Phase 2: Org Setup & Configuration

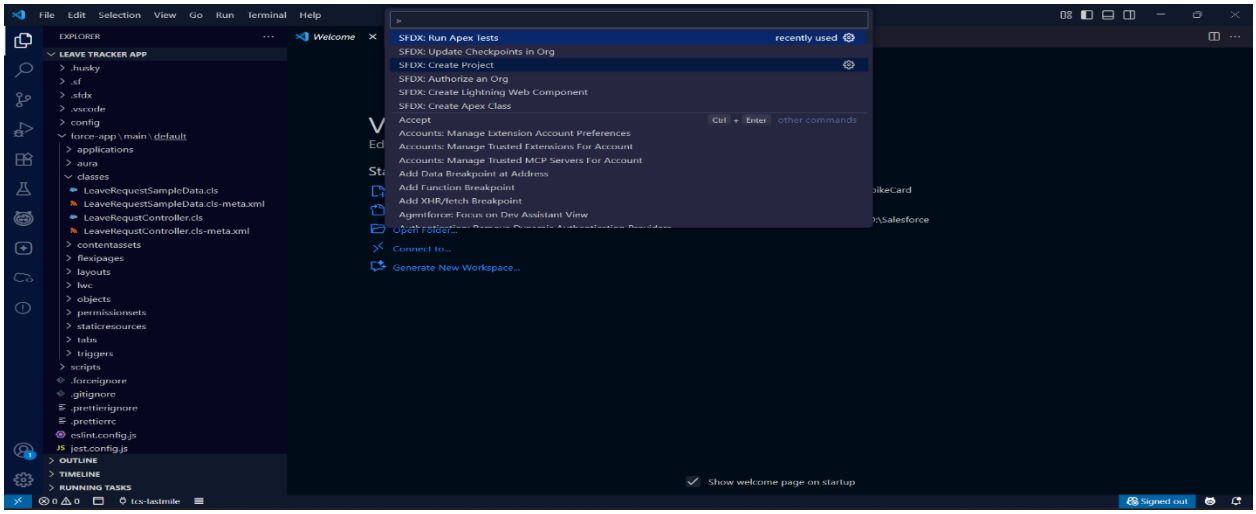
Purpose:

The purpose of this phase was to establish a secure and structured Salesforce environment where all insurance claim-related operations could be configured. This setup ensures that data security, access control, and company-wide standards are properly implemented before moving on to data modeling and automation.

(Org Setup & Configuration This phase involves preparing the Salesforce Developer Org for the application build.)

1. Salesforce Edition:
   Purpose: The purpose of this phase was to establish a secure and structured Salesforce environment where all insurance claim-related operations could be configured. This setup ensures that data security, access control, and company-wide standards are properly implemented before moving on to data modelling and automation.

   Implementation: -
   a. Developer Org/Sandbox Creation:
      • A Salesforce Developer Org was created for initial configuration and testing.
      • Alternatively, a Sandbox can be used to mirror production settings.
      To create the org, we are using the lwc (lightning web component/vs code) and also mostly for all the things and implementation like object creation/implementation we are using vs code and then deploy to the org
      Step 1: -
      - On VScode open command palatte or add (>) symbol and write (SFDX: Create A PROJECT) and name it “Leave Tracker App”

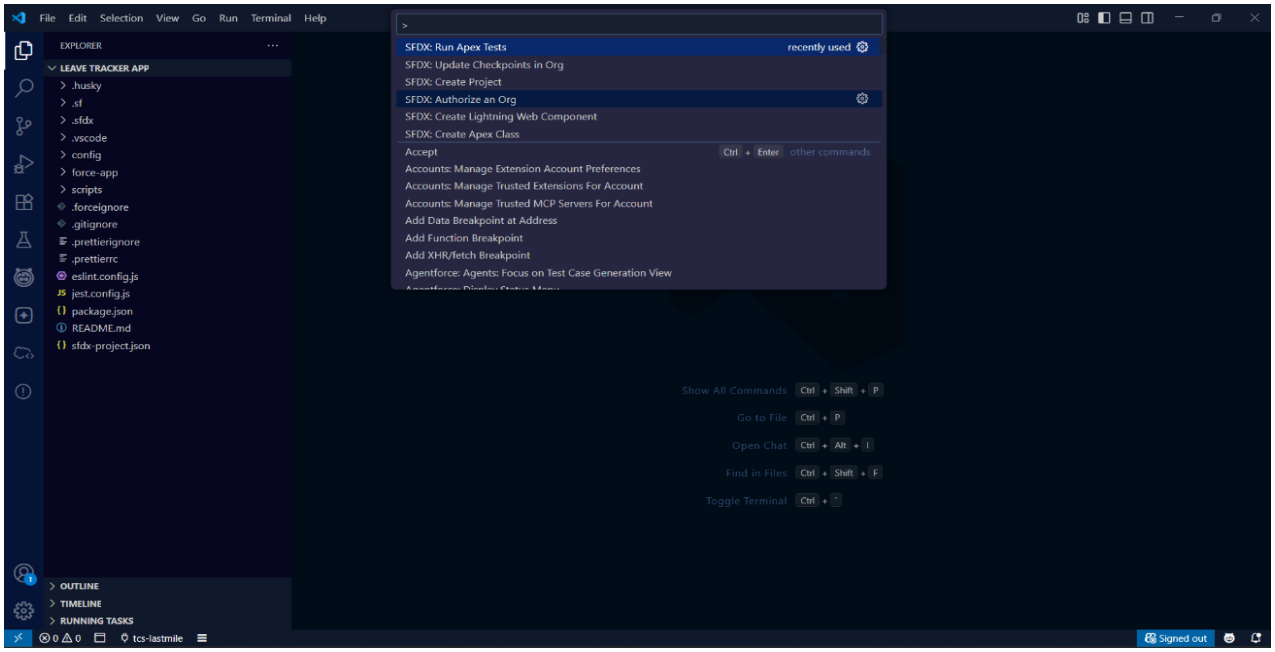


- A new project file will get created and then save it according to the system.

Step 2: -

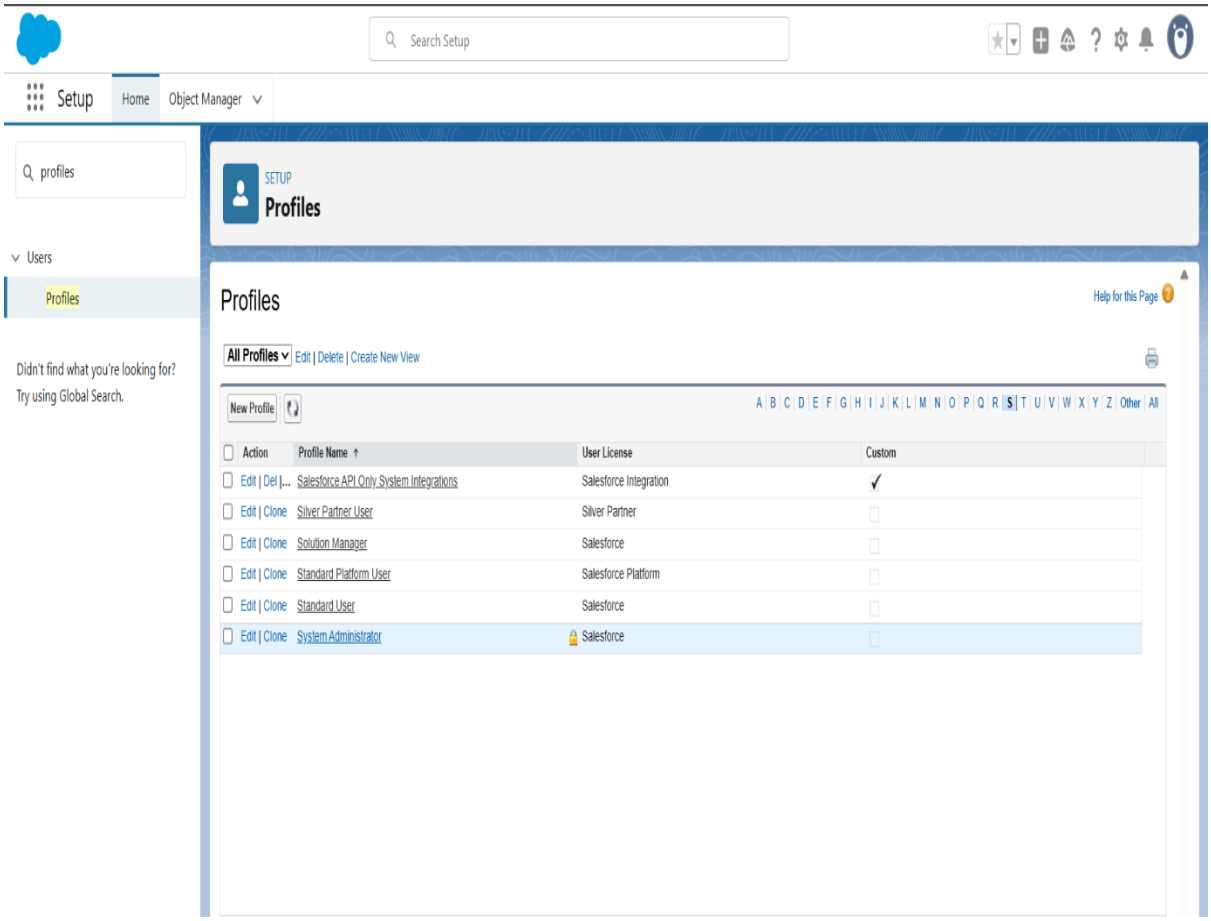
Authorize the org:

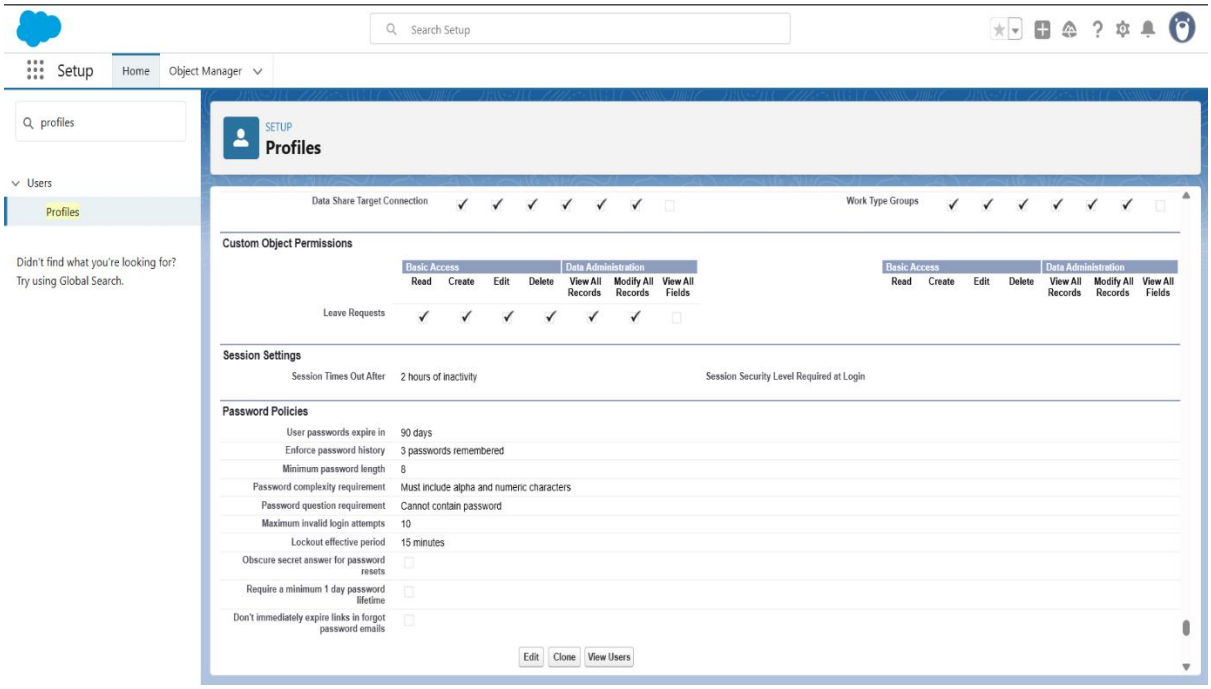
- In this we have to authorize the org to the salesforce account, for that in the same command pallatte search for (SFDX: Authorize an org).
- Press enter, then a new screen will show on web browser on that we have to login using the salesforce credentials.

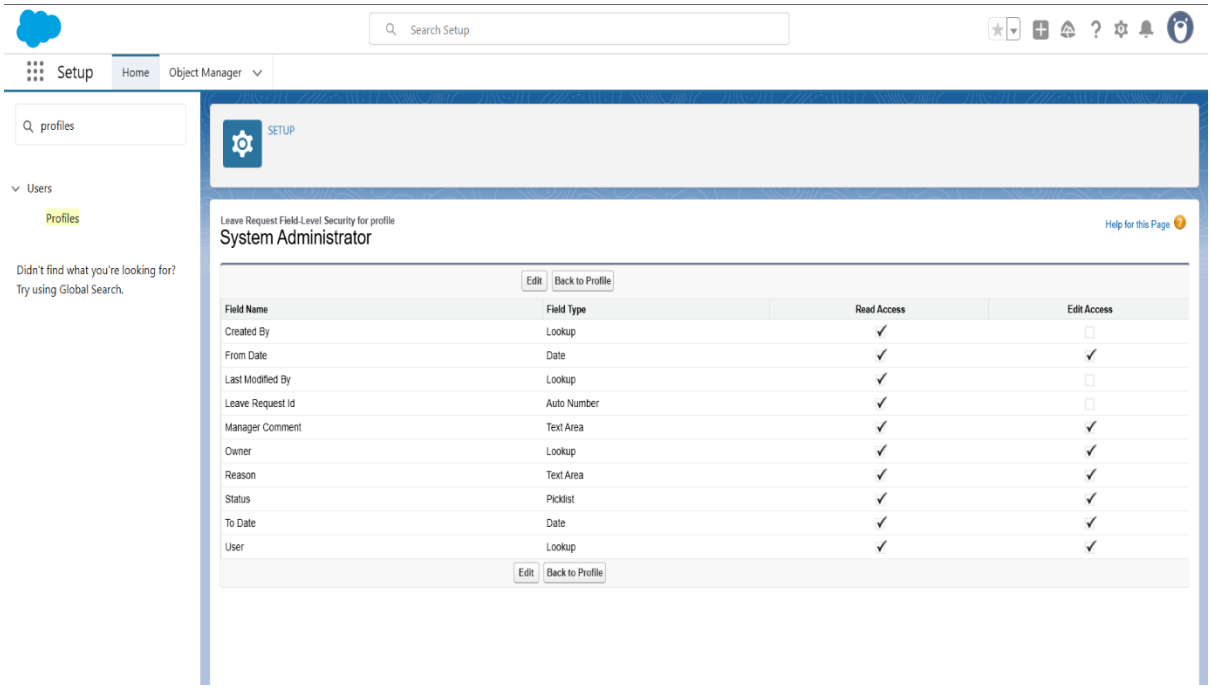


- After connecting the account to the project, it will show how many logs have been created to the vs code, so choose according to the account you want to create the project on it.
- Now you have created the project titled “Leave Tracker App” and also the “Salesforce Edition” is also comes under the dev org setup.

