Phase 6: User Interface Development

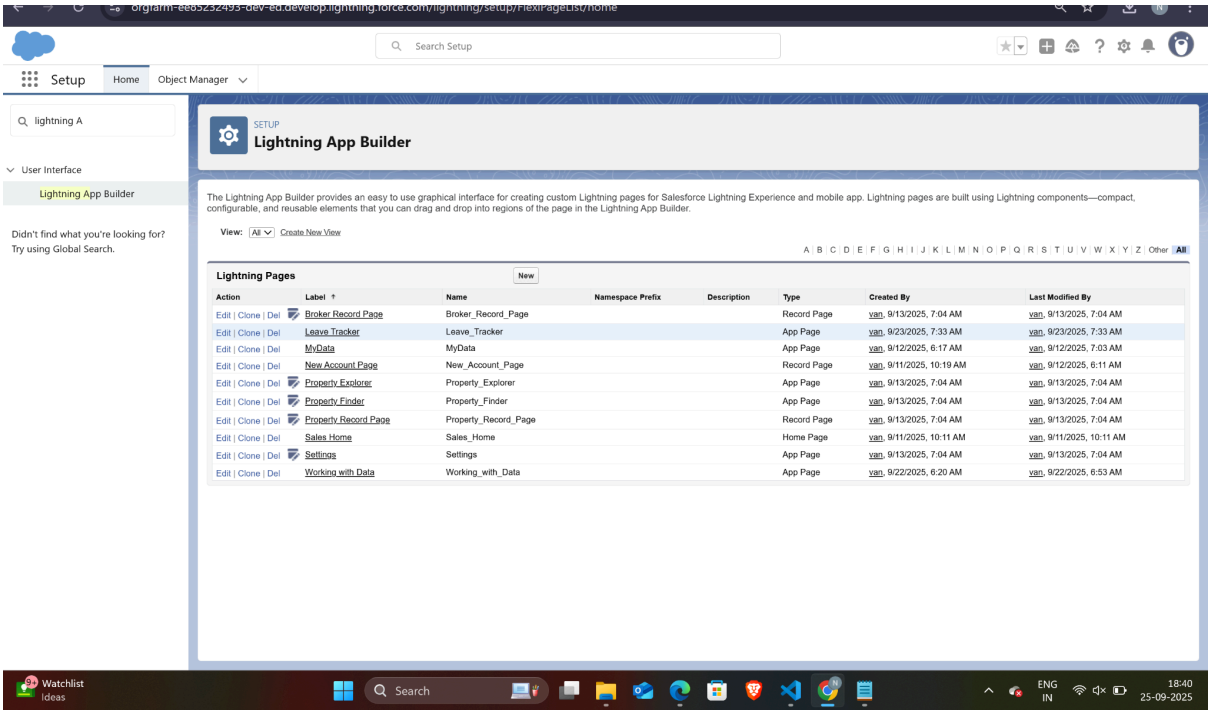
Goal: Deliver an intuitive and interactive UI for the Leave Tracking application.

1. Lightning App Builder

Created a dedicated **Leave Tracking App** in Salesforce.

The app provides navigation tabs for:

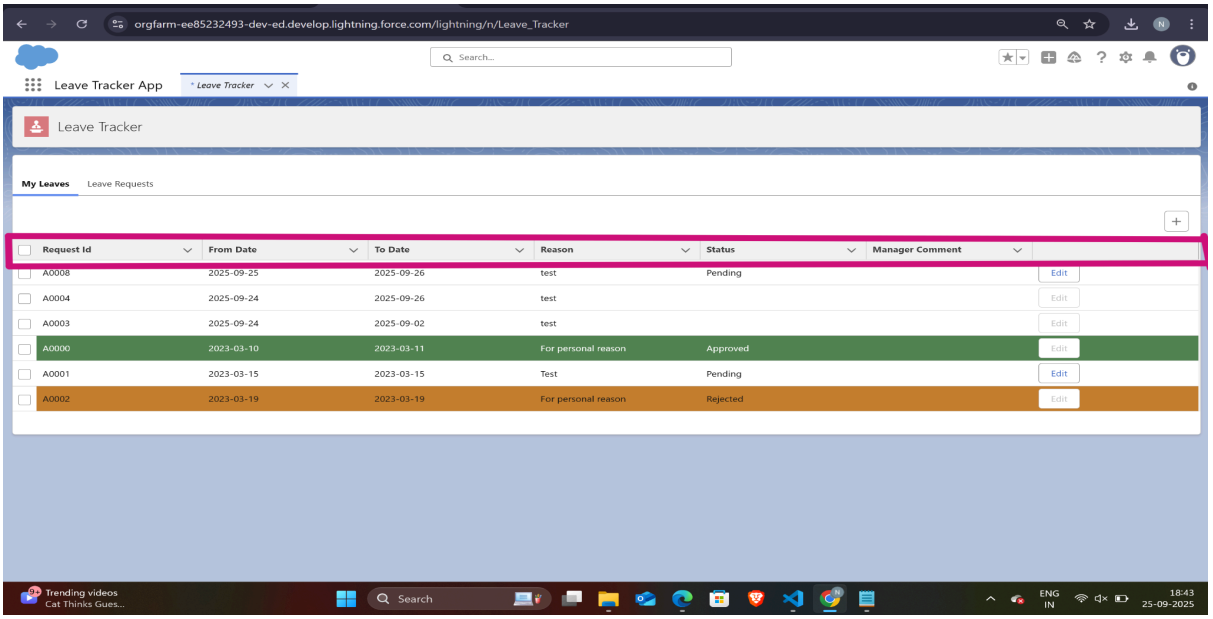
- **Home** (overview dashboard & notifications)
- **My Leaves** (employee's personal leave history)
- **Team Requests** (manager view for approvals)
- **Analytics** (reports and dashboards)



