Title: Integrated Customer Sales and Support System

Phase 6 – Salesforce UI Development Documentation

1. Objective

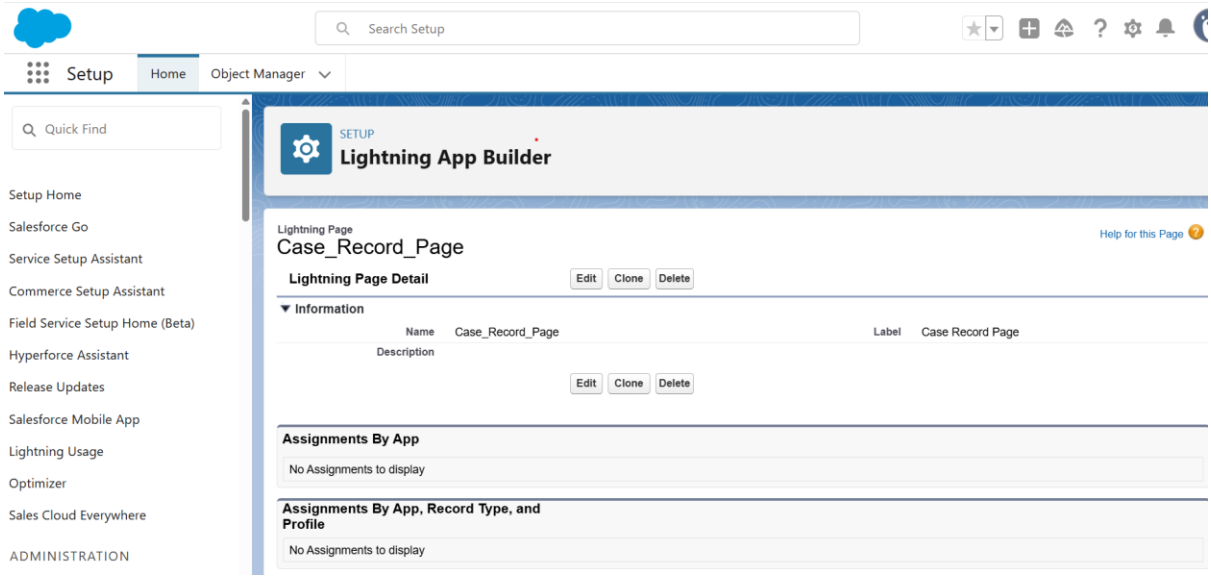
The goal of Phase 6 is to design and implement the **User Interface (UI)** for the Salesforce project.
This includes:

- Customizing **Record Pages**
- Building a **Home Page** with Lightning App Builder
- Adding **Tabs** and **Utility Bar** for user efficiency
- Using **Lightning Web Components (LWC)** to display dynamic data
- Connecting LWC with **Apex (Wire Adapters & Imperative Calls)**
- Activating and deploying these pages to users

2. Lightning App Builder

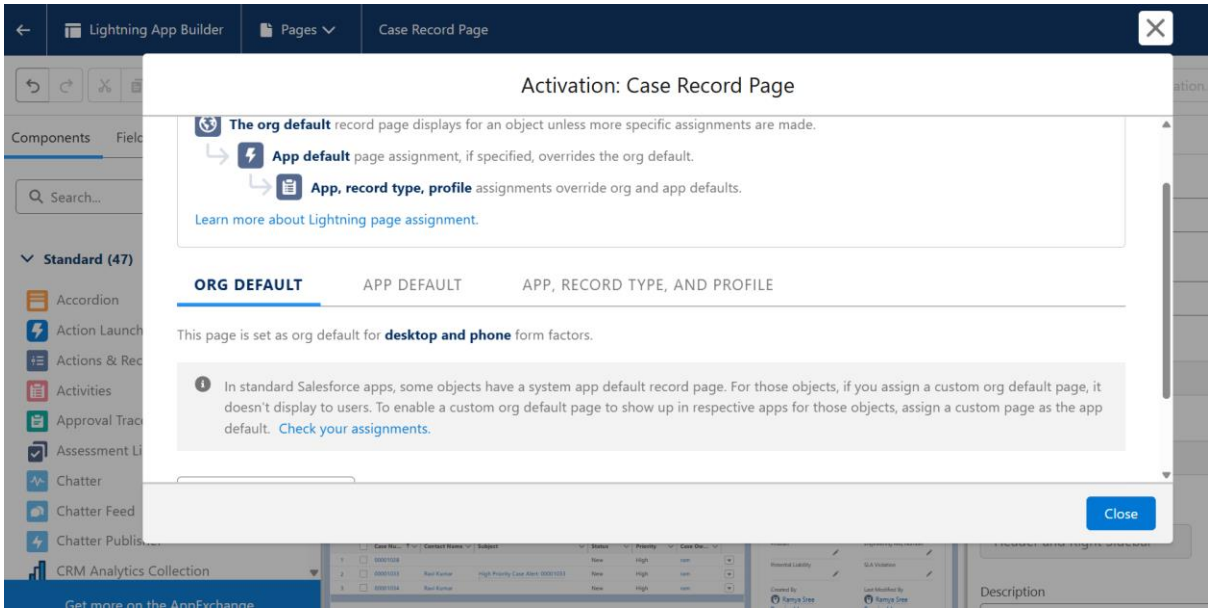
Steps:

1. Go to **Setup → User Interface → Lightning App Builder**.
2. Choose what you want to create:
    - **Record Page**
    - **Home Page**
    - **App Page**
3. Drag-and-drop components onto the page.



Steps Performed:

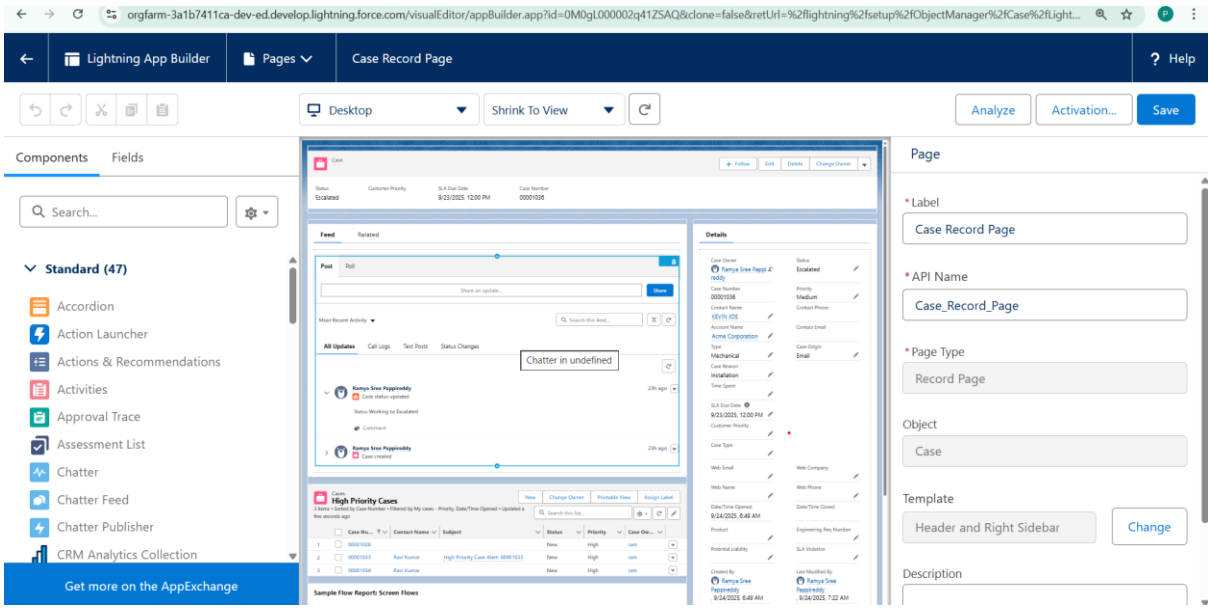
1. Navigated to **Setup → Lightning App Builder**.
2. Selected **New Page**.
3. Options available: App Page, Record Page, Home Page.
4. Selected **Record Page** for Case object.
5. Used **drag-and-drop interface** to add components like Highlights Panel, Tabs, Path, Related Lists, etc.
6. Saved and **Activated** the page as **Org Default** so that all users can view the new design.



3. Record Pages (Case Record Page)

Steps:

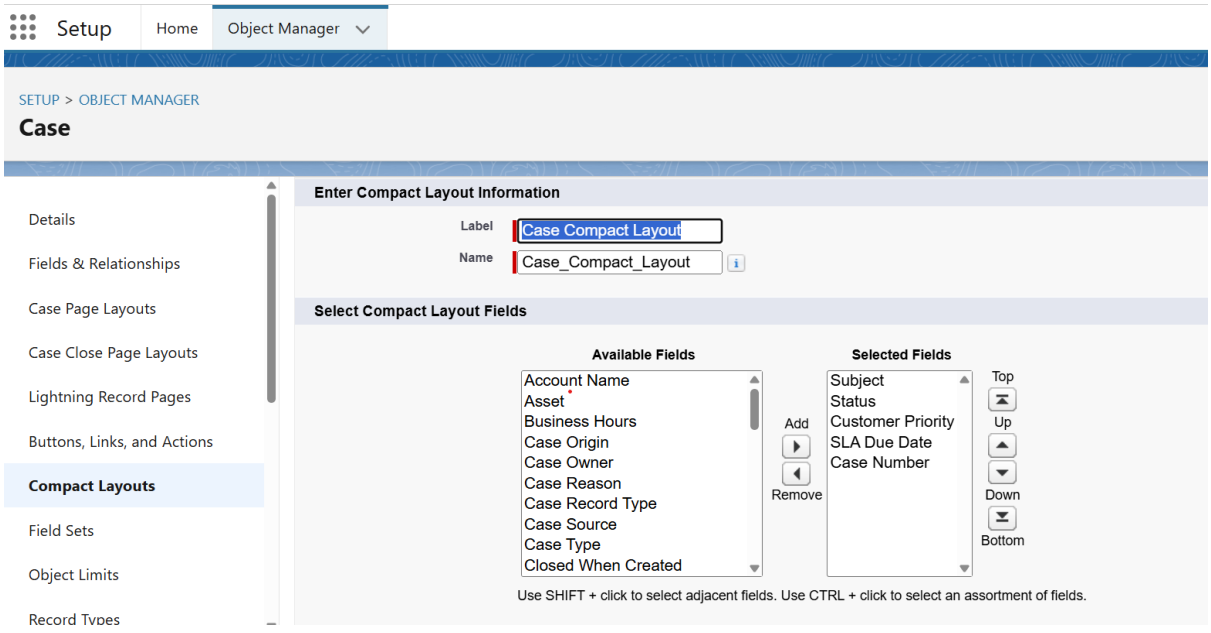
1. Go to **Setup → Object Manager → Case → Lightning Record Pages**.
2. Click **New** or edit an existing page.
3. Select **2-Column Layout**.
4. Drag the following components:
    - **Highlights Panel** → Top area (uses Compact Layout).
    - **Record Detail / Dynamic Forms** → Shows Case fields.
    - **Related Lists** → Contacts, Feedback, Account Products.
    - **Tabs** → Added **Details | Related | Activity | Feedback**.
    - **Path Component** → Shows Case Status stages.
    - **Custom LWC (c:highPriorityCases)** → Shows high-priority cases.
5. Click **Save**.
6. Click **Activation** → Set as **Org Default**.

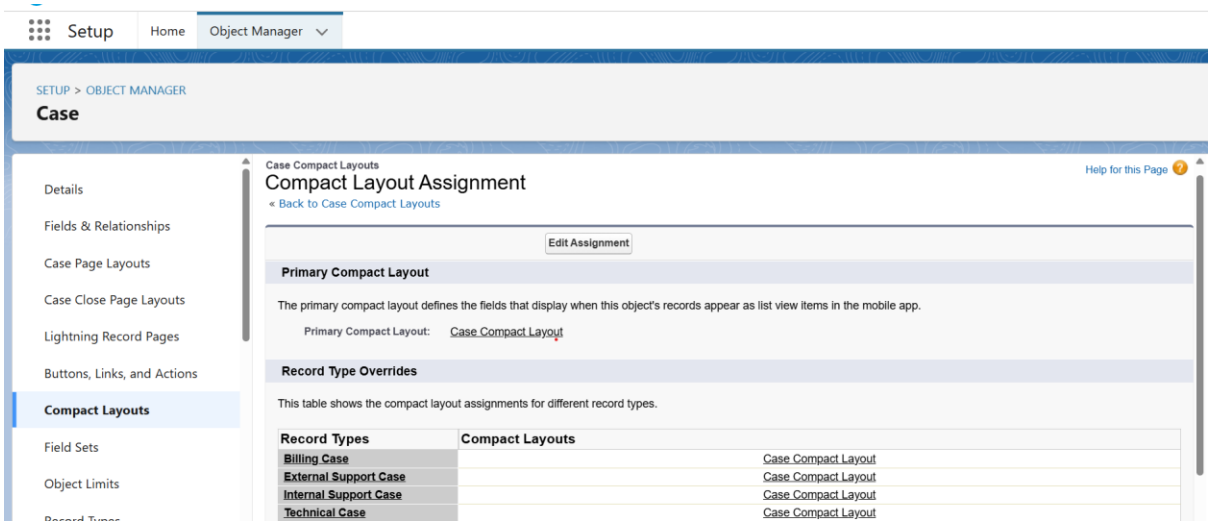


4. Compact Layout (Highlights Panel)

Steps:

1. Go to **Setup → Object Manager → Case → Compact Layouts**.
2. Click **New** (or edit an existing layout).
3. Add these fields:
    - Case Number
    - Subject
    - Status
    - Priority
    - Owner
4. Click **Save**.
5. Mark this layout as **Primary Compact Layout**.





