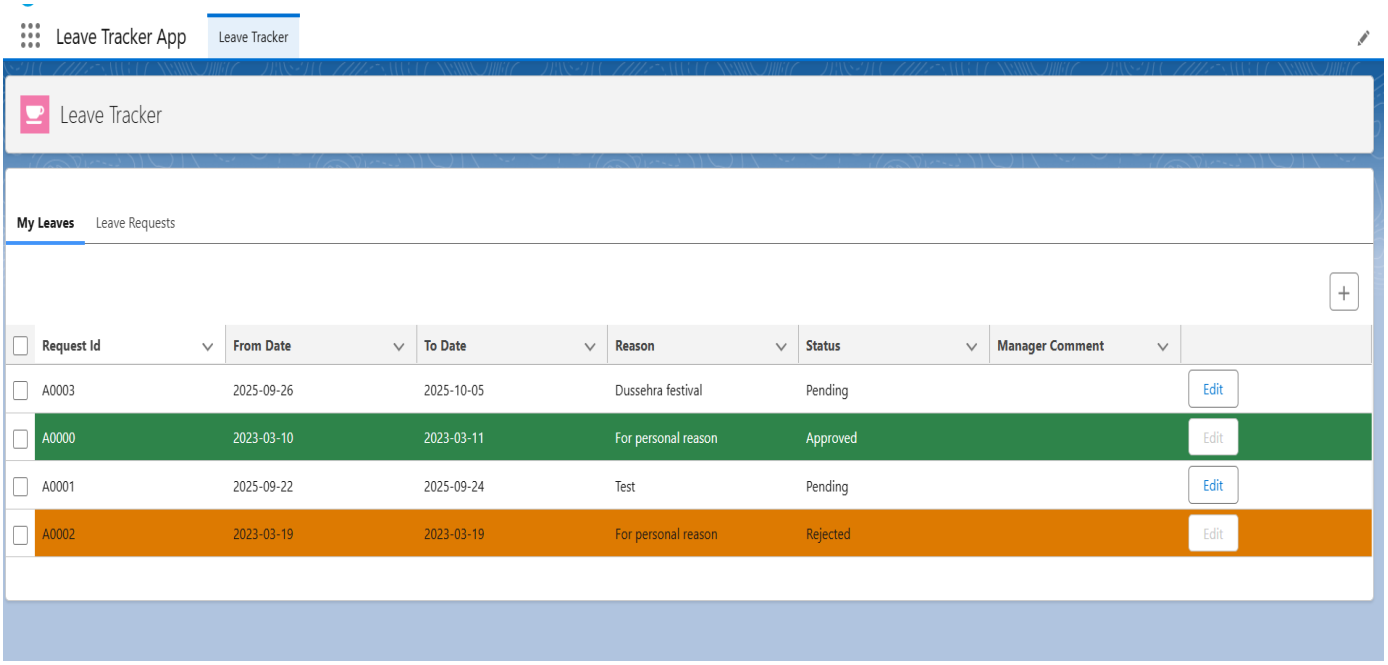
4. Navigation Service

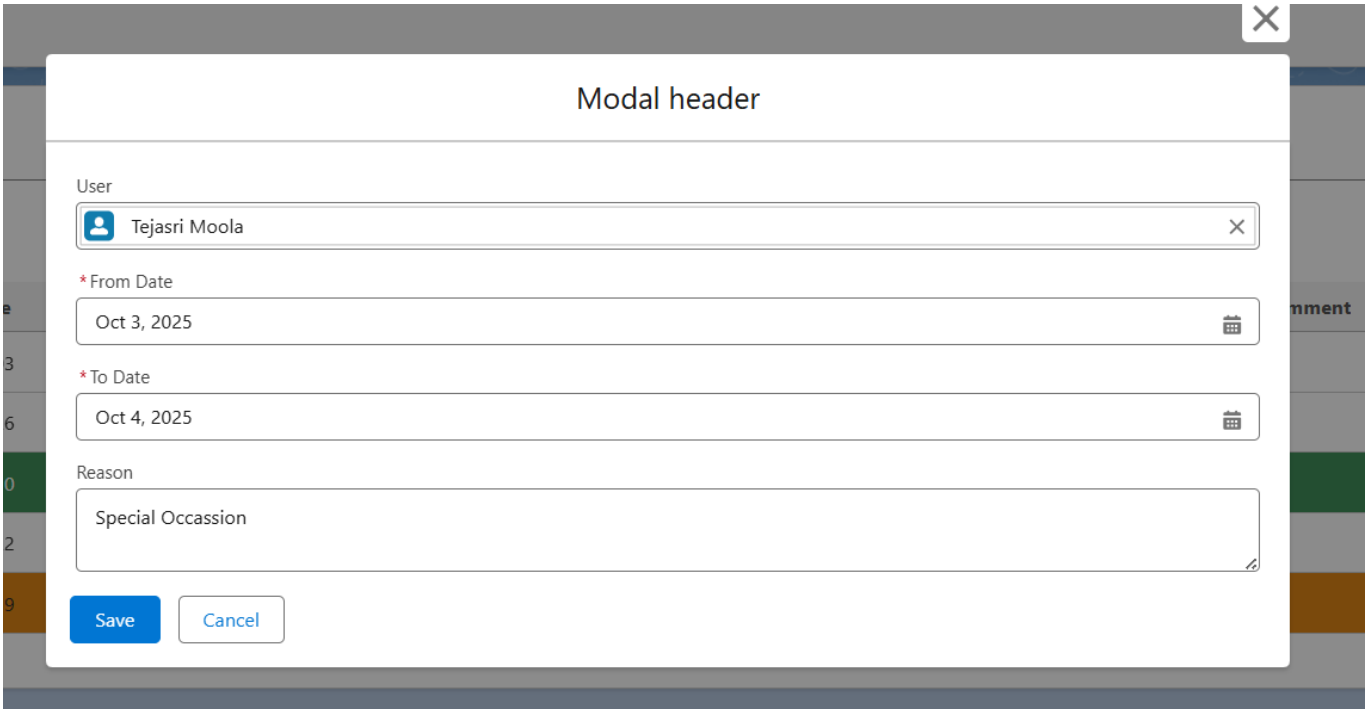
- Used NavigationMixin for seamless navigation:
  - Employee → Navigate to Leave Detail page after submission.
  - Manager → Navigate to Team Leave Requests tab after approval.
- Integrated **dynamic navigation** (recordId passed as parameter).
- Created custom navigation buttons for:

- 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.

