Salesforce-CRM-Leave-Management-App

Project Implementation Phases Documentation **-By Harshit S**

Phase 1: Problem Understanding & Industry Analysis

Problem Statement:

Traditional methods of managing employee leave, such as spreadsheets or email-based processes, are often inefficient, prone to error, and lack transparency. This can lead to administrative overhead for HR departments and a poor experience for employees. There is a clear need for a unified, secure, and automated system that provides real-time visibility and a seamless workflow for all stakeholders.

- Installed **Visual Studio Code (VS Code)** together with **Salesforce CLI** and the necessary extensions.
- Used the command palette (**Ctrl + Shift + P**) to generate a new Salesforce project.
- Chose the **"SFDX: Create Project"** option with the standard template
- Assigned the project name **"LeaveTrackerApp."**
- Connected to the Salesforce org through the command palette using **SFDX: Authorize an Org**.
- Set the alias to **"LiveProject"** with default developer org preferences.
- Confirmed the successful connection of the org inside **VS Code**.

Phase 2: Custom Object Creation

Objective: Build the **Leave_Request__c** object to store and manage leave request information.

- Retrieved the **Leave_Request__c** object metadata from the given GitHub repository.
- Placed the object folder inside the project path: **force-app/main/default/objects**.
- Deployed the object to the Salesforce org using **SFDX: Deploy Source to Org**.
- Checked and confirmed the object in Salesforce **Setup → Object Manager**.
- Added custom fields:
    - **From_Date__c** (Date)
    - **To_Date__c** (Date)
    - **Reason__c** (Text)
    - **Status__c** (Picklist: Pending, Approved, Rejected)
    - **Manager_Comment__c** (Text)
    - **User__c** (Lookup → User)
- Updated object and field-level permissions for the **System Administrator** profile manually through **Setup**.

Phase 3: Apex Class Development

Objective: Implement Apex classes to manage data operations and generate sample records.

- Added two Apex classes from the provided GitHub repository: **LeaveRequestSampleData** and **LeaveRequestController**.
- **LeaveRequestSampleData.cls:**

    - Includes a static method **createData()** that creates three sample leave requests for the logged-in user.
    - Executed this method using **SFDX: Execute Anonymous Apex with Currently Selected Text** in VS Code to insert dummy data.
- **LeaveRequestController.cls:**
    - Provides two **@AuraEnabled** methods:
        - **getMyLeaves():** Fetches leave requests for the current user, returning fields **Id, Name, From_Date__c, To_Date__c, Reason__c, Status__c, and Manager_Comment__c**.
        - **getLeaveRequests():** Retrieves leave requests where the current user is the manager, including extra fields such as **User__r.Name** and **User__r.ManagerId**.
- Deployed both classes to the Salesforce org using **SFDX: Deploy Source to Org**.

Phase 4: Lightning Web Component Development

Objective: Build LWC components to design the application's user interface.

- Created three Lightning Web Components using **SFDX: Create Lightning Web Component**:
    - **leaveTracker** → Parent component to contain and manage tab navigation.
    - **myLeaves** → Child component to display the **"My Leaves"** tab.
    - **leaveRequest** → Child component to display the **"Leave Requests"** tab.
- Modified the **leaveTracker** component's **meta.xml** file by:
    - Setting **isExposed = true**.
    - Adding **lightning__AppPage** as a target.
- Deployed all three components to the Salesforce org successfully.

Phase 5: Lightning App and Page Configuration

Objective: Set up a Lightning App and App Page to display the LWC components.

- Opened **Setup → App Manager** and created a new Lightning App named **"Leave Tracker App."**
- Assigned access to the **System Administrator** profile and skipped non-essential options.
- Using **Lightning App Builder**, created a new App Page titled **"Leave Tracker"** with a **single-column layout**.
- Placed the **leaveTracker** component onto the page, then saved and activated it.
- Linked the page to the **"Leave Tracker App"** as a new tab.
- Verified functionality via the **App Launcher**, confirming that the **leaveTracker** component rendered correctly.

Phase 6: Tab Implementation in Parent Component

Objective: Enhance the **leaveTracker** component by enabling tab navigation between **"My Leaves"** and **"Leave Requests."**

- Edited **leaveTracker.html** to include a **lightning-tabset** with two tabs:
    - **"My Leaves"** → Loads the **c-my-leaves** child component.
    - **"Leave Requests"** → Loads the **c-leave-request** child component.
- Added labels and temporary placeholder text inside **myLeaves.html** and **leaveRequest.html** to support initial testing.
- Redeployed the updated components to the Salesforce org.
- Confirmed that the tab functionality worked as expected in the application.

Phase 7: My Leaves Tab Implementation

Objective: Develop the **"My Leaves"** tab with a data table, modal form, and client-side validations.

