Phase 6: User Interface Development

This phase covers Salesforce UI customization using Lightning App Builder, Lightning Web Components (LWC), and Apex integration. It enables building dynamic, user-friendly interfaces for Salesforce users.

1⃣ Lightning App Builder

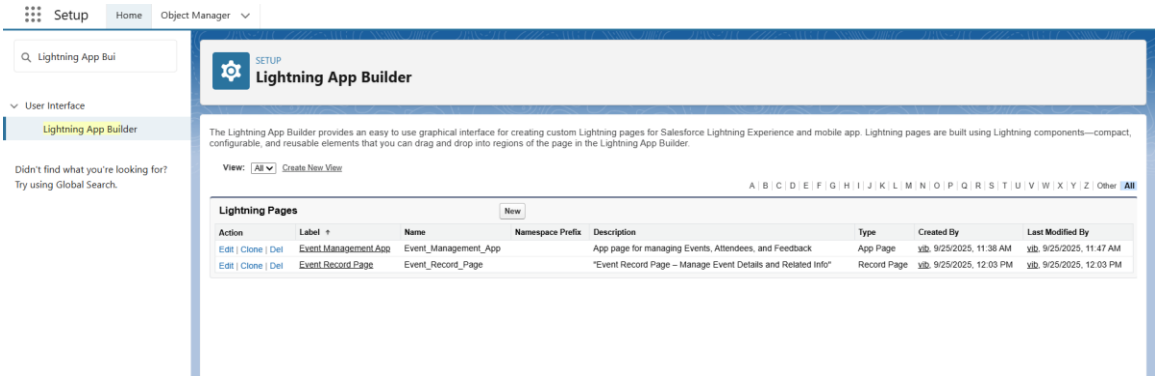
Purpose: Customize pages (Home, Record, App pages) using drag-and-drop components.

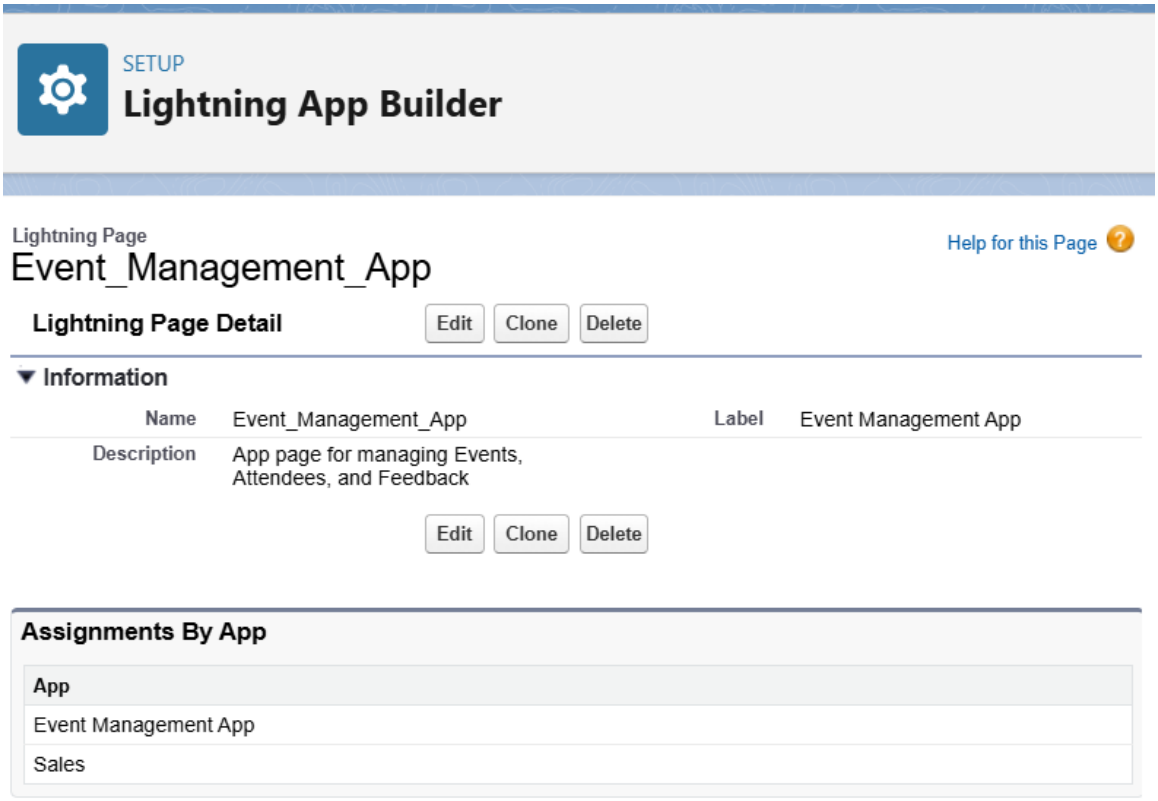
Steps:

- Go to Setup → Lightning App Builder.
- Click New → Choose Page Type (App Page, Home Page, Record Page).
- Enter Name & Description → Click Next.
- Choose Layout → Click Finish.
- Drag & drop standard or custom components.
- Click Save → Activate to make the page live.

Notes:

- Assign pages to Profiles or Apps.
- Use Preview to see the page before activation.





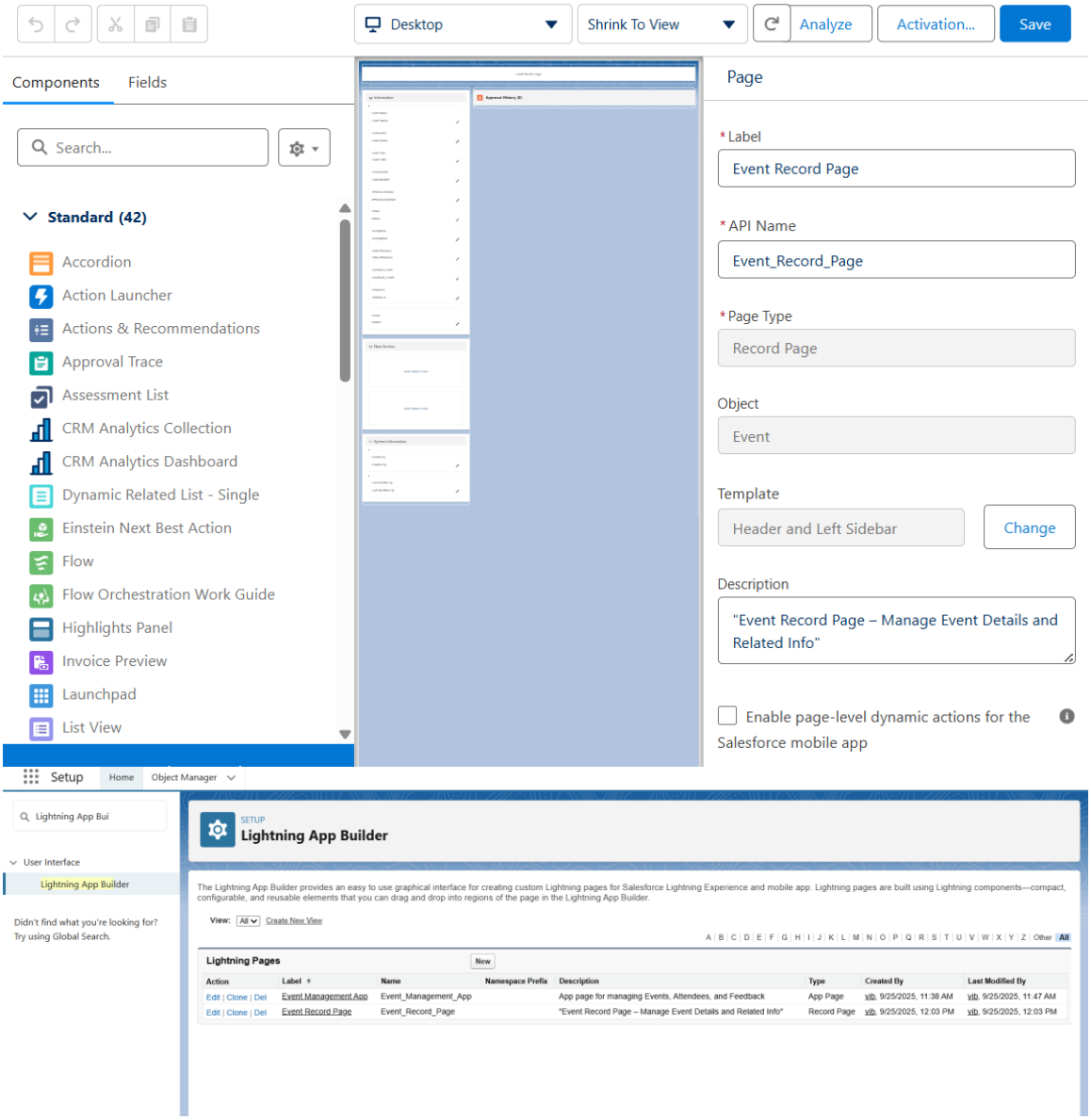
2️⃣ Record Pages

Purpose: Customize object record layouts.

Steps:

- Open Lightning App Builder → Record Page.
- Select Object → Existing Page / New Page.
- Configure Sections and add components: Related lists, Custom LWCs, Tabs.

- Click Save → Activate → Assign to App, Record Type, or Profile.