2. Permission Sets:
   - Customized Profiles to restrict or grant access based on responsibilities.
   - Added Permission Sets for advanced tasks like claim approvals.




Search Setup
Setup
Home
Object Manager
profiles
Users
Profiles
Didn't find what you're looking for?
Try using Global Search.
SETUP
Profiles
Data Share Target Connection
Work Type Groups
Custom Object Permissions
Basic Access
Data Administration
Read
Create
Edit
Delete
View All Records
Modify All Records
View All Fields
Leave Requests
Session Settings
Session Times Out After
2 hours of inactivity
Session Security Level Required at Login
Password Policies
User passwords expire in
90 days
Enforce password history
3 passwords remembered
Minimum password length
8
Password complexity requirement
Must include alpha and numeric characters
Password question requirement
Cannot contain password
Maximum invalid login attempts
10
Lockout effective period
15 minutes
Obscure secret answer for password resets
Require a minimum 1 day password lifetime
Don't immediately expire links in forgot password emails
Edit
Clone
View Users


Search Setup
Setup
Home
Object Manager
profiles
Users
Profiles
Didn't find what you're looking for?
Try using Global Search.
SETUP
Leave Request Field-Level Security for profile
System Administrator
Help for this Page
Edit
Back to Profile
Field Name
Field Type
Read Access
Edit Access
Created By
Lookup
From Date
Date
Last Modified By
Lookup
Leave Request Id
Auto Number
Manager Comment
Text Area
Owner
Lookup
Reason
Text Area
Status
Picklist
To Date
Date
User
Lookup
Edit
Back to Profile

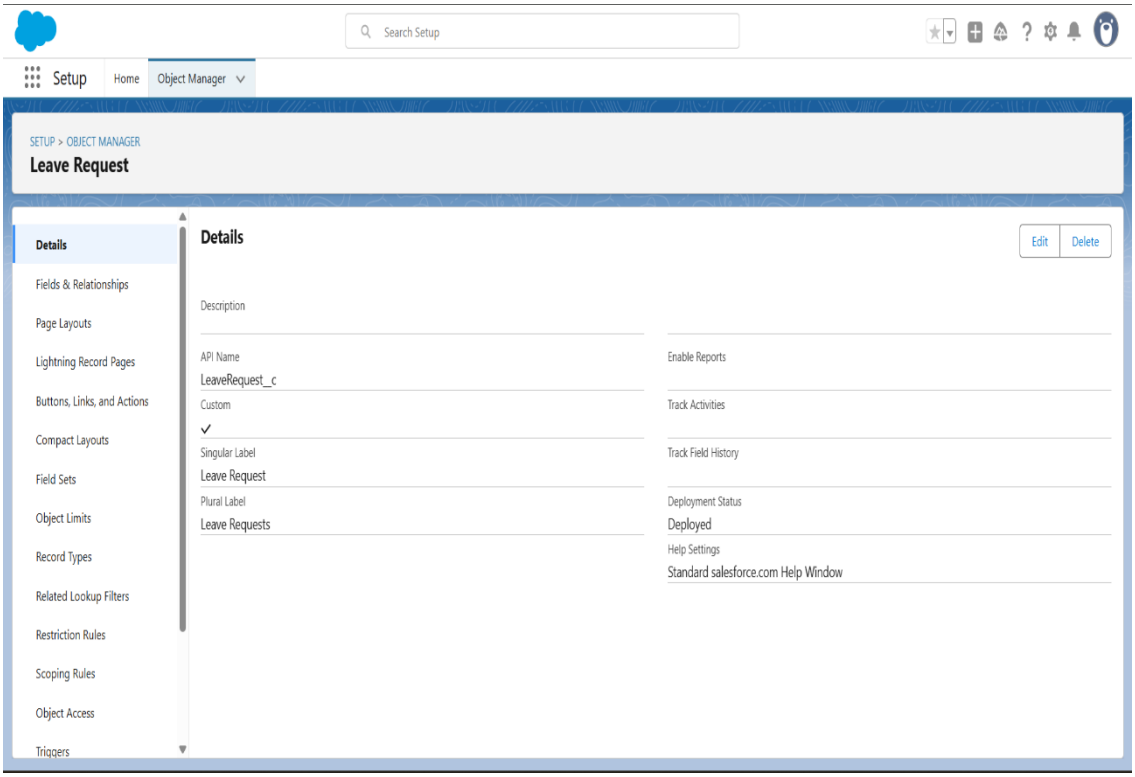
Phase 3: Data Modeling & Relationships

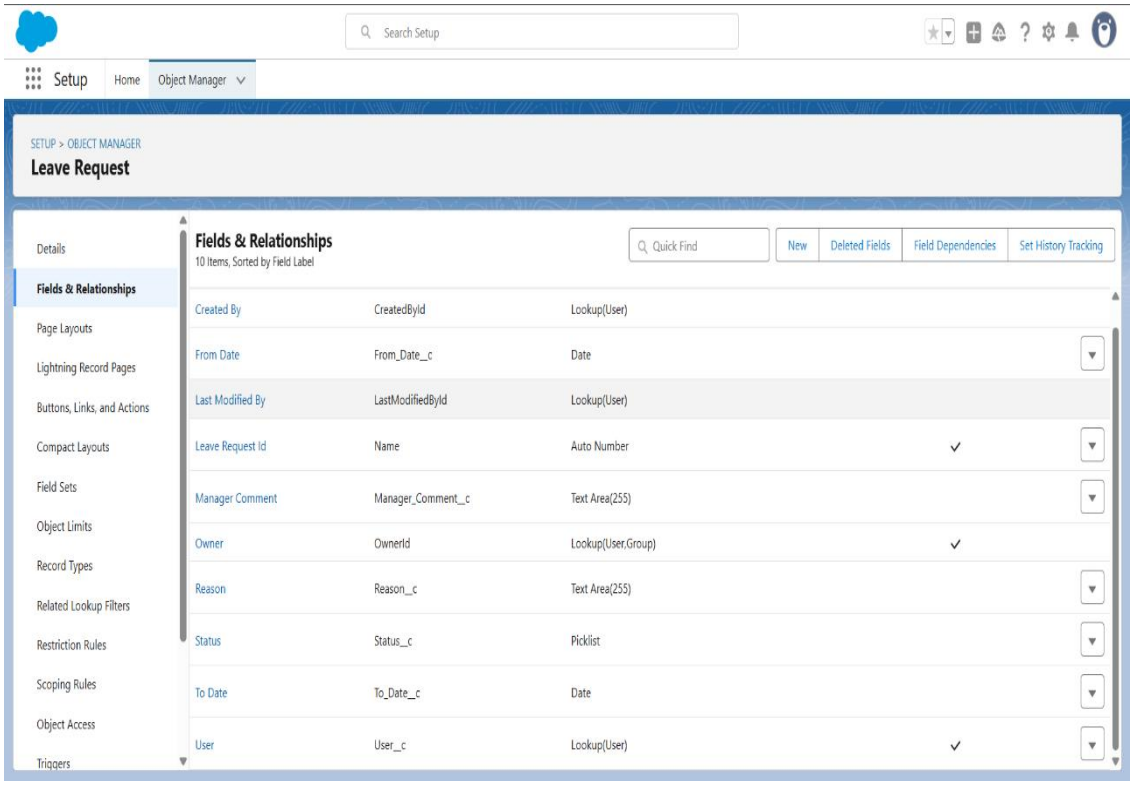
Purpose:

This phase focused on designing the underlying data model to store and manage information about policyholders, policies, and claims. A scalable and logical data structure was essential for enabling claim processing, reporting, and automation.

Implementation:

1. Custom/Standard object creation:
    a. LeaveRequest__c (includes: From_Date__c, To_Date__c, Manager_Comment__c, Reason__c, Status__c and User__c)




Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Leave Request
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Restriction Rules
Scoping Rules
Object Access
Triggers
Fields & Relationships
10 Items, Sorted by Field Label
Quick Find
New
Deleted Fields
Field Dependencies
Set History Tracking
Created By CreatedById Lookup(User)
From Date From_Date__c Date
Last Modified By LastModifiedById 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) ✓

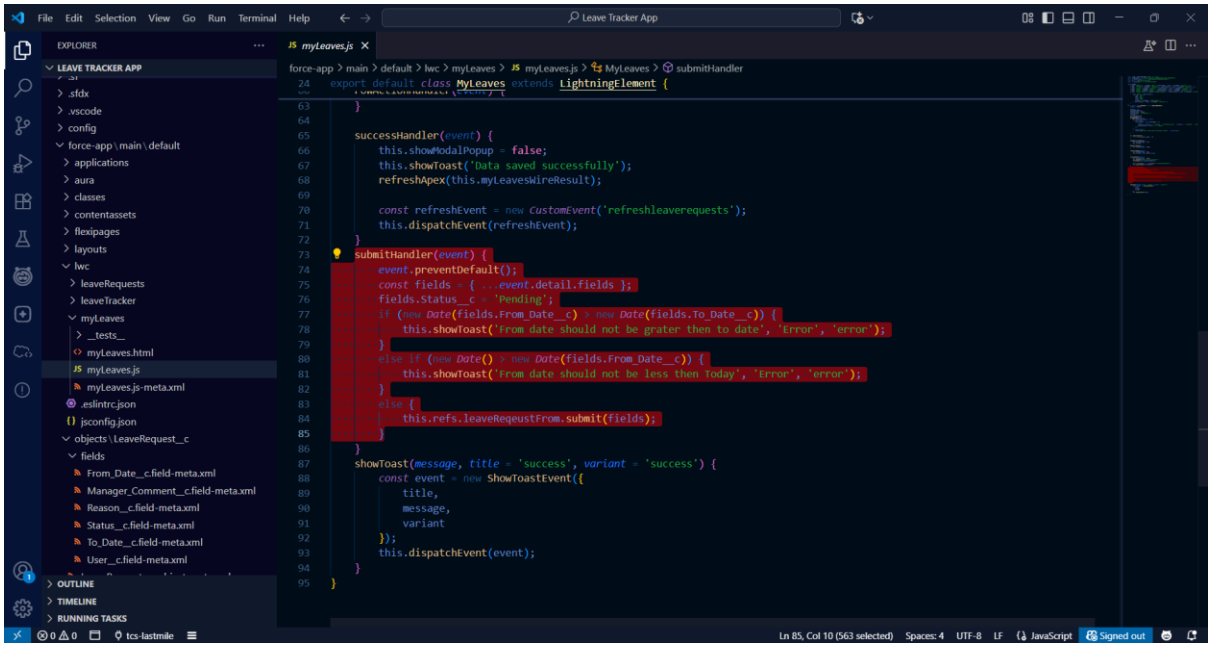
Phase 4: Process Automation (Admin) (Only client-side validation)

Purpose:

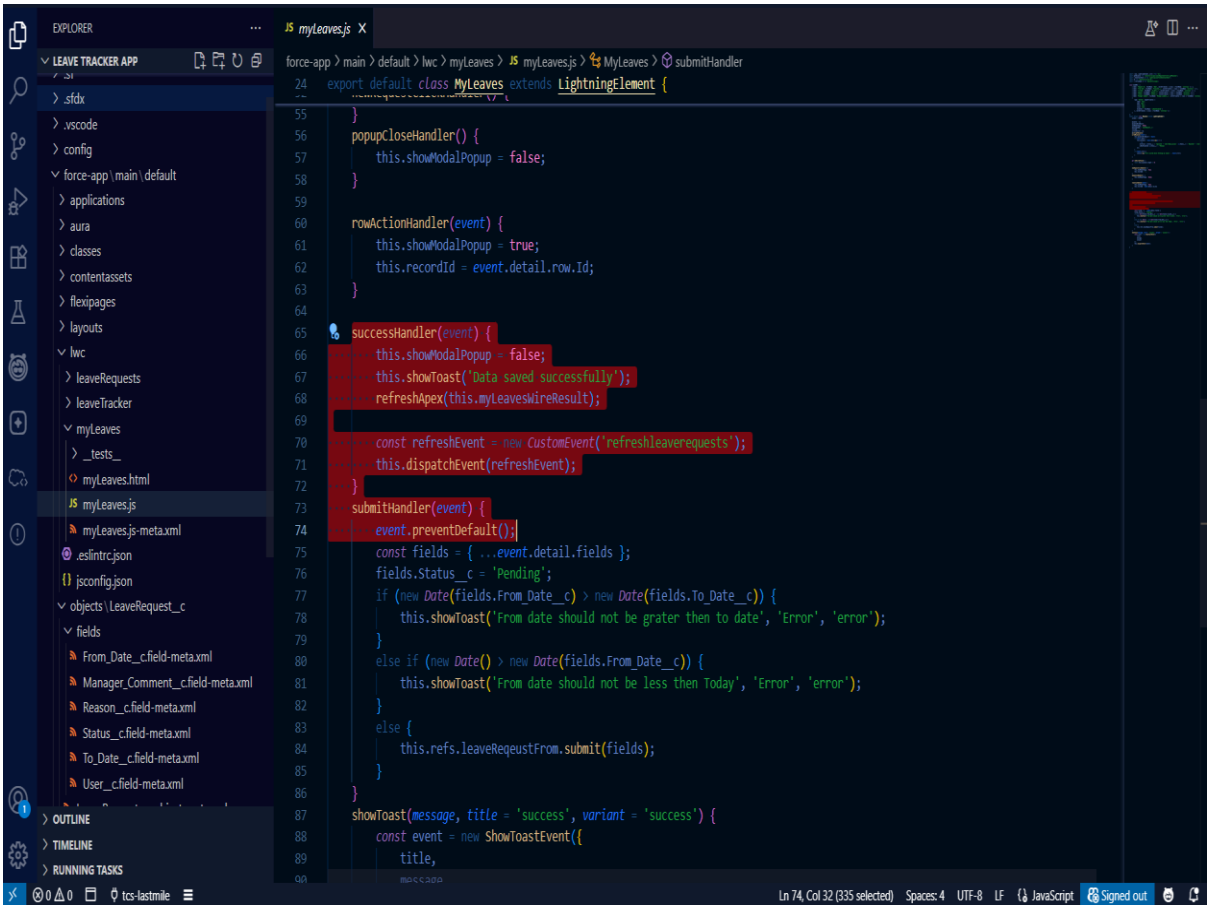
The goal of this phase was to automate business processes to improve efficiency, ensure compliance with rules, and streamline claim submission and approval workflows.

Implementation: -

1. Custom validation using form submit event:
   We are using lwc and Vs code to create the validation in the submit handler.