- **HTML (myLeaves.html):**
  - Created a **lightning-card** containing a **lightning-datatable** to display leave records.
  - Configured the datatable with **key-field="Id"**, **data={myLeaves}**, and **columns={columns}**.
  - Added a conditional section using **lwc:if** and a **noRecords** getter to show **"No records found"** when the dataset is empty.
  - Designed a modal popup with **SLDS markup** and a **lightning-record-edit-form** for adding or editing leave requests.
  - Included fields: **User__c, From_Date__c, To_Date__c, and Reason__c**, along with **Save** and **Cancel** buttons.
  - Added a **lightning-button-icon** for creating new leave requests.
- **JavaScript (myLeaves.js):**
  - Imported the **getMyLeaves** Apex method and wired it to retrieve leave request data.
  - Defined datatable columns, including an **edit row action** with conditional disabling when **Status__c** is not editable.
  - Applied row-level styling using **cellAttributes** with SLDS classes (**slds-theme_success** for Approved, **slds-theme_warning** for Rejected).
  - Managed modal visibility with a **showModalPopup** property.
  - Set the current user as the default for **User__c** using **@salesforce/user/Id**.
  - Implemented client-side validations in **onsubmit** to ensure:
    - **From_Date__c** is not earlier than today.
    - **From_Date__c** is not later than **To_Date__c**.
  - Used **refreshApex** to auto-refresh the datatable after saving.
  - Handled **onsuccess** to display a success toast and close the modal.
  - Implemented a **rowaction handler** to pre-fill the form with existing record data during edits.
- **Deployment & Testing:**
  - Deployed the component to the org.
  - Verified that the data grid loaded correctly, validations were enforced, and the modal popup worked as intended

Phase 8: Leave Requests Tab Implementation

Objective: Develop the **"Leave Requests"** tab for managers to review and update leave requests from subordinates.

- **HTML (leaveRequest.html):**
  - Based on **myLeaves.html**, removed the **add button** since managers cannot create new requests.
  - Updated the modal title to **"Leave Request Details."**
  - Displayed **User__c, From_Date__c, To_Date__c, and Reason__c** as **lightning-output-field** for read-only purposes.
  - Added **Status__c** and **Manager_Comment__c** as **lightning-input-field** to allow manager edits.
  - Removed the **onsubmit** handler because no client-side validations were required.
- **JavaScript (leaveRequest.js):**

    - Switched from **getMyLeaves** to **getLeaveRequests** Apex method to fetch subordinate leave data.
    - Renamed properties from **myLeaves** to **leaveRequests** for clarity and consistency.
    - Introduced a **Username** column by mapping **User__r.Name** to a new **username** property.
    - Added a public **refreshGrid** method with the **@api** decorator to refresh the datatable programmatically.
- **Deployment & Testing:**
    - Deployed the component to the Salesforce org.
    - Verified that the datatable correctly displayed subordinate leave requests and that the modal allowed managers to update **status** and **comments**.

Phase 9: Inter-Component Communication

Objective: Enable the **"Leave Requests"** tab to refresh automatically whenever a new leave is created in the **"My Leaves"** tab.

- In **myLeaves.js**, created a custom event **leaverequestsave** and dispatched it in the **onsuccess** handler after saving a new leave.
- In **leaveTracker.html**, added an **onleaverequestsave** listener on the **c-my-leaves** component.
- In **leaveTracker.js**, implemented the handler to invoke the **refreshGrid** method of the **c-leave-request** component using the **lwc:ref** attribute.
- Fixed an initial mistake where **lwc:ref** was incorrectly applied to **c-my-leaves** instead of **c-leave-request**.
- Deployed and tested the components, confirming that the **"Leave Requests"** grid refreshed automatically after a new leave was saved.

Phase 10: Testing and Finalization

Objective: Conduct thorough testing and finalize the **Leave Tracker App**.

- **Testing the "My Leaves" tab:**
    - Verified that the datatable displayed records correctly with conditional row colors (**green for Approved**, **yellow for Rejected**).
    - Confirmed that the **edit button** was disabled for leave requests not in **Pending** status.
    - Tested the modal popup for adding/editing requests, ensuring pre-filled values and client-side validations worked.
    - Ensured the datatable automatically refreshed after saving a new leave request.
- **Testing the "Leave Requests" tab:**
    - Verified that the datatable showed subordinate leave requests with the **Username** column.
    - Confirmed read-only fields (**User__c, From_Date__c, To_Date__c, Reason__c**) and editable **Status__c** and **Manager_Comment__c** in the modal.
    - Ensured the grid refreshed automatically when a new leave was created in **"My Leaves."**
- Created a test user (e.g., **"Manoj"**) with the current user assigned as the manager in **Setup → Users**.
- Added additional leave requests to validate grid updates and manager approval workflow.
- Fixed minor issues, such as the **To_Date__c** field spelling in **leaveRequest.js**.
- Documented the project and confirmed that all components were successfully deployed.