2. Record Pages

Customized the **LeaveRequest__c Record Page** to display:

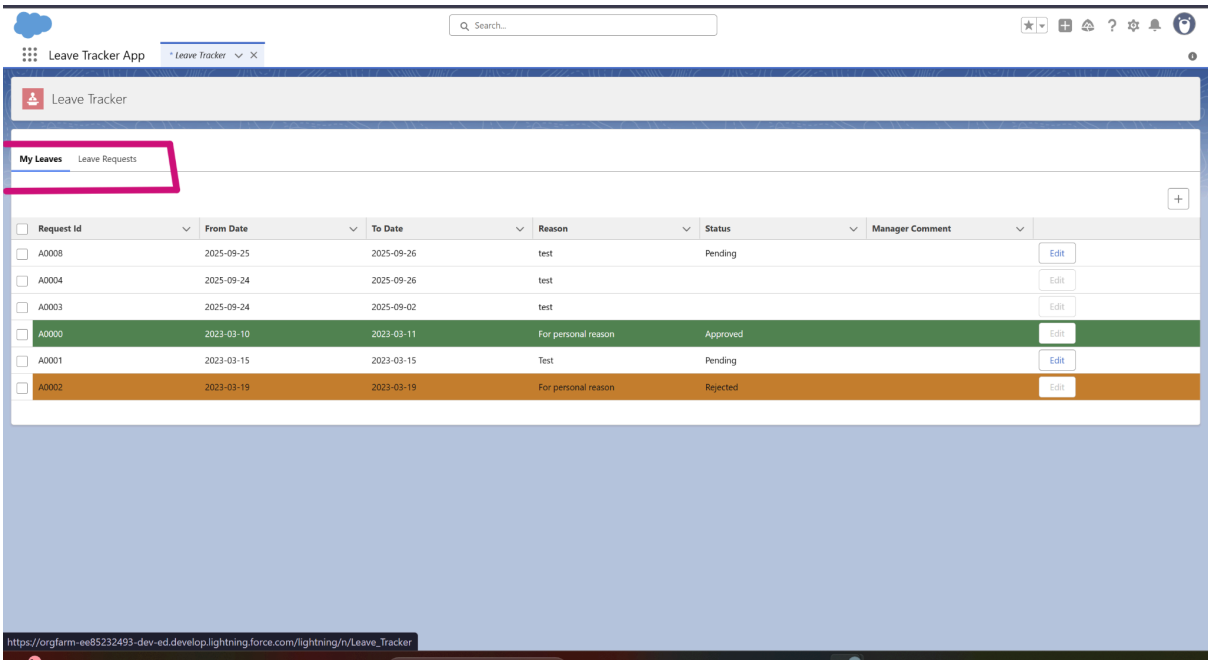
- Key employee details (Name, Department)
- Leave type, dates, and submitted reason
- Manager's comments and decision section
- Quick Actions for **Approve** and **Reject**
- Compact Layout for faster glance at status



3. Tabs

Configured custom tabs for:

- **Leave Request** (object tab for direct access)
- **My Leaves** (Lightning page with user Leaves)