2. Date validation logic:
   This is for the logic for the from date to date for both MyLeaves and Request.

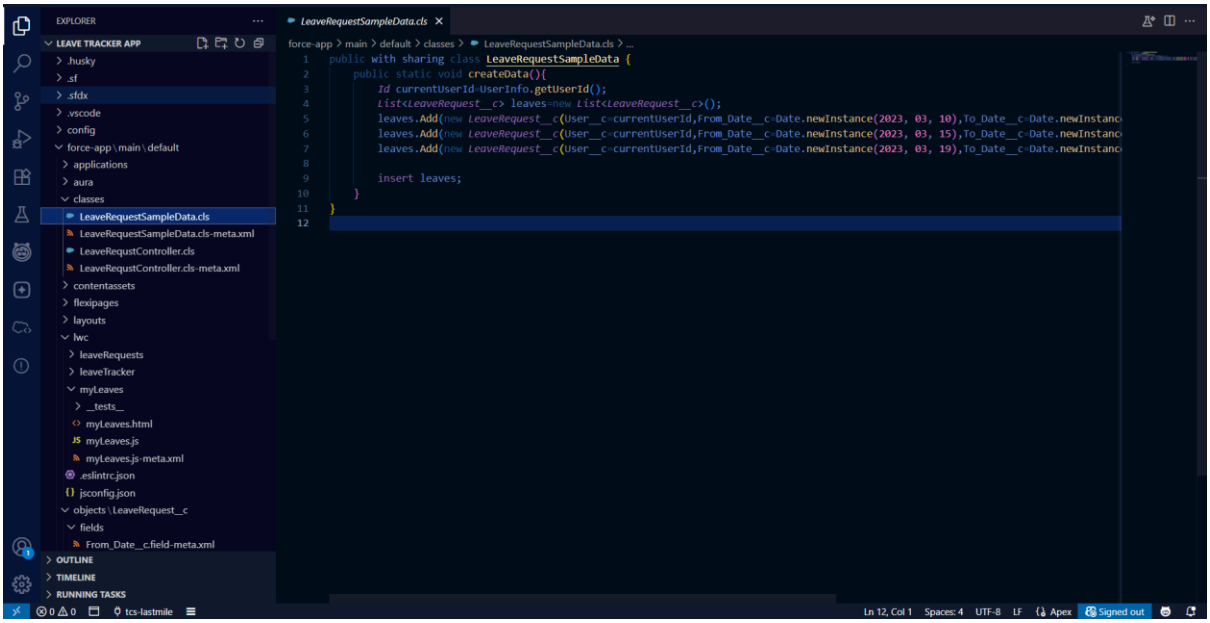

EXPLORER
LEAVE TRACKER APP
.sfdx
.vscode
config
force-app\main\default
applications
aura
classes
contentassets
flexipages
layouts
lwc
leaveRequests
leaveTracker
myLeaves
__tests__
myLeaves.html
myLeaves.js
myLeaves.js-meta.xml
.eslintrc.json
jsconfig.json
objects\LeaveRequest__c
fields
From_Date__c.field-meta.xml
Manager_Comment__c.field-meta.xml
Reason__c.field-meta.xml
Status__c.field-meta.xml
To_Date__c.field-meta.xml
User__c.field-meta.xml
OUTLINE
TIMELINE
RUNNING TASKS
myLeaves.js
force-app › main › default › lwc › myLeaves › myLeaves.js › MyLeaves › submitHandler
export default class MyLeaves extends LightningElement {
}
popupCloseHandler() {
this.showModalPopup = false;
}
rowActionHandler(event) {
this.showModalPopup = true;
this.recordId = event.detail.row.Id;
}
successHandler(event) {
this.showModalPopup = false;
this.showToast('Data saved successfully');
refreshApex(this.myLeavesWireResult);
const refreshEvent = new CustomEvent('refreshleaverequests');
this.dispatchEvent(refreshEvent);
}
submitHandler(event) {
event.preventDefault();
const fields = { ...event.detail.fields };
fields.Status__c = 'Pending';
if (new Date(fields.From_Date__c) > new Date(fields.To_Date__c)) {
this.showToast('From date should not be grater then to date', 'Error', 'error');
}
else if (new Date() > new Date(fields.From_Date__c)) {
this.showToast('From date should not be less then Today', 'Error', 'error');
}
else {
this.refs.leaveReqeustFrom.submit(fields);
}
}
showToast(message, title = 'success', variant = 'success') {
const event = new ShowToastEvent({
title,
tcs-lastmile
Ln 74, Col 32 (335 selected)
Spaces: 4
UTF-8
LF
JavaScript
Signed out

Phase 5: Apex Programming (Developer)

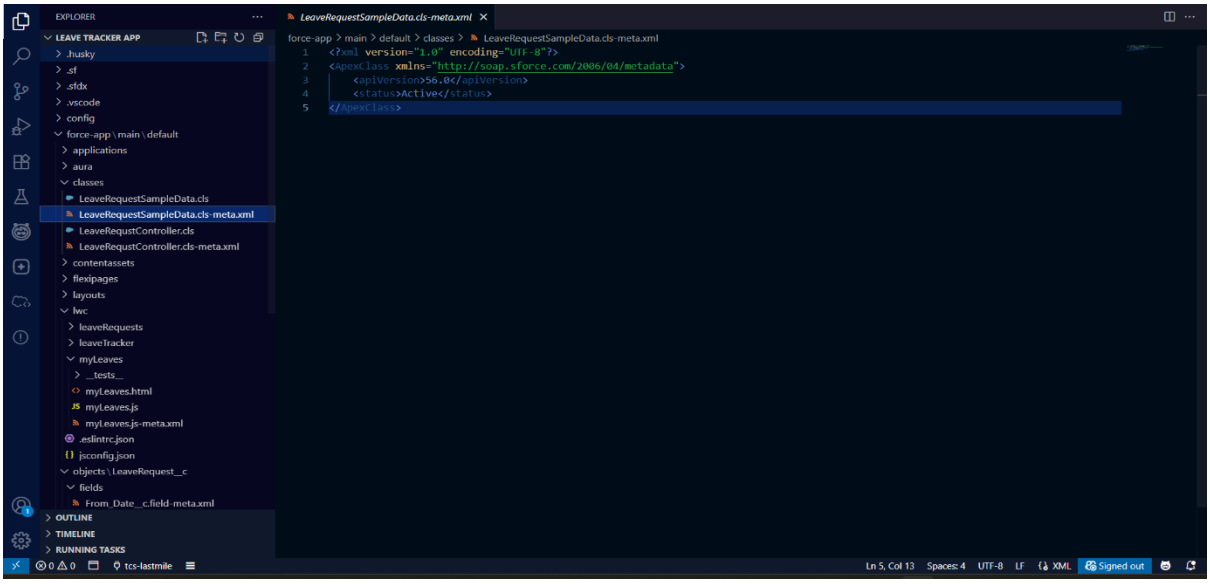
Purpose: Custom logic was required to extend Salesforce's standard functionality. Apex programming allowed automation of tasks not possible with declarative tools.

Implementation:

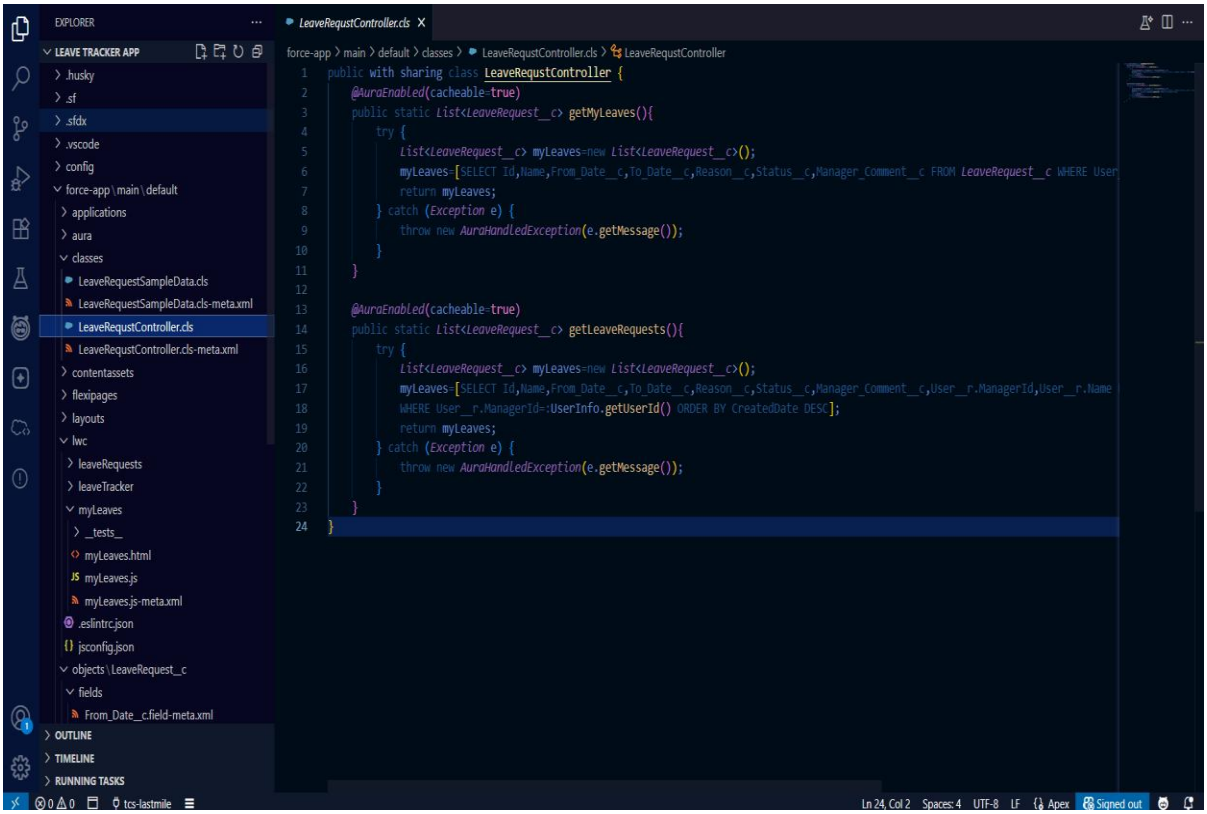
1. Classes & Object (Apex classes setup:
    a. LeaveRequestSampleData
    b. LeaveRequestController



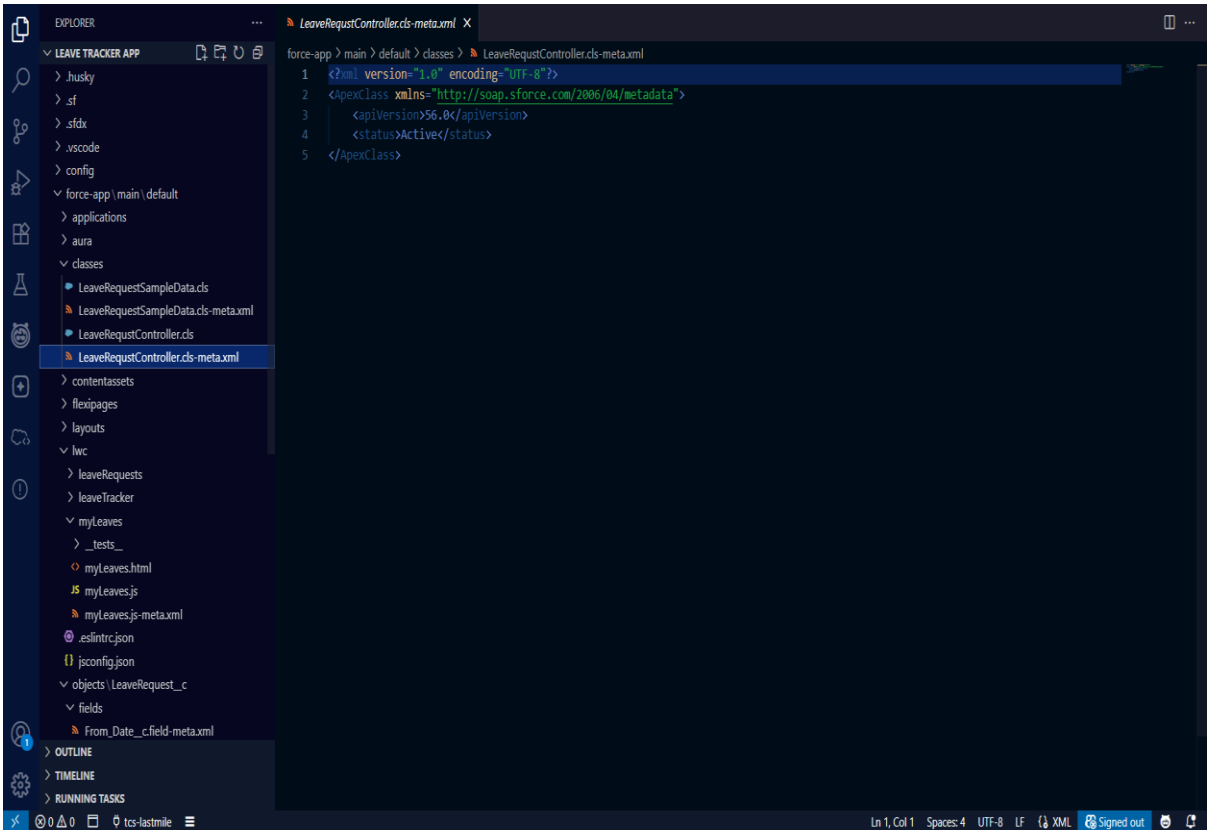
LeaveRequestSampleData.cls



LeaveRequestSampleData.cls-meta.xml

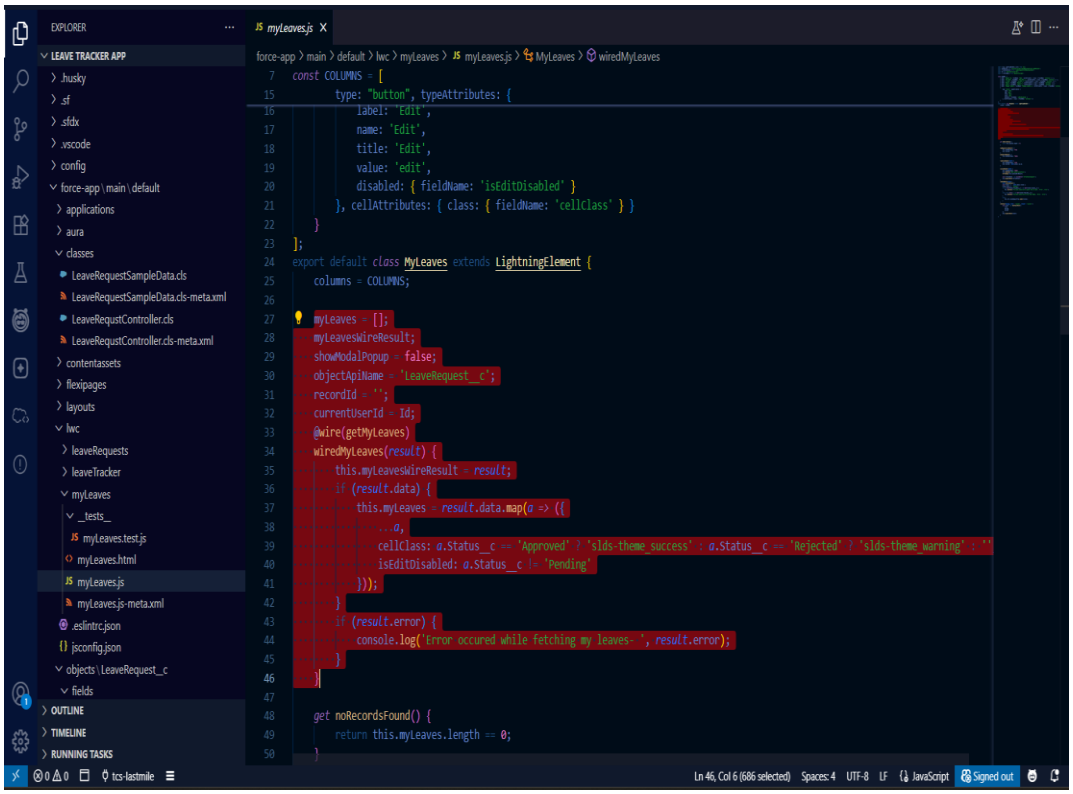


LeaveRequestController.cls



LeaveRequestController.cls-meta.xml

- Now to connect to the objects/classes the (@wire) is used to connect



(note: the LeaveRequestSampleData is a dummy data for testing the app)