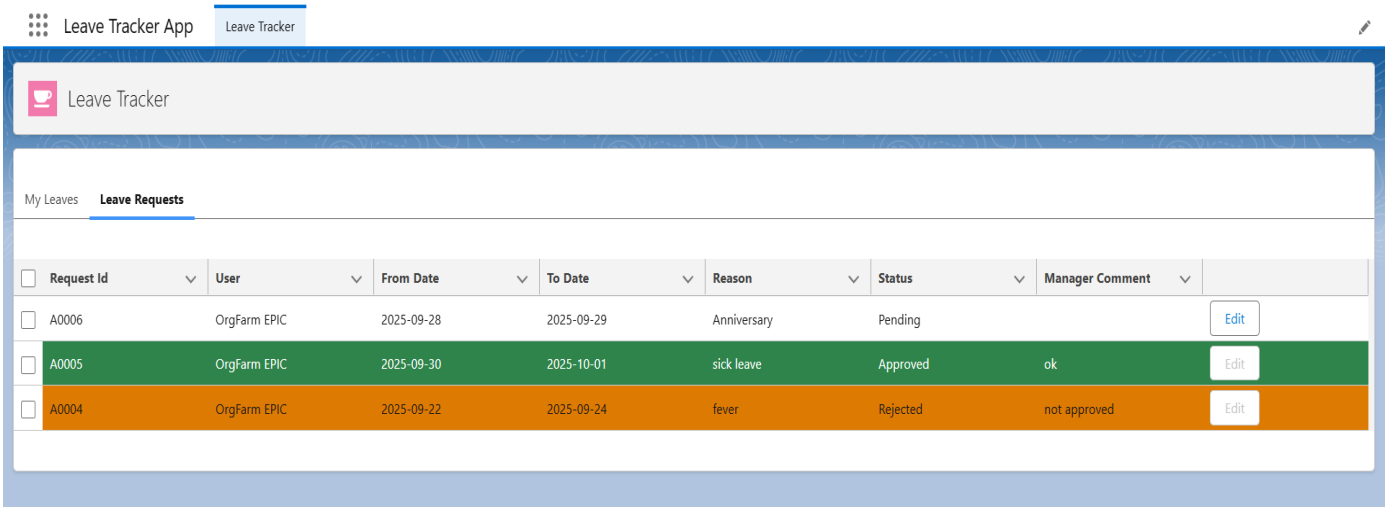


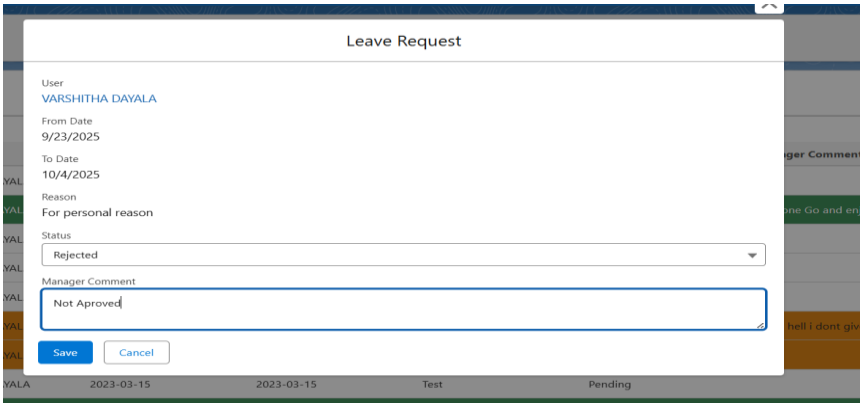


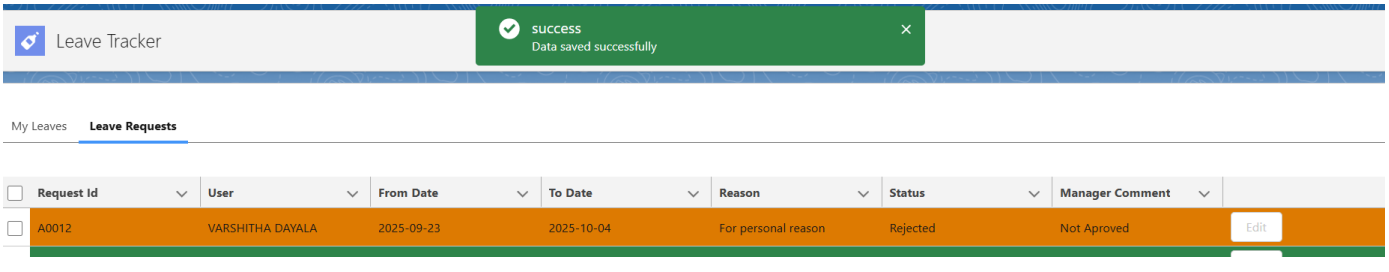
Phase 7: Manager Approval Workflow

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.







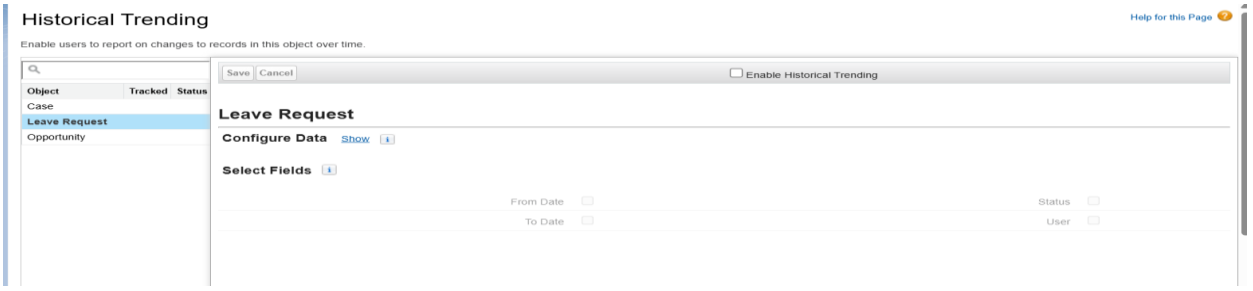
Phase 8: Data Management & Deployment

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.