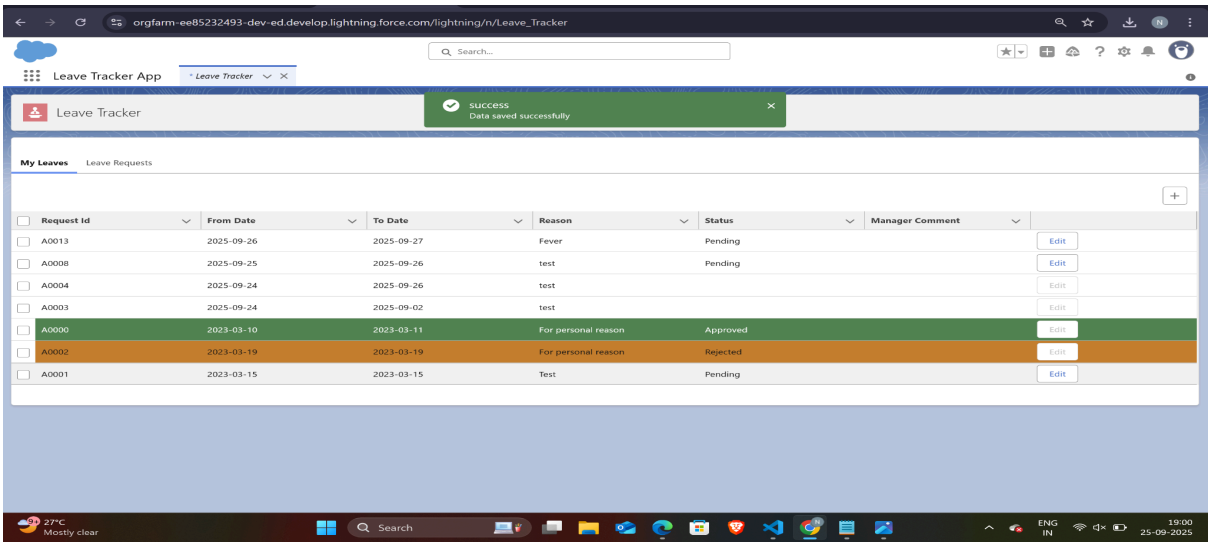


4. Home Page Layouts

Designed a tailored home page with:

- **Leave Balance Chart** (available vs. used leave days)

- **Quick Action: Request Leave**
- **Recent Leave Submissions** list view



5. Utility Bar

Added shortcuts for:

- **Apply for Leave** (record form popup)
- **Ask HR** (case creation for leave-related queries)

6. Lightning Web Components (LWC)

Developed multiple LWCs to improve user experience:

applyLeave

Form for employees to submit leave requests.
Fields include: Leave Type, From Date, To Date, Reason.
Validates input before sending to Apex.

```
// applyLeave.js
import { LightningElement, track } from 'lwc';
import createLeave from '@salesforce/apex/LeaveRequestController.createLeave';

export default class ApplyLeave extends LightningElement {
    @track leaveType;
    @track fromDate;
    @track toDate;
    @track reason;

    handleSubmit() {
```

```
        if(!this.leaveType || !this.fromDate || !this.toDate) {
            alert('Please complete all required fields');
            return;
        }
        createLeave({
            leaveType: this.leaveType,
            fromDate: this.fromDate,
            toDate: this.toDate,
            reason: this.reason
        })
        .then(() => {
            alert('Leave submitted successfully');
        })
        .catch(error => {
            console.error(error);
        });
    }
}
```

myLeaves

- Shows logged-in user's past and upcoming leave requests.
- Data fetched using @wire(getMyLeaves).
- Status highlighted with colors:
    - ✅ Approved = Green
    - ⏳ Pending = Orange
    - ❌ Rejected = Red

teamRequests (Manager View)

- Displays all pending leave requests from direct reports.
- Includes **Approve** / **Reject** buttons.
- On action, calls Apex to update record and sends email notifications.

```
// teamRequests.js
import { LightningElement, wire } from 'lwc';
import getTeamRequests from
'@salesforce/apex/LeaveRequestController.getTeamRequests';
import updateRequestStatus from
'@salesforce/apex/LeaveRequestController.updateRequestStatus';

export default class TeamRequests extends LightningElement {
    @wire(getTeamRequests) requests;
```

```
    handleAction(event) {
        const leaveId = event.target.dataset.id;
        const status = event.target.dataset.status;

        updateRequestStatus({ leaveId, status })
            .then(() => {
                alert(`Leave ${status}`);
            })
            .catch(error => {
                console.error(error);
            });
    }
}
```

7. Apex with LWC

- **Imperative Apex Calls** → used in applyLeave and teamRequests for inserts/updates.

  ```
  import {refreshApex} from '@salesforce/apex';
  ```

- **Wire Adapters** → used in myLeaves to auto-refresh data.

  ```
  @wire(getMyLeaves)
   wiredMyLeaves(result){
   this.myLeavesWireResult=result;
   if(result.data){
   this.myLeaves=result.data.map(a=>({
    ...a,
    cellClass:a.Status__c == 'Approved' ? 'slds-theme_success': a.Status__c ==
'Rejected' ? 'slds-theme_warning' : '',
   isEditDisabled:a.Status__c!= 'Pending'
    }));
  }
  if(result.error){
      console.log('Error occured while fetching my leaves- ',result.error);
  }
  }
  ```

8. Navigation Service

- After submitting a request → navigate to **Leave Record page**.

- After manager approval/rejection → redirect to **Team Requests dashboard**.
- Used Salesforce **NavigationMixin** for seamless transitions.

Phase 6 Outcome

- Built a **modern Lightning interface** with reusable LWCs.
- Employees can submit, track, and cancel leave requests effortlessly.
- Managers can quickly review and act on pending requests.
- The app aligns with Salesforce Lightning UX standards and ensures productivity.