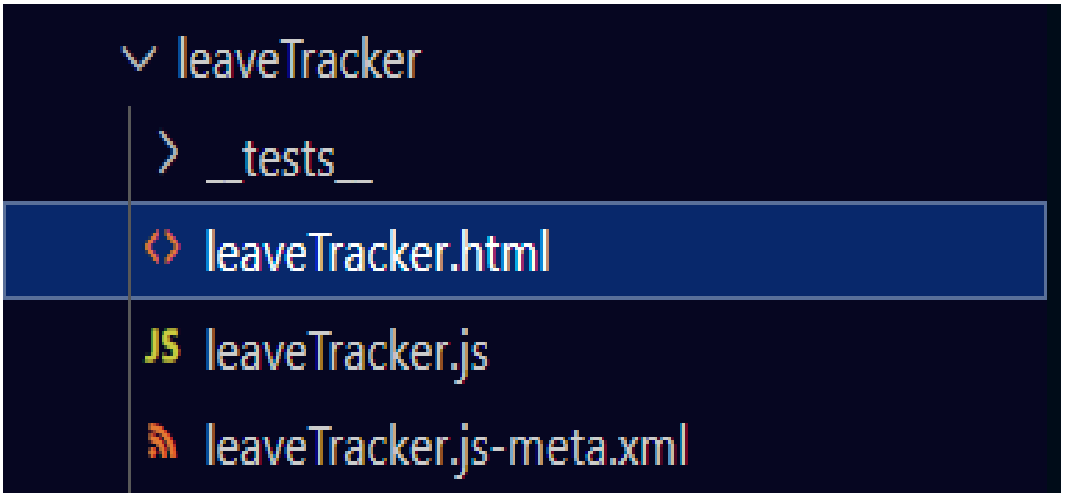
Phase 6: User Interface Development

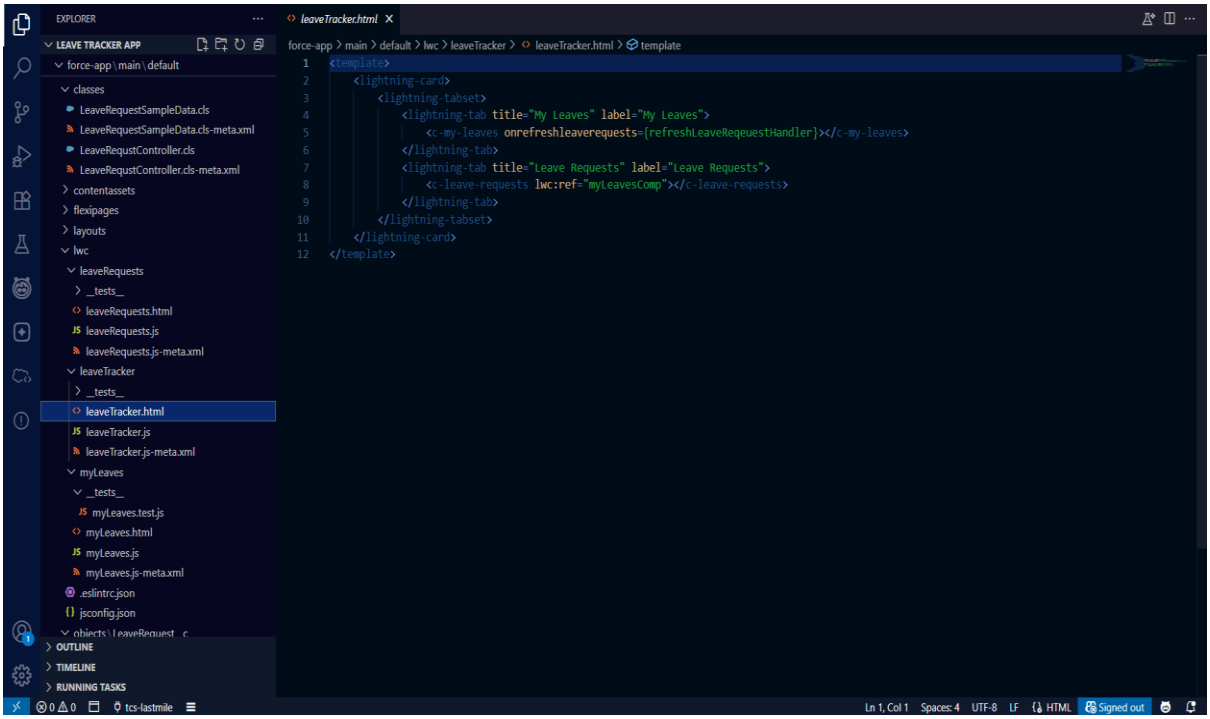
Purpose:

The purpose of this phase was to design an interactive and user-friendly interface for agents, managers, and policyholders using Salesforce Lightning Web Components (LWC). This ensures that claim submissions, claim tracking, and managerial dashboards are intuitive, efficient, and aligned with Salesforce Lightning Design System (SLDS) standards.

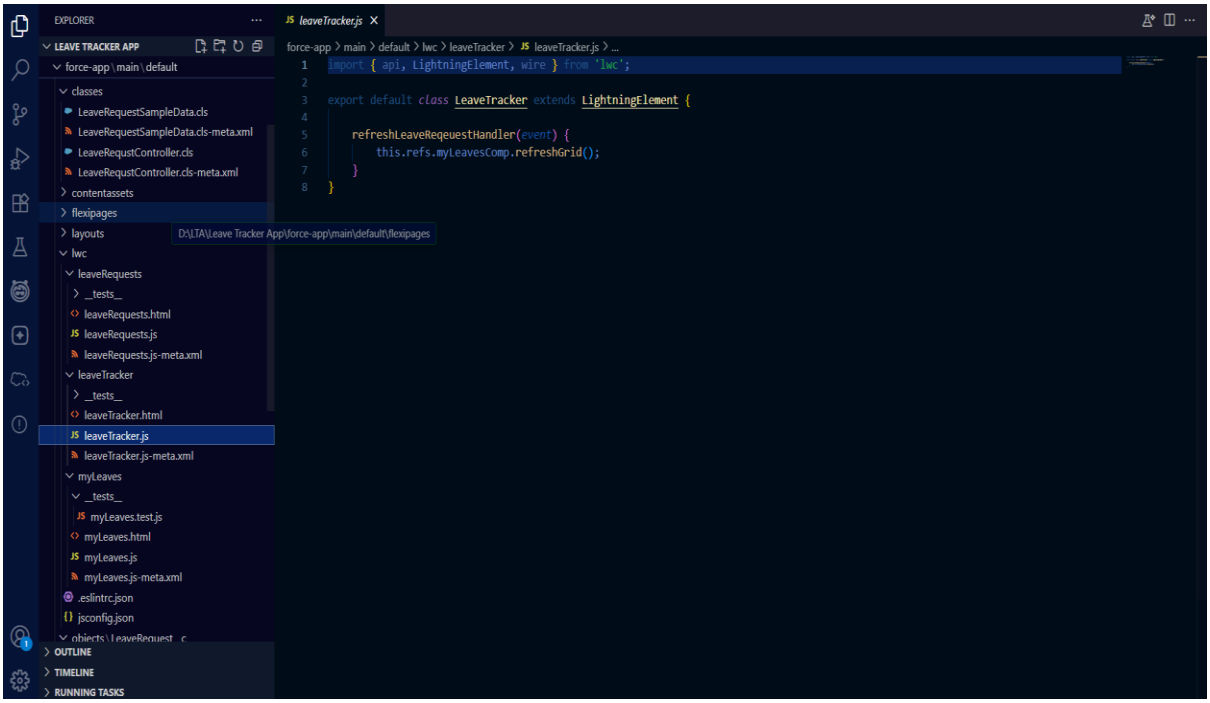
Implementation:

1. Parent component using LWC:
    - Using LWC (lightning web component) the parent component is created the project
    - The parent component is ‘leave Tracker’.
    - SFDX: Create Lightning Web Component from the command palette and give the name “leaveTracker”.

