Phase 6: User Interface Development

🎯 Objective

The goal of this phase is to design and implement the **user interface (UI)** for the **Customer Complaint Management System** using Salesforce **Lightning Experience**.
This ensures that users can **easily view, create, and manage complaints, customers, and reports** through an intuitive and interactive layout.

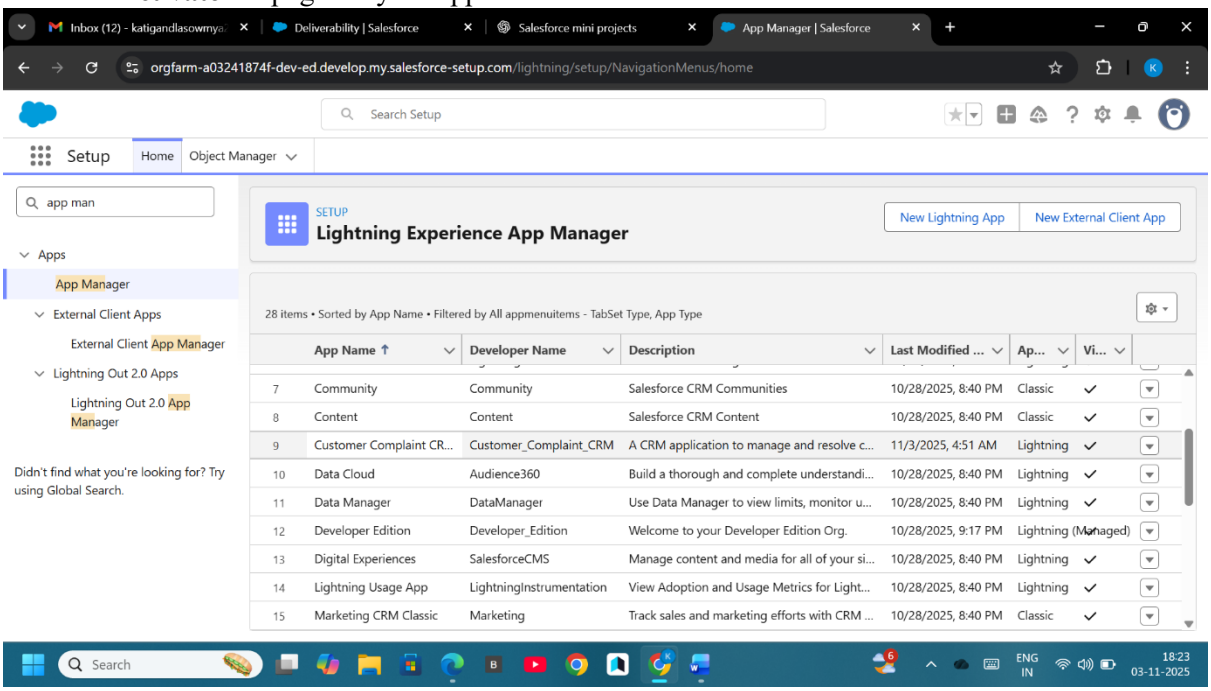
⚙️ 1. Lightning App Builder

Description:

Lightning App Builder allows developers to create and customize app pages visually without code. It is used to design record pages, home pages, and app pages for better user experience.

Steps:

1. Go to **Setup → Lightning App Builder → New**.
2. Choose **App Page / Record Page / Home Page**.
3. Add components like:
    - **Record Detail**
    - **Highlights Panel**
    - **Related Lists**
    - **Charts / Reports**
4. Save and **Activate** the page for your app.



Example:

- Created a **Complaint Record Page** showing key fields like:

- Complaint Number
- Status
- Priority
- Assigned Agent
- Related Customer Details

📄 2. Record Pages

Description:

Record Pages display all details of a specific record in a structured format.
You can customize layouts to highlight the most important information.

Example:

When a user opens a complaint:

- Header shows **Complaint Number**, **Priority**, **Status**
- Tabs display **Details**, **Related Complaints**, **Notes**, **History**

🗂 3. Tabs

Description:

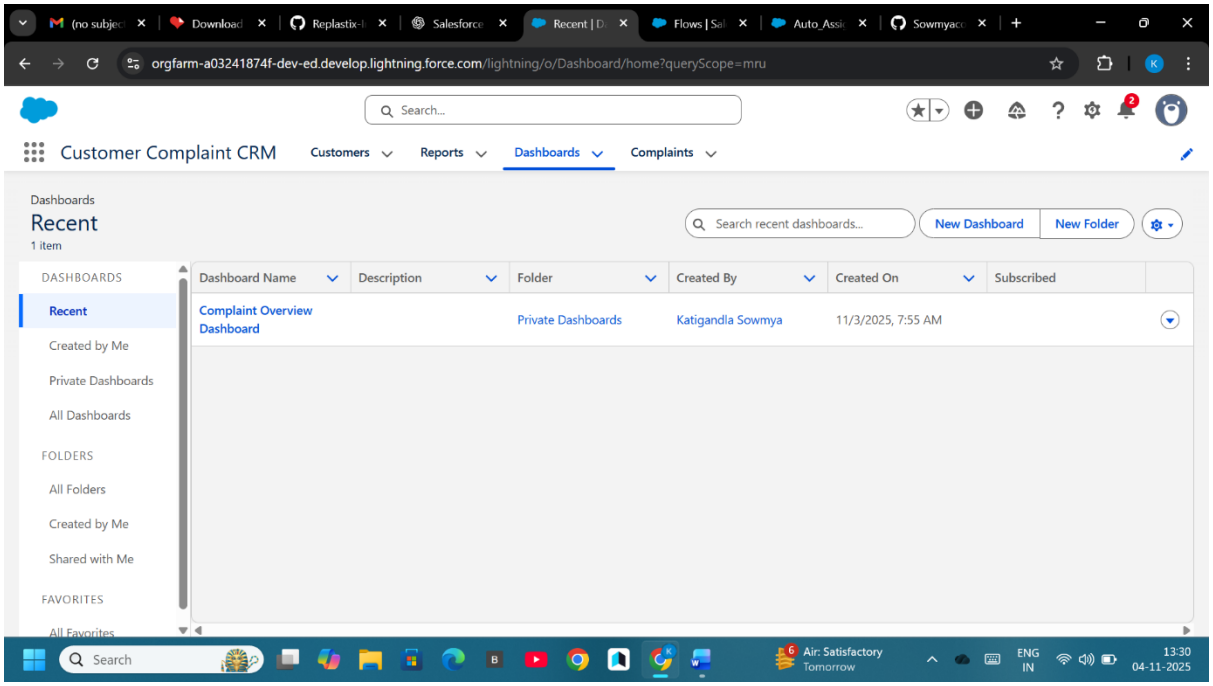
Tabs are used for easy navigation within the app.
Each object (Customer, Complaint, Reports) has its own tab.

Example:

Tabs added:

- **Customers**
- **Complaints**
- **Reports**
- **Dashboards**

So users can quickly move between data sections



🏠 4. Home Page Layout

Description:

The Home Page is the main dashboard of the app, showing important metrics and shortcuts.

Example Layout:

- Total Complaints Count
- Open Complaints
- High-Priority Complaints
- Pie Chart (Complaints by Status)
- Shortcut to “Create New Complaint”

Created using **Lightning App Builder → Home Page → Add Dashboard Components.**

🧰 5. Utility Bar

Description:

The Utility Bar provides quick access to tools that are always available at the bottom of the screen.

Example:

Added:

- “Quick Complaint” Button → to create a complaint directly
- “Notes” → for agents to take quick notes
- “Recent Items” → to access last viewed records easily

⚡ 6. Lightning Web Components (LWC)

Description:

Lightning Web Components (LWC) are reusable, lightweight, and fast components built using **HTML, JavaScript, and Apex**.
They help create dynamic interfaces.

Example Component:

MyComplaintsDashboard
Displays complaints assigned to the logged-in user.

Sample Apex Controller:

```
public with sharing class MyComplaintsController {
    @AuraEnabled(cacheable=true)
    public static List<Complaint__c> getMyComplaints() {
        return [SELECT Id, Name, Status__c, Priority__c
                FROM Complaint__c
                WHERE Assigned_Agent__c = :UserInfo.getUserId()];
    }
}
```

Sample LWC (myComplaintsDashboard.js):

```
import { LightningElement, wire } from 'lwc';
import getMyComplaints from '@salesforce/apex/MyComplaintsController.getMyComplaints';

export default class MyComplaintsDashboard extends LightningElement {
    @wire(getMyComplaints) complaints;
}
```

Template (myComplaintsDashboard.html):

```
<template>
    <lightning-card title="My Complaints">
        <template if:true={complaints.data}>
            <lightning-datatable
                key-field="Id"
                data={complaints.data}
                columns={[
                    { label: 'Complaint Name', fieldName: 'Name' },
                    { label: 'Status', fieldName: 'Status__c' },
                    { label: 'Priority', fieldName: 'Priority__c' }
```

```
            ]}>
          </lightning-datatable>
        </template>
        <template if:true={complaints.error}>
          <p>Error loading complaints</p>
        </template>
    </lightning-card>
</template>
```

□ 7. Apex Integration with LWC

Description:

LWC components use **Apex** to access Salesforce data when logic is complex.
This allows two-way communication between frontend (LWC) and backend (Apex).

Example:

When user clicks “Resolve Complaint” → Apex updates complaint status to “Resolved”.

🔁 8. Events in LWC

Description:

Events are used to communicate between components.

- **Child → Parent** using custom events
- **Parent → Child** using public properties

Example:

When complaint is resolved → event triggers dashboard refresh.

📡 9. Wire Adapters

Description:

The @wire service is used to automatically fetch Salesforce data from standard objects or Apex methods.

Example:

```
@wire(getRecord, { recordId: '$recordId', fields: ['Complaint__c.Status__c'] })
complaint;
```

□ 10. Imperative Apex Calls

Description:

Imperative calls let developers manually execute Apex methods (e.g., after clicking a button).

Example:

```
handleAssignAgent() {
    assignAgent({ complaintId: this.recordId })
        .then(() => alert('Agent Assigned Successfully'))
        .catch(error => console.log(error));
}
```

🚀 11. Navigation Service

Description:

Navigation Service is used to open records, list views, or pages from LWC code.

Example:

```
this[NavigationMixin.Navigate]({
    type: 'standard__recordPage',
    attributes: {
        recordId: complaintId,
        objectApiName: 'Complaint__c',
        actionName: 'view'
    }
});
```