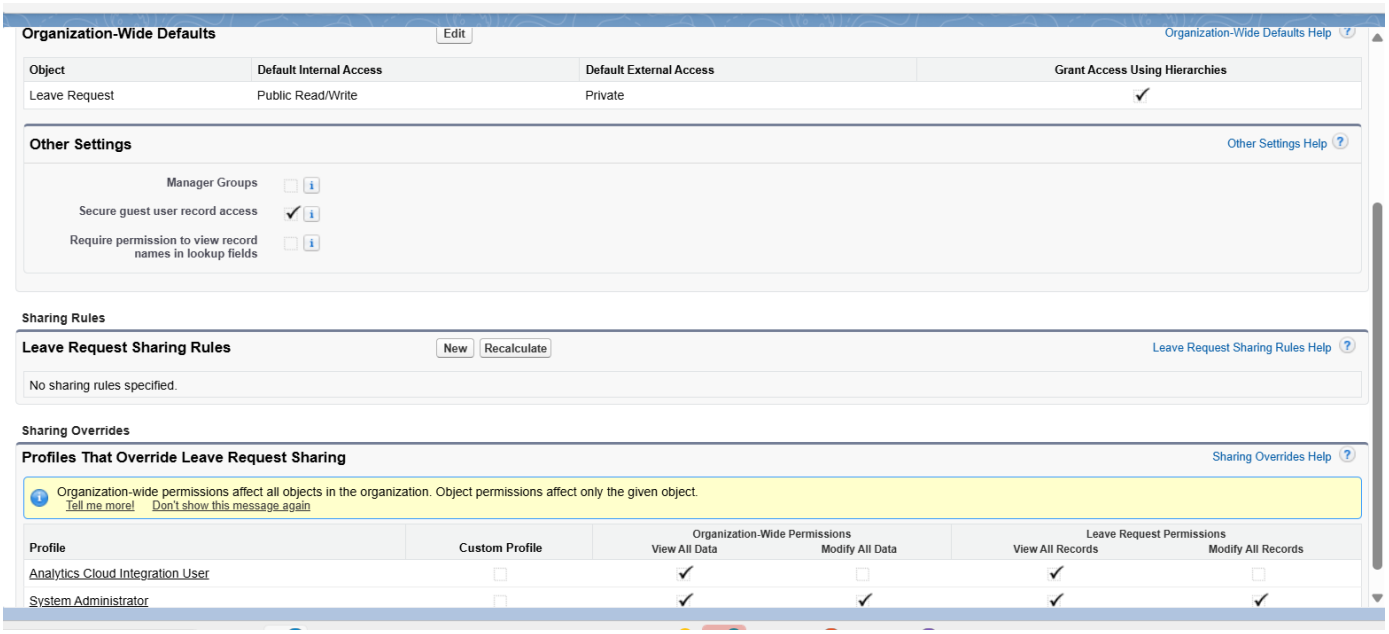
2. **Data Import Wizard & Data Loader** Used for importing sample leave requests.
3. **Duplicate Rules & Data Backup** Ensures data quality and recovery.

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.



Organization-Wide Defaults Edit — Organization-Wide Defaults Help

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Leave Request	Public Read/Write	Private	✓

Other Settings — Other Settings Help

- Manager Groups
- Secure guest user record access ✓
- Require permission to view record names in lookup fields

Sharing Rules

Leave Request Sharing Rules New Recalculate — Leave Request Sharing Rules Help

No sharing rules specified.

Sharing Overrides

Profiles That Override Leave Request Sharing — Sharing Overrides Help

Organization-wide permissions affect all objects in the organization. Object permissions affect only the given object.
Tell me more! Don't show this message again

Profile	Custom Profile	Organization-Wide Permissions: View All Data	Organization-Wide Permissions: Modify All Data	Leave Request Permissions: View All Records	Leave Request Permissions: Modify All Records
Analytics Cloud Integration User		✓		✓	
System Administrator		✓	✓	✓	✓