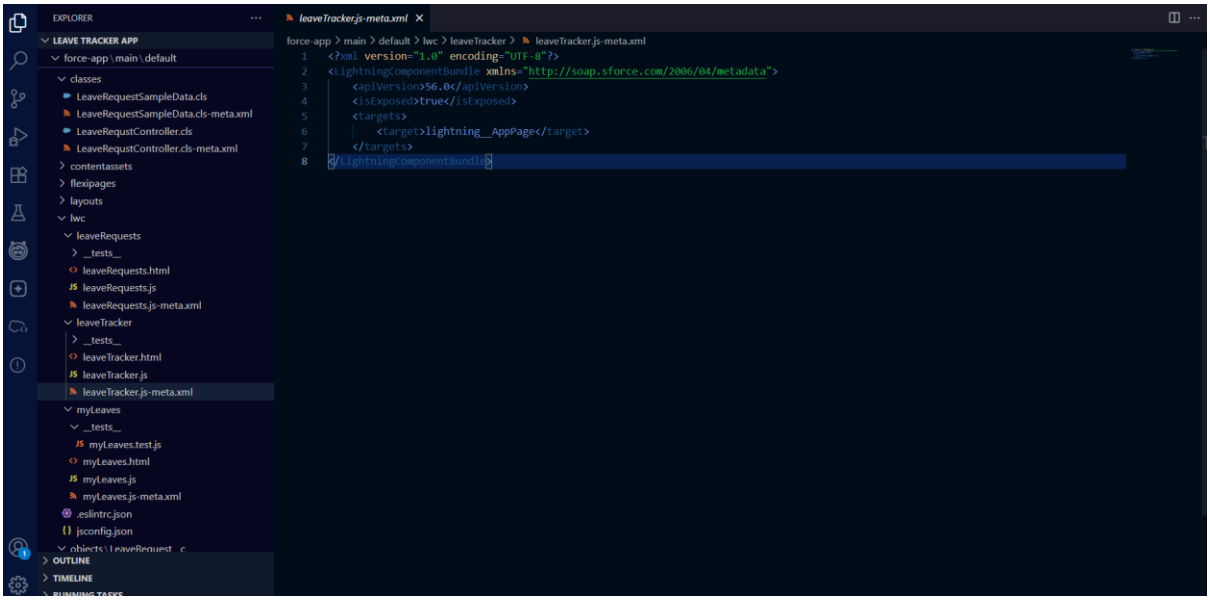




leaveTracker.html

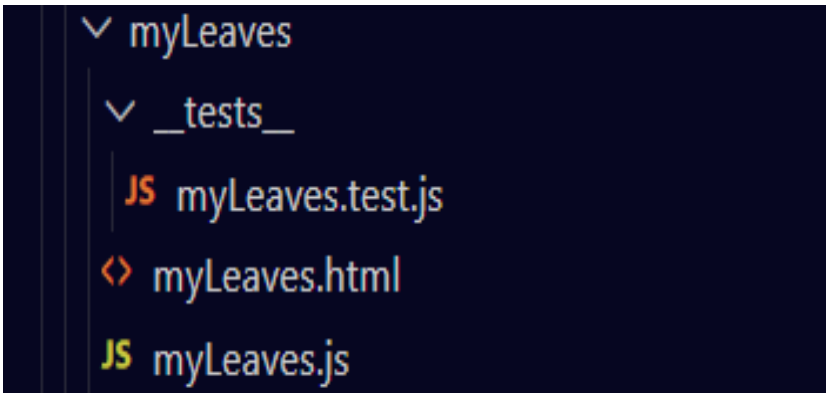


leaveTracker.js

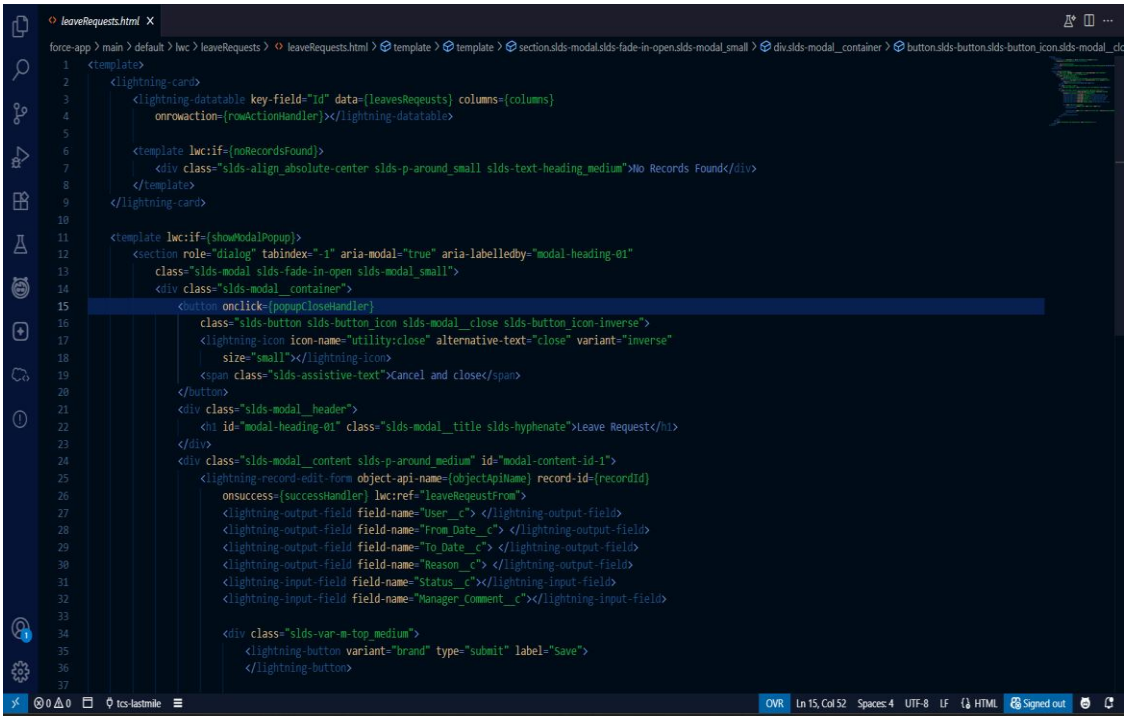


leaveTracker.js-meta.xml

2. Child component using LWC:
   a. leaveRequests
   b. myLeaves

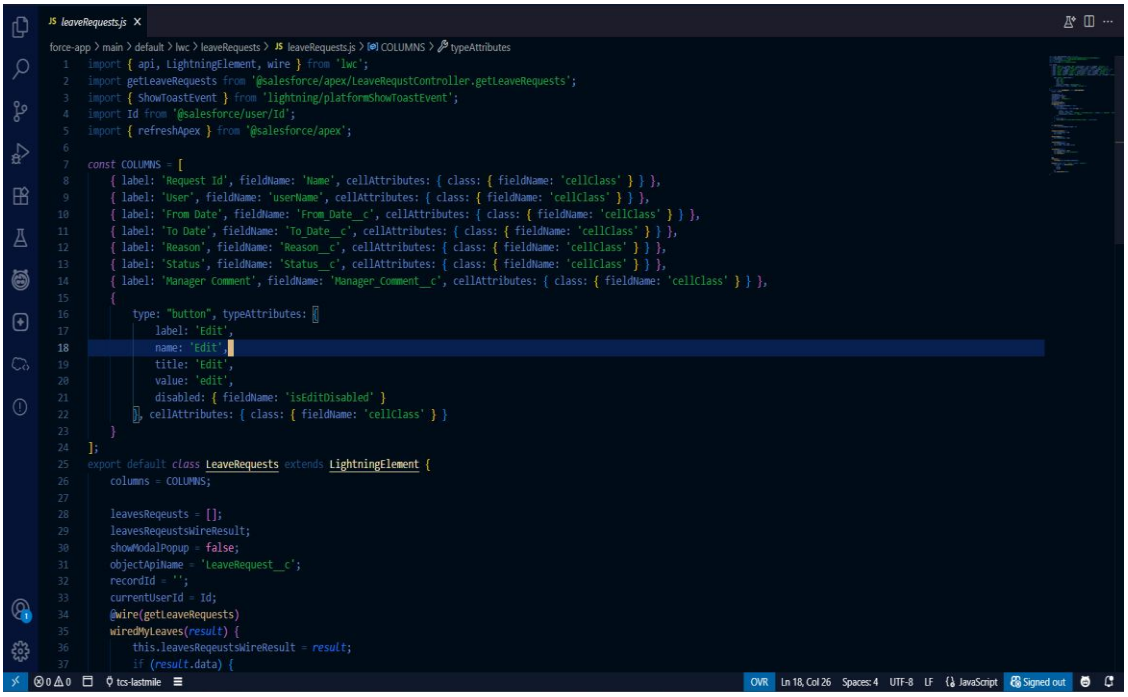


- TheScreenShots for leaveRequests including its code:



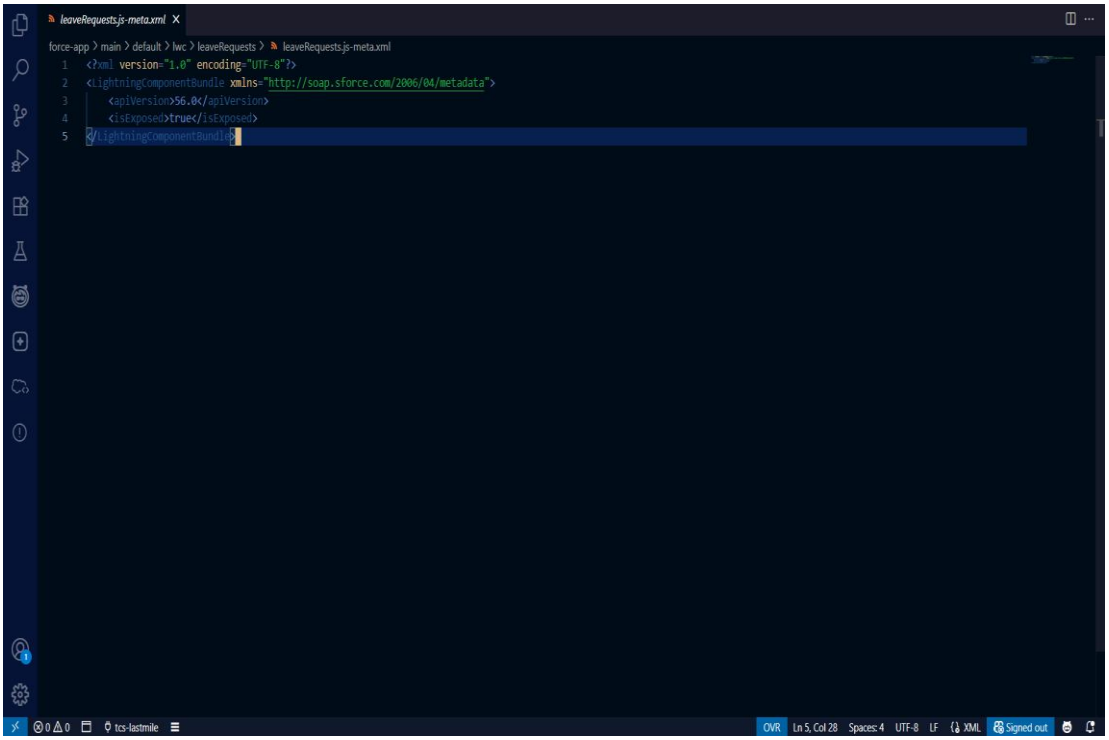
```
<template>
    <lightning-card>
        <lightning-datatable key-field="Id" data={leavesReqeusts} columns={columns}
            onrowaction={rowActionHandler}></lightning-datatable>

        <template lwc:if={noRecordsFound}>
            <div class="slds-align_absolute-center slds-p-around_small slds-text-heading_medium">No Records Found</div>
        </template>
    </lightning-card>

    <template lwc:if={showModalPopup}>
        <section role="dialog" tabindex="-1" aria-modal="true" aria-labelledby="modal-heading-01"
            class="slds-modal slds-fade-in-open slds-modal_small">
            <div class="slds-modal__container">
                <button onclick={popupCloseHandler}
                    class="slds-button slds-button_icon slds-modal__close slds-button_icon-inverse">
                    <lightning-icon icon-name="utility:close" alternative-text="close" variant="inverse"
                        size="small"></lightning-icon>
                    <span class="slds-assistive-text">Cancel and close</span>
                </button>
                <div class="slds-modal__header">
                    <h1 id="modal-heading-01" class="slds-modal__title slds-hyphenate">Leave Request</h1>
                </div>
                <div class="slds-modal__content slds-p-around_medium" id="modal-content-id-1">
                    <lightning-record-edit-form object-api-name={objectApiName} record-id={recordId}
                        onsuccess={successHandler} lwc:ref="leaveReqeustFrom">
                        <lightning-output-field field-name="User__c"> </lightning-output-field>
                        <lightning-output-field field-name="From_Date__c"> </lightning-output-field>
                        <lightning-output-field field-name="To_Date__c"> </lightning-output-field>
                        <lightning-output-field field-name="Reason__c"> </lightning-output-field>
                        <lightning-input-field field-name="Status__c"></lightning-input-field>
                        <lightning-input-field field-name="Manager_Comment__c"></lightning-input-field>

                        <div class="slds-var-m-top_medium">
                            <lightning-button variant="brand" type="submit" label="Save">
                            </lightning-button>
```

leaveRequests.html



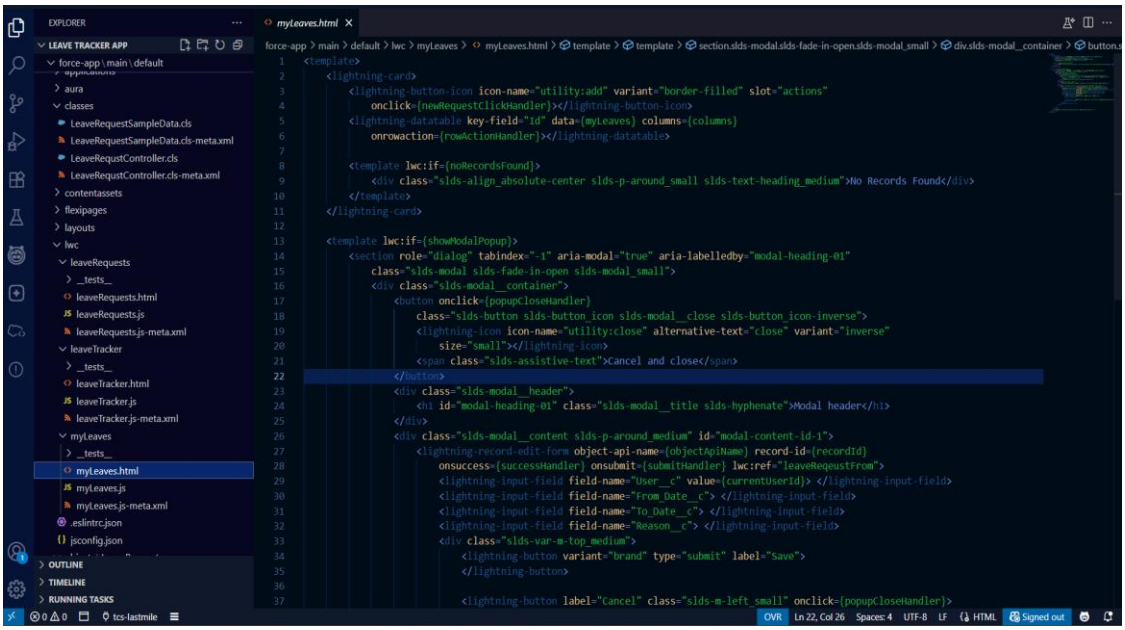
```
import { api, LightningElement, wire } from 'lwc';
import getLeaveRequests from '@salesforce/apex/LeaveRequstController.getLeaveRequests';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import Id from '@salesforce/user/Id';
import { refreshApex } from '@salesforce/apex';

const COLUMNS = [
    { label: 'Request Id', fieldName: 'Name', cellAttributes: { class: { fieldName: 'cellClass' } } },
    { label: 'User', fieldName: 'userName', cellAttributes: { class: { fieldName: 'cellClass' } } },
    { label: 'From Date', fieldName: 'From_Date__c', cellAttributes: { class: { fieldName: 'cellClass' } } },
    { label: 'To Date', fieldName: 'To_Date__c', cellAttributes: { class: { fieldName: 'cellClass' } } },
    { label: 'Reason', fieldName: 'Reason__c', cellAttributes: { class: { fieldName: 'cellClass' } } },
    { label: 'Status', fieldName: 'Status__c', cellAttributes: { class: { fieldName: 'cellClass' } } },
    { label: 'Manager Comment', fieldName: 'Manager_Comment__c', cellAttributes: { class: { fieldName: 'cellClass' } } },
    {
        type: "button", typeAttributes: {
            label: 'Edit',
            name: 'Edit',
            title: 'Edit',
            value: 'edit',
            disabled: { fieldName: 'isEditDisabled' }
        }, cellAttributes: { class: { fieldName: 'cellClass' } }
    }
];
export default class LeaveRequests extends LightningElement {
    columns = COLUMNS;

    leavesReqeusts = [];
    leavesReqeustsWireResult;
    showModalPopup = false;
    objectApiName = 'LeaveRequest__c';
    recordId = '';
    currentUserId = Id;
    @wire(getLeaveRequests)
    wiredMyLeaves(result) {
        this.leavesReqeustsWireResult = result;
        if (result.data) {
```

leaveRequests.js

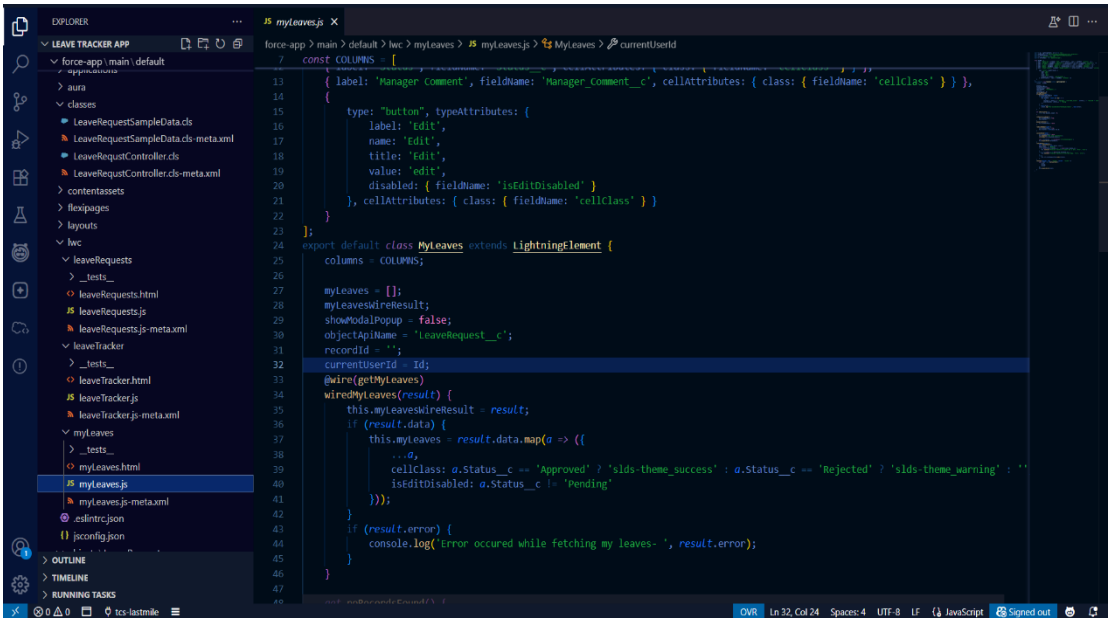


leaveRequests.js-meta.xml

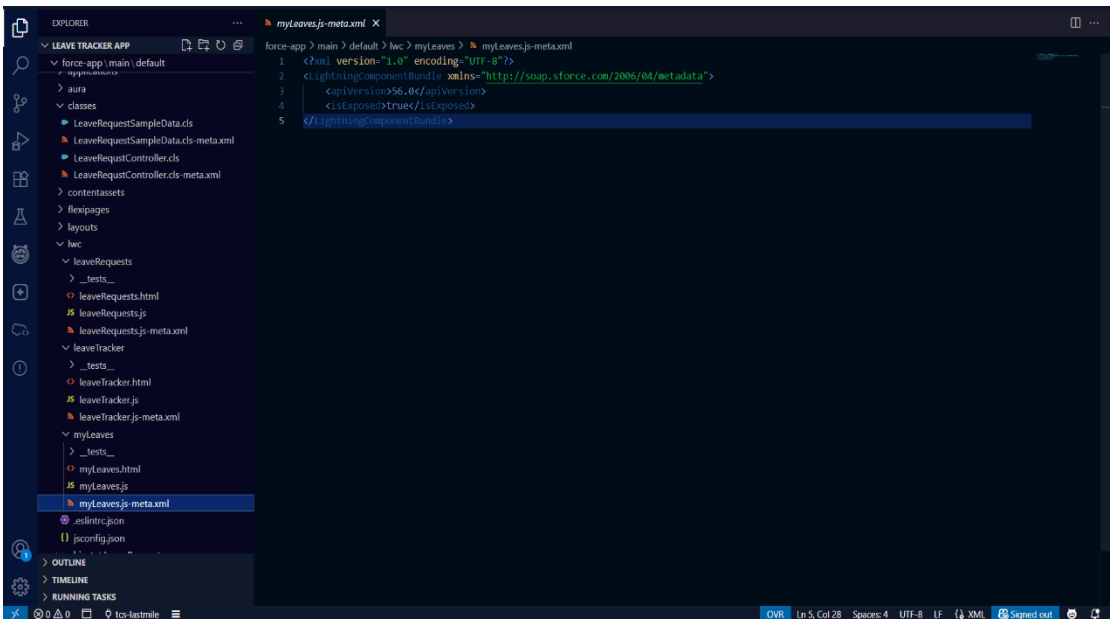
- The ScreenShots for myLeaves:



myLeaves.html



myLeaves.js



myLeaves.js-meta.xml

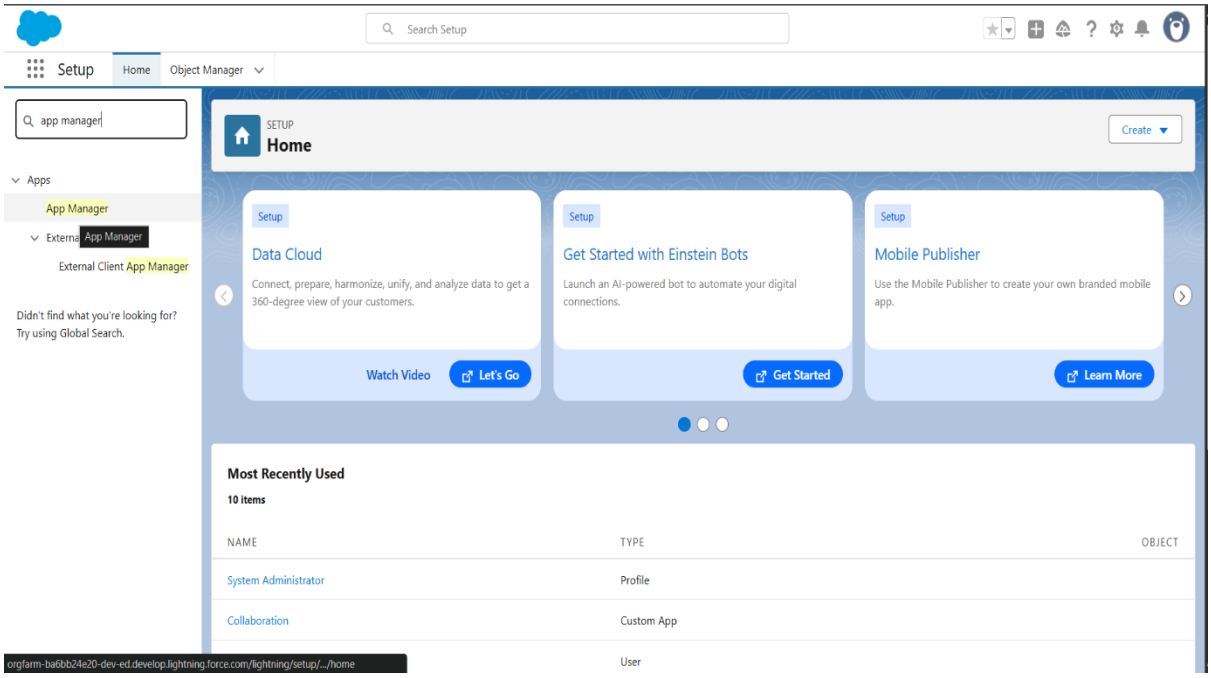
(note: All the ‘Parent and ‘Child’ component’s “-meta.xml” file’s [<isExposed> </isExposed>] should be ‘true’.)

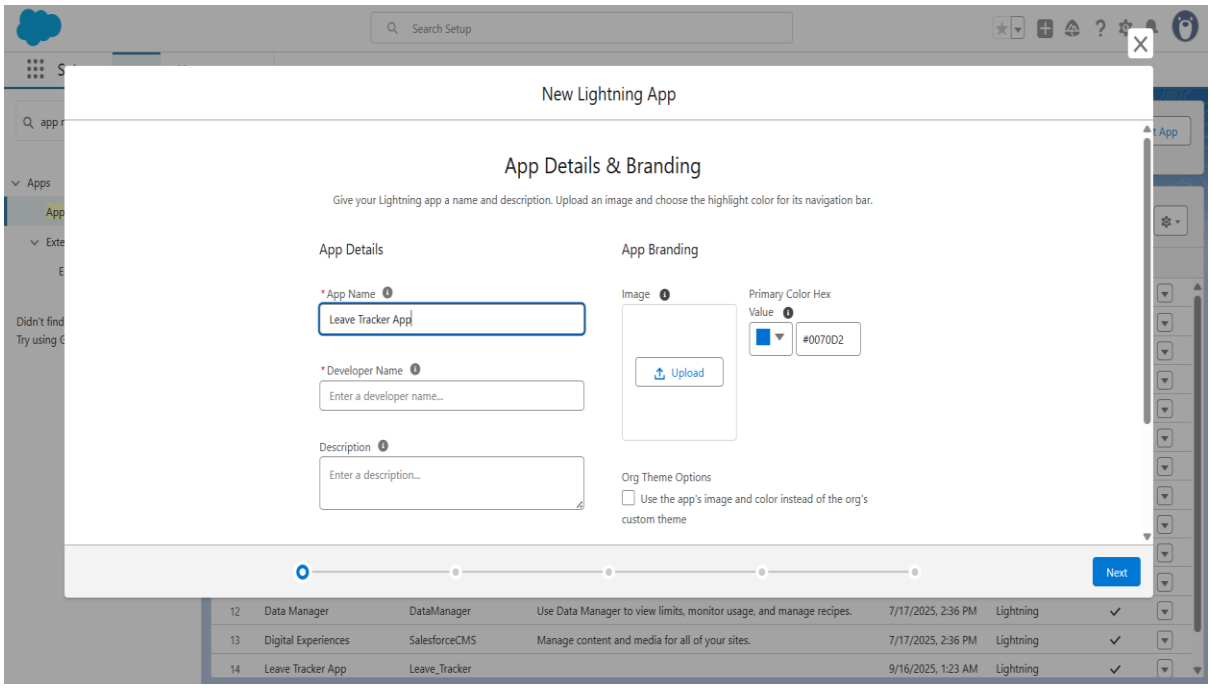
3. Lightning App Creation/Implimentation:

   After completion of the codes and deploying it on the salesforce org we have to create a Lightning App and add the “custom component” developed using LWC and VScode in it.

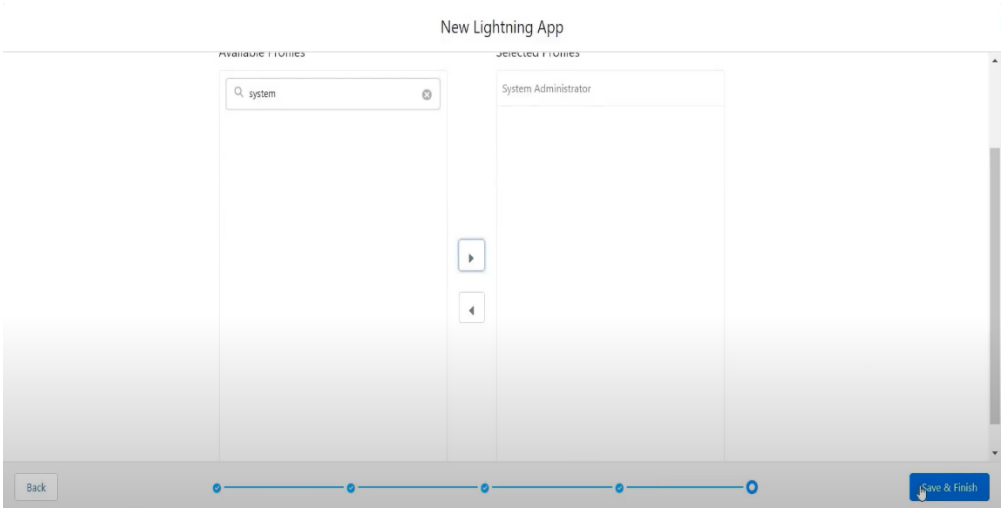
   Step 1:

   Goto setup→App manager→New Lightning App

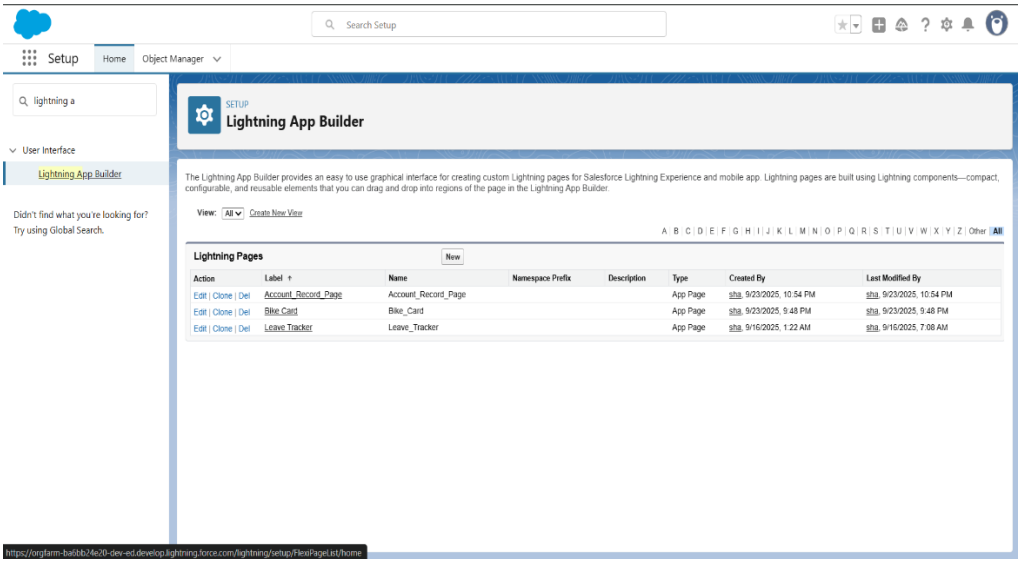


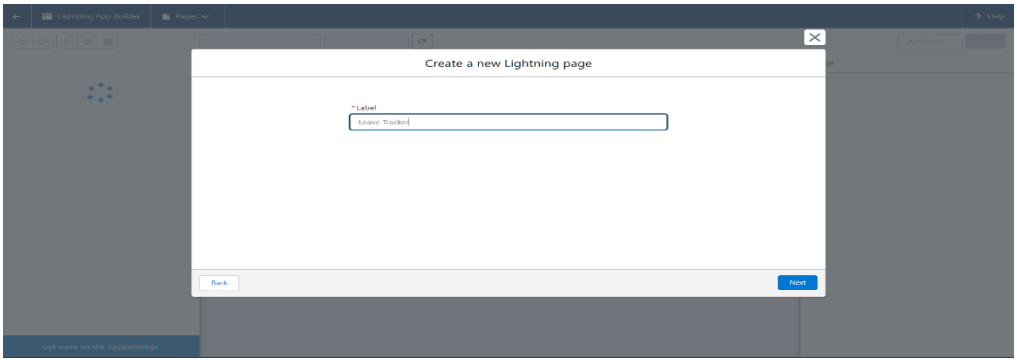


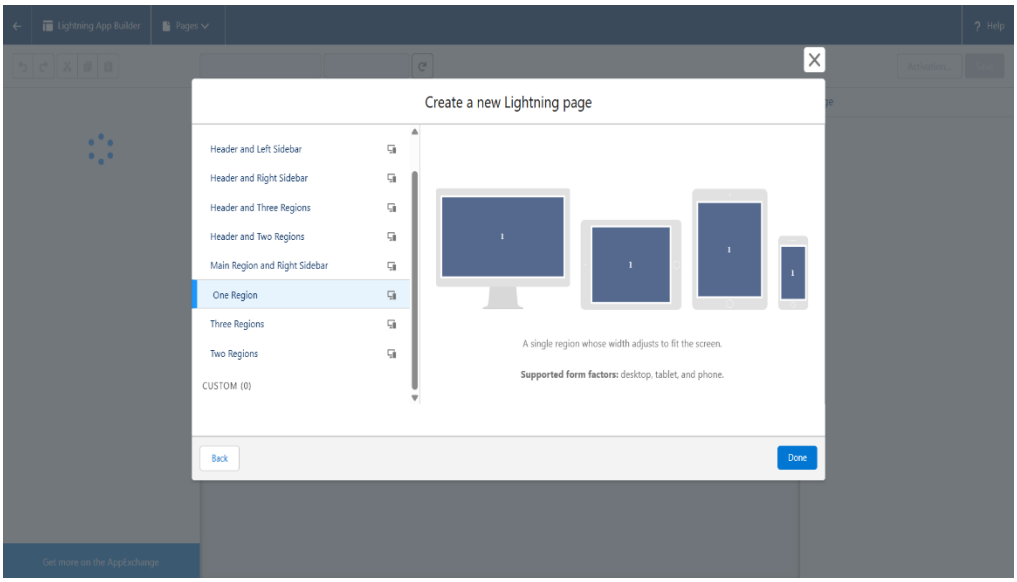
- After inserting the App name “Leave Tracker App” click on Next→Next→Next→Next→ now select “System Administrator” and click Save & Finish.