5. Dynamic Actions

Steps:

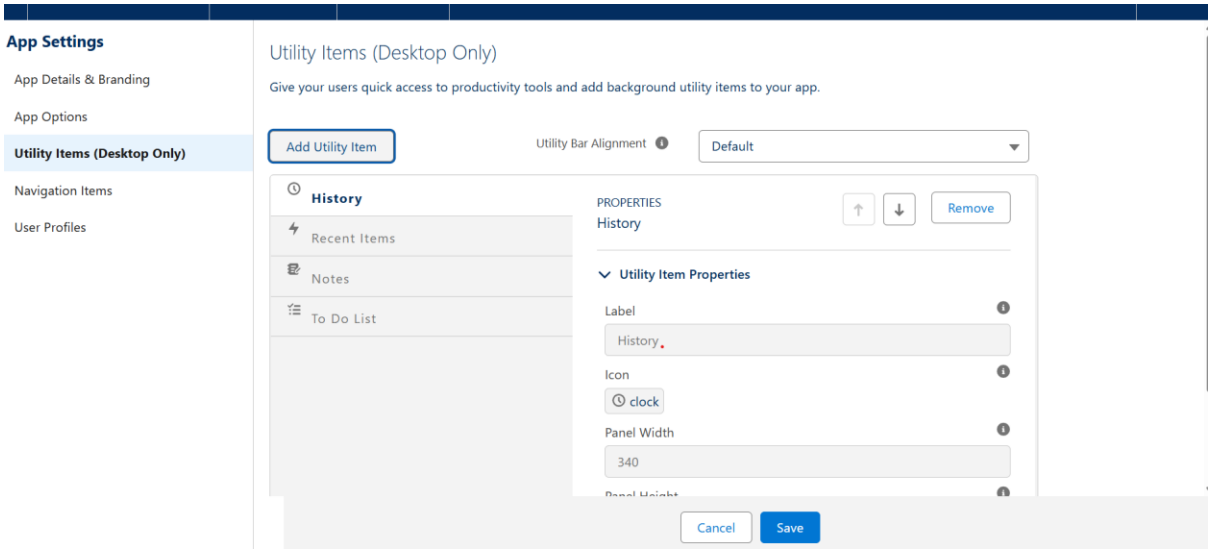
1. Open the Case Record Page in Lightning App Builder.
2. Click **Highlights Panel** → Enable **Upgrade to Dynamic Actions**.
3. Select **Source Page Layout** → (choose the existing Case Layout).
4. Migrate all actions (Edit, Close Case, Escalate, etc.).
5. Add **visibility rules** for actions (example: Escalate only when Status ≠ Closed).
6. Save & Activate.

5. Utility Bar

The **Utility Bar** was configured for quick access to tools.

Steps Performed:

1. Navigated to **Setup → App Manager**.
2. Opened the relevant **Lightning App**.
3. Edited the App and added the following Utility Bar Items:
    - **Notes** → For quick note-taking.
    - **Recent Items** → To quickly revisit last opened records.
    - **History** → To track navigation history.
4. Saved and assigned the Utility Bar for the app.
5. Save & Assign.



6. Lightning Web Components (LWC)

- ➢ **Not included in this project.**
- Reason: All UI requirements were successfully achieved using **standard Salesforce components** (Record Pages, Related Lists, Tabs, Path, Reports).
- Since no additional custom UI logic was required, no LWCs were created.

7. Apex with LWC

- ➢ **Not included in this project.**
- Reason: We did not build custom LWCs, so there was no need to integrate Apex classes.
- Direct configuration in Org was sufficient for project needs.

8. Events in LWC

- ➢ **Not included in this project.**
- Reason: No parent-child component communication was required. Standard components already displayed all necessary data.

9. Wire Adapters

- ➢ **Not included in this project.**
- Reason: Since no LWCs were created, we did not fetch data using @wire. Instead, we used built-in Record Detail and Related Lists components.

10. Imperative Apex Calls

- **Not included in this project.**

- Reason: No need to refresh or manually fetch records from Apex since Salesforce's standard components already updated data automatically.

11. Navigation Service

- **Not included in this project.**

- Reason: Navigation between records and list views was handled using **standard Lightning navigation**. No custom navigation logic was required in LWC.