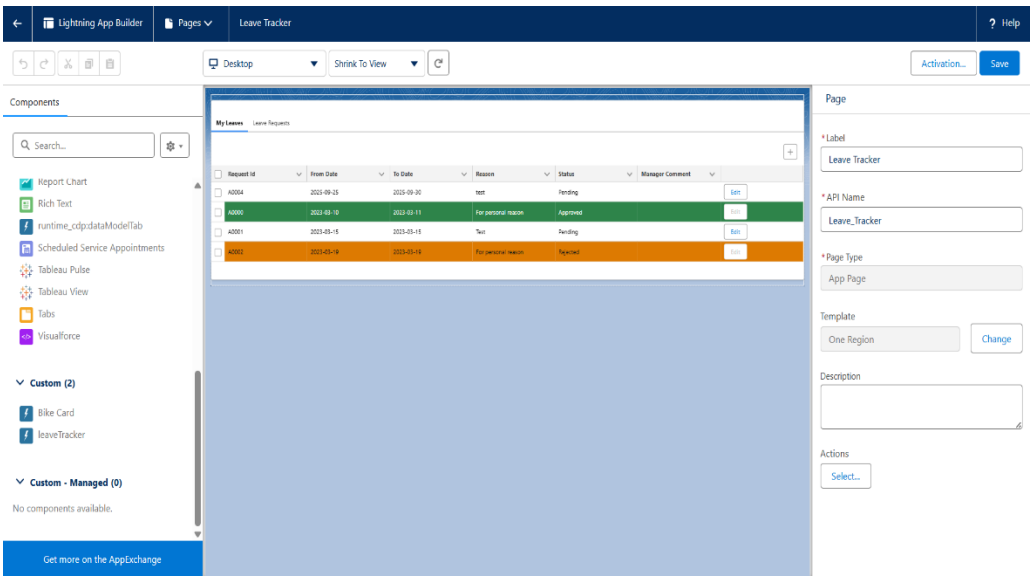
- Now goto home→ Lightning App Builder→ New and label it “leave Tracker”. (select the App page before it then hit Next then select One region).



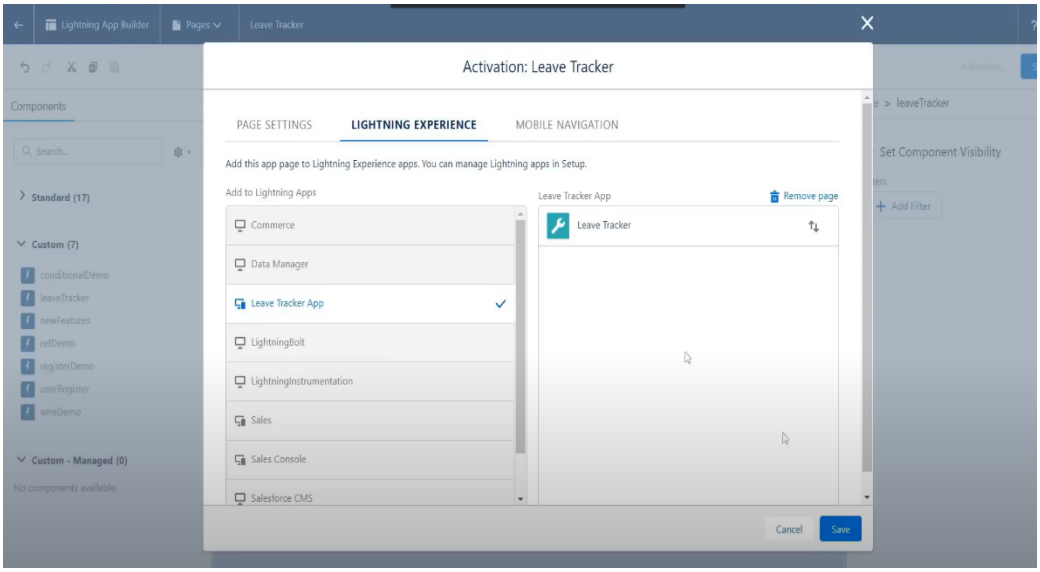




- Now drag and drop the Custom ‘leave Tracker’ and then save and activate.



- Now save it and activate it (also in the Lightning Experience add this to Leave Tracker App we created Earlier).



- Also I implemented the some other things like:
  a. Wire decorator implementation
  b. Edit button click handler
  c. Add button implementation
  d. Grid auto-refresh using refreshApex (imperative Apex call)
  e. Custom event creation and dispatch
  f. Component communication

  These all are implemented using VS code and then deployed using [SFDX: Deploy this source to org].

  (note: I have pushed all the codes in the GitHub Repository, so you will abe to see these implimentations in the code).

Phase 7: Integration & External Access (not applicable for the project/only Conceptually)

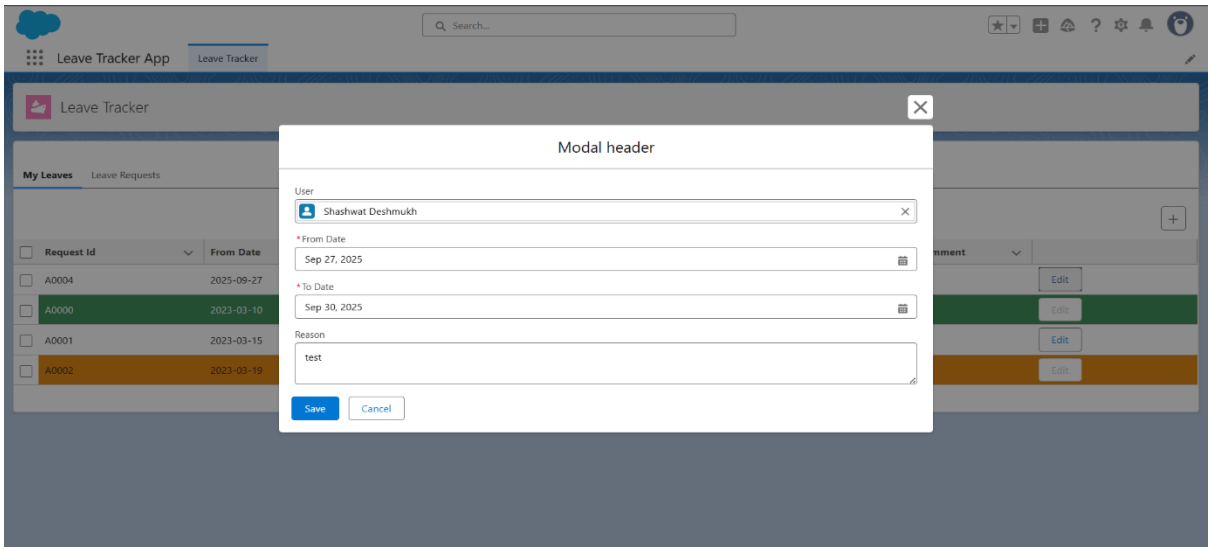
Purpose:

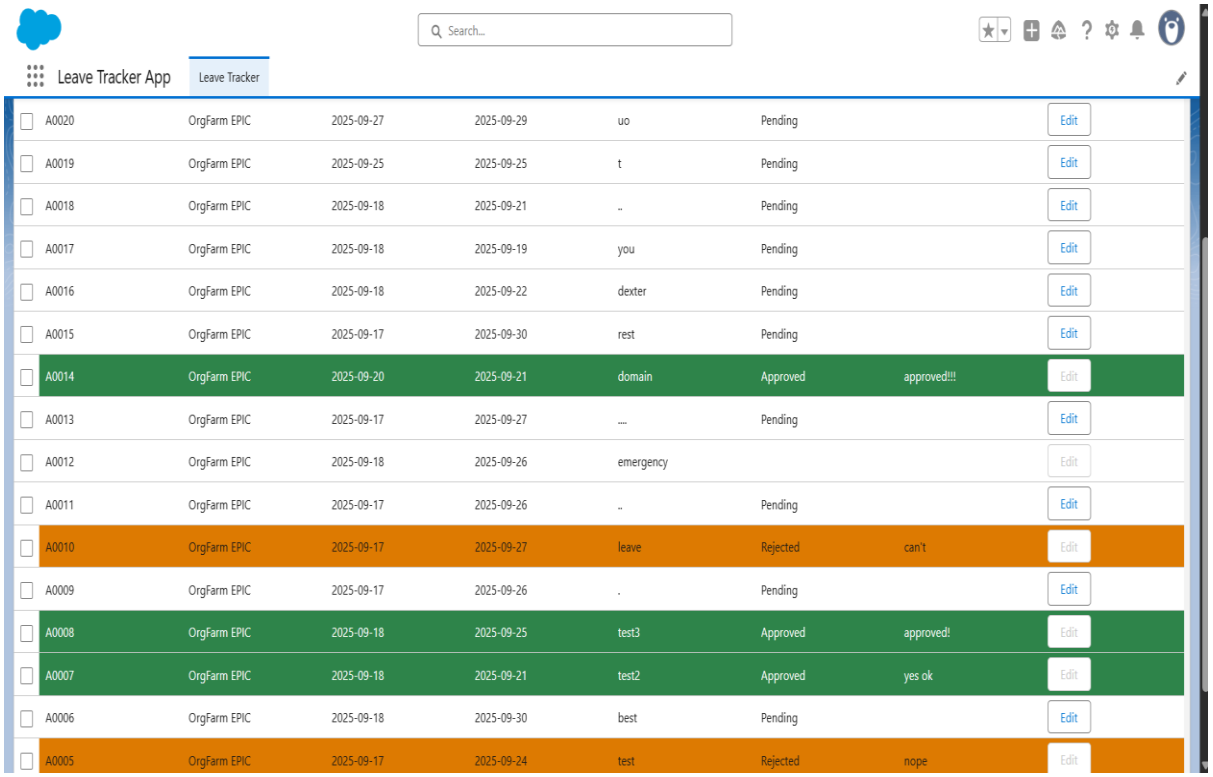
This phase aimed to connect Salesforce with external systems, such as a policyholder portal and third-party payment services, ensuring seamless data exchange and notification delivery.

1. Request and Reply Integration:
   The Leave Tracker App implements Request & Reply pattern for real-time interactions with external systems. When employees submit leave requests, the system can instantly verify leave balances from external payroll systems and return approval status.

2. Batch Data Synchronization:
   For large-scale data operations, the app utilizes Batch Data Synchronization pattern to sync employee data, leave policies, and historical leave records. This ensures data consistency across multiple systems without impacting real-time user experience.

3. Data Virtualization with Salesforce Connect
   External employee data from HRMS systems can be accessed in real-time using Salesforce Connect without storing duplicate records. This zero-copy approach maintains data integrity while providing unified access to employee information.




Search...
Leave Tracker App
Leave Tracker
A0020 OrgFarm EPIC 2025-09-27 2025-09-29 uo Pending Edit
A0019 OrgFarm EPIC 2025-09-25 2025-09-25 t Pending Edit
A0018 OrgFarm EPIC 2025-09-18 2025-09-21 .. Pending Edit
A0017 OrgFarm EPIC 2025-09-18 2025-09-19 you Pending Edit
A0016 OrgFarm EPIC 2025-09-18 2025-09-22 dexter Pending Edit
A0015 OrgFarm EPIC 2025-09-17 2025-09-30 rest Pending Edit
A0014 OrgFarm EPIC 2025-09-20 2025-09-21 domain Approved approved!!! Edit
A0013 OrgFarm EPIC 2025-09-17 2025-09-27 .... Pending Edit
A0012 OrgFarm EPIC 2025-09-18 2025-09-26 emergency Edit
A0011 OrgFarm EPIC 2025-09-17 2025-09-26 .. Pending Edit
A0010 OrgFarm EPIC 2025-09-17 2025-09-27 leave Rejected can't Edit
A0009 OrgFarm EPIC 2025-09-17 2025-09-26 . Pending Edit
A0008 OrgFarm EPIC 2025-09-18 2025-09-25 test3 Approved approved! Edit
A0007 OrgFarm EPIC 2025-09-18 2025-09-21 test2 Approved yes ok Edit
A0006 OrgFarm EPIC 2025-09-18 2025-09-30 best Pending Edit
A0005 OrgFarm EPIC 2025-09-17 2025-09-24 test Rejected nope Edit

Phase 8: Data Management & Deployment

Purpose:

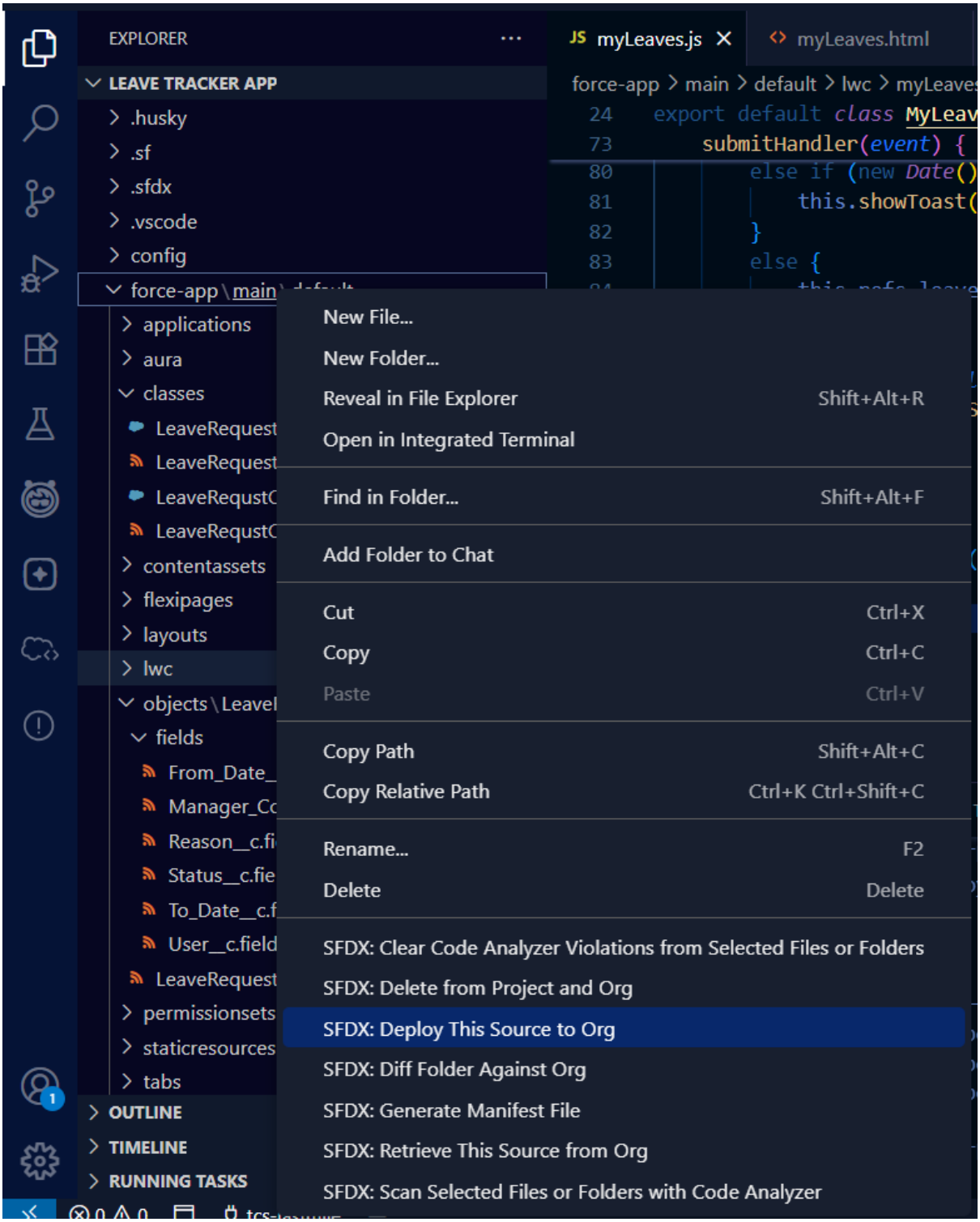
The purpose of this phase was to ensure accurate data migration, prevent duplication, and deploy the solution across different Salesforce environments securely.

1. VS Code & SFDX:
   VS Code (Visual Studio Code) is a free, open-source code editor developed by Microsoft that serves as the primary development environment for Salesforce developers. It's lightweight, extensible, and available across Windows, Linux, and macOS platforms.

   SFDX (Salesforce DX) is Salesforce's modern development platform that includes a commandline interface (CLI) and development tools designed for source-driven development. It represents the official future of Salesforce development, offering enhanced capabilities for working with metadata, scratch orgs, and version control systems.

2. How they work together:
   VS Code and SFDX integrate seamlessly through the Salesforce Extension Pack, which brings SFDX functionality directly into the VS Code interface. This combination enables developers to create projects, authorize orgs, retrieve source code, deploy changes, and manage Salesforce metadata all within a unified development environment.

   After Completing the code and debugging all the errors it is then deployed to the salesforce org using [SFDX Deploy this source to org]


EXPLORER
LEAVE TRACKER APP
.husky
.sf
.sfdx
.vscode
config
force-app \ main
applications
aura
classes
contentassets
flexipages
layouts
lwc
fields
permissionsets
staticresources
tabs
OUTLINE
TIMELINE
RUNNING TASKS
myLeaves.js
myLeaves.html
force-app > main > default > lwc > myLeaves
export default class MyLeav
submitHandler(event) {
else if (new Date()
this.showToast(
}
else {
New File...
New Folder...
Reveal in File Explorer Shift+Alt+R
Open in Integrated Terminal
Find in Folder... Shift+Alt+F
Add Folder to Chat
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Copy Path Shift+Alt+C
Copy Relative Path Ctrl+K Ctrl+Shift+C
Rename... F2
Delete Delete
SFDX: Clear Code Analyzer Violations from Selected Files or Folders
SFDX: Delete from Project and Org
SFDX: Deploy This Source to Org
SFDX: Diff Folder Against Org
SFDX: Generate Manifest File
SFDX: Retrieve This Source from Org
SFDX: Scan Selected Files or Folders with Code Analyzer

Phase 9: Reporting, Dashboards & Security Review

Purpose:

The goal of this phase was to provide managers and agents with actionable insights through Salesforce reports and dashboards, enabling better decision-making and performance tracking.

1. Reports & Dashboards:
   Leave Summary Reports: Create tabular reports showing employee leave requests by status (Pending, Approved, Rejected) and summary reports grouped by department and manager. Include matrix reports for comparing leave usage across different time periods.

   Leave Analytics Dashboard: Build dashboards displaying key metrics like total leave requests, pending approvals, and manager workload distribution. Add dynamic dashboard components that filter data based on user roles.

   Custom Report Types: Create report types combining Leave Request records with User data to enable detailed leave pattern analysis.

2. Security Implementations:
   Sharing Settings: Configure Organization-Wide Defaults for Leave Request object to Private, with sharing rules allowing managers to see their team's requests and HR to access all records.

   Field Level Security: Restrict access to sensitive fields like Manager Comments to appropriate user profiles only. Ensure employees can only edit their own pending requests.

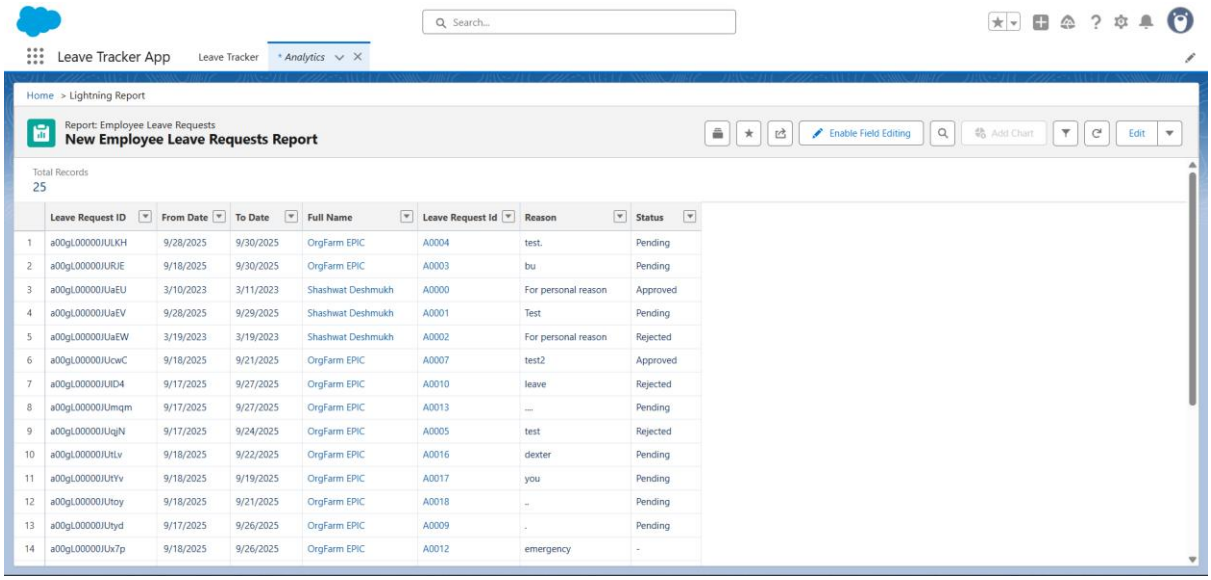
   Audit Trail: Enable field history tracking on critical fields like Status and Approval Date for compliance monitoring.

Reports

● Employee Leave Report: Total leaves taken by each employee in a period.

● Leave Type Report: Breakdown of Sick Leave, Casual Leave, etc.

● Pending Requests Report: Leaves still awaiting manager approval.



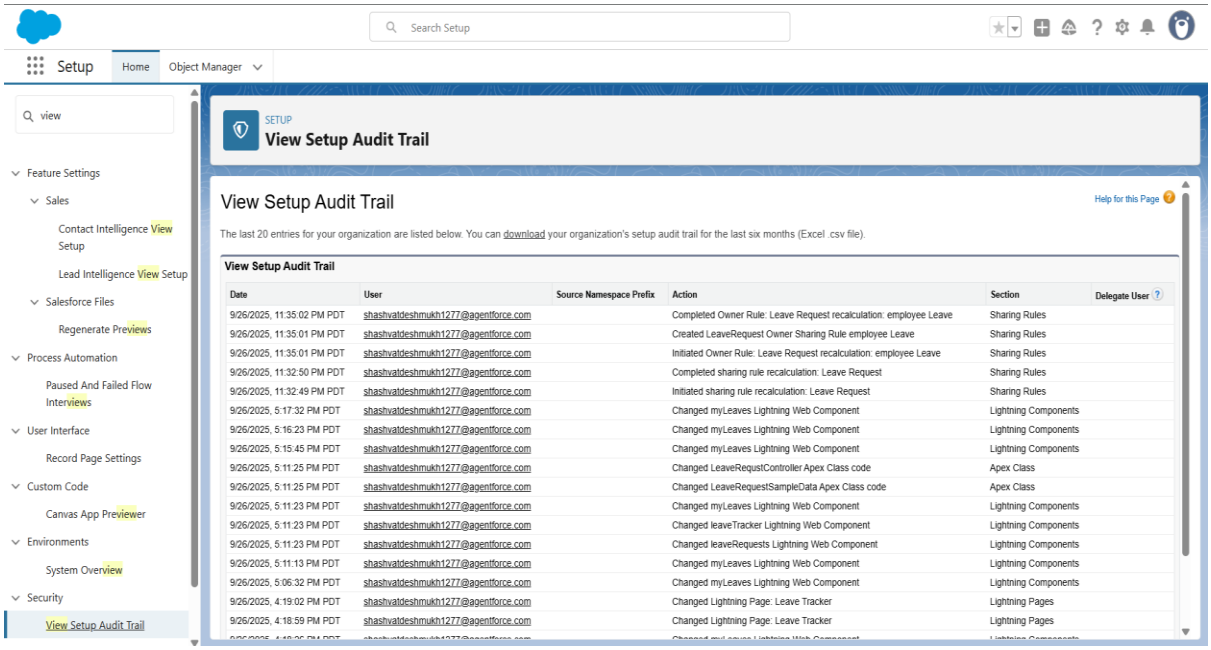
	Leave Request ID	From Date	To Date	Full Name	Leave Request Id	Reason	Status
1	a00gL00000JULKH	9/28/2025	9/30/2025	OrgFarm EPIC	A0004	test.	Pending
2	a00gL00000JURJE	9/18/2025	9/30/2025	OrgFarm EPIC	A0003	bu	Pending
3	a00gL00000JUaEU	3/10/2023	3/11/2023	Shashwat Deshmukh	A0000	For personal reason	Approved
4	a00gL00000JUaEV	9/28/2025	9/29/2025	Shashwat Deshmukh	A0001	Test	Pending
5	a00gL00000JUaEW	3/19/2023	3/19/2023	Shashwat Deshmukh	A0002	For personal reason	Rejected
6	a00gL00000JUcwC	9/18/2025	9/21/2025	OrgFarm EPIC	A0007	test2	Approved
7	a00gL00000JUID4	9/17/2025	9/27/2025	OrgFarm EPIC	A0010	leave	Rejected
8	a00gL00000JUmqm	9/17/2025	9/27/2025	OrgFarm EPIC	A0013	....	Pending
9	a00gL00000JUqjN	9/17/2025	9/24/2025	OrgFarm EPIC	A0005	test	Rejected
10	a00gL00000JUtLv	9/18/2025	9/22/2025	OrgFarm EPIC	A0016	dexter	Pending
11	a00gL00000JUtYv	9/18/2025	9/19/2025	OrgFarm EPIC	A0017	you	Pending
12	a00gL00000JUtoy	9/18/2025	9/21/2025	OrgFarm EPIC	A0018	-	Pending
13	a00gL00000JUtyd	9/17/2025	9/26/2025	OrgFarm EPIC	A0009	.	Pending
14	a00gL00000JUx7p	9/18/2025	9/26/2025	OrgFarm EPIC	A0012	emergency	-

Audit Trail

● Track all changes to LeaveRequest__c records.

● Example: Who approved/rejected a leave, and when.

● Provides compliance visibility for HR audits.



Date	User	Source Namespace Prefix	Action	Section	Delegate User
9/26/2025, 11:35:02 PM PDT	shashvatdeshmukh1277@agentforce.com		Completed Owner Rule: Leave Request recalculation: employee Leave	Sharing Rules	
9/26/2025, 11:35:01 PM PDT	shashvatdeshmukh1277@agentforce.com		Created LeaveRequest Owner Sharing Rule employee Leave	Sharing Rules	
9/26/2025, 11:35:01 PM PDT	shashvatdeshmukh1277@agentforce.com		Initiated Owner Rule: Leave Request recalculation: employee Leave	Sharing Rules	
9/26/2025, 11:32:50 PM PDT	shashvatdeshmukh1277@agentforce.com		Completed sharing rule recalculation: Leave Request	Sharing Rules	
9/26/2025, 11:32:49 PM PDT	shashvatdeshmukh1277@agentforce.com		Initiated sharing rule recalculation: Leave Request	Sharing Rules	
9/26/2025, 5:17:32 PM PDT	shashvatdeshmukh1277@agentforce.com		Changed myLeaves Lightning Web Component	Lightning Components	
9/26/2025, 5:16:23 PM PDT	shashvatdeshmukh1277@agentforce.com		Changed myLeaves Lightning Web Component	Lightning Components	
9/26/2025, 5:15:45 PM PDT	shashvatdeshmukh1277@agentforce.com		Changed myLeaves Lightning Web Component	Lightning Components	
9/26/2025, 5:11:25 PM PDT	shashvatdeshmukh1277@agentforce.com		Changed LeaveRequstController Apex Class code	Apex Class	
9/26/2025, 5:11:25 PM PDT	shashvatdeshmukh1277@agentforce.com		Changed LeaveRequestSampleData Apex Class code	Apex Class	
9/26/2025, 5:11:23 PM PDT	shashvatdeshmukh1277@agentforce.com		Changed myLeaves Lightning Web Component	Lightning Components	
9/26/2025, 5:11:23 PM PDT	shashvatdeshmukh1277@agentforce.com		Changed leaveTracker Lightning Web Component	Lightning Components	
9/26/2025, 5:11:23 PM PDT	shashvatdeshmukh1277@agentforce.com		Changed leaveRequests Lightning Web Component	Lightning Components	
9/26/2025, 5:11:13 PM PDT	shashvatdeshmukh1277@agentforce.com		Changed myLeaves Lightning Web Component	Lightning Components	
9/26/2025, 5:06:32 PM PDT	shashvatdeshmukh1277@agentforce.com		Changed myLeaves Lightning Web Component	Lightning Components	
9/26/2025, 4:19:02 PM PDT	shashvatdeshmukh1277@agentforce.com		Changed Lightning Page: Leave Tracker	Lightning Pages	
9/26/2025, 4:18:59 PM PDT	shashvatdeshmukh1277@agentforce.com		Changed Lightning Page: Leave Tracker	Lightning Pages	

Phase 10: Final Presentation & Demo Day

Purpose:

The final phase ensured smooth knowledge transfer to stakeholders through demos, documentation, and training. It prepared the project for production use.

1. Project Presentation:
   Business Impact Overview: Present how the Leave Tracker App transformed manual leave processes into an automated Lightning Web Component solution. Highlight key improvements like real-time updates and streamlined approval workflows.

   Technical Showcase: Demonstrate the LWC architecture including wire decorators, component communication between "My Leaves" and "Leave Requests" tabs, and Lightning Data Services integration.

2. Live Demo:
   End-to-End Walkthrough: Show complete leave request process from employee submission through manager approval, highlighting modal popups, form validations, and automatic grid refresh without page reload.

   Component Communication Demo: Demonstrate how creating requests in one tab automatically updates the other tab through custom events and parent-child component interaction.

3. Documentation & Feedback:
   Handoff Materials: Provide technical documentation covering component structure, deployment using VS Code/SFDX, and administrator configuration guides.

   User Training: Create step-by-step guides for both employees and managers covering the new Lightning Web Component interface.

   Code Repository: Upload clean, documented code to GitHub demonstrating LWC best practices, wire adapters, and component communication patterns.

Demo Video: Create screen recording showcasing the Leave Tracker App functionality with technical explanations suitable for portfolio presentation.

This simplified approach focuses on the core deliverables that directly relate to the Leave Management System demonstrated in the video, emphasizing practical implementation over extensive theoretical details.