3️⃣ Tabs

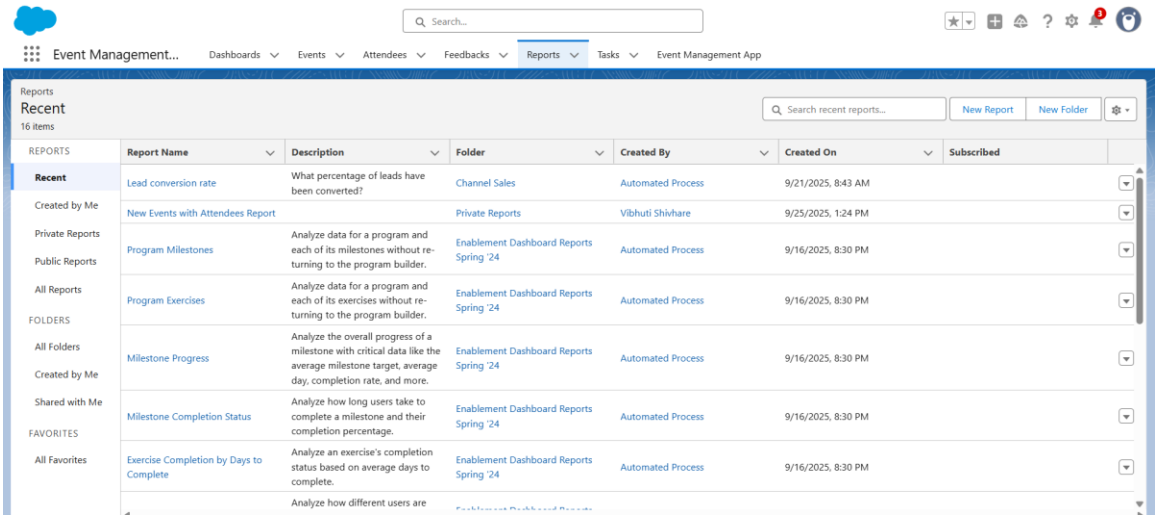
Purpose: Organize objects, components, or pages in a single app.

Steps:

- Go to App Manager → Edit App.
- Under Navigation Items, click Add Tab.
- Select Standard Object / Custom Object / Lightning Page.
- Save changes → Open app to verify tabs.

Notes:

- Control tab access using App Personalization Settings.
- Tabs can include Lightning Components, Visualforce Pages, or Reports.



4️⃣ Home Page Layouts

Purpose: Create dashboards or personalized home pages.

Steps:

- Go to Lightning App Builder → Home Page.
- Click New → Standard Home Page.
- Drag & drop components: Reports, Dashboards, Custom LWCs.
- Save → Activate → Assign to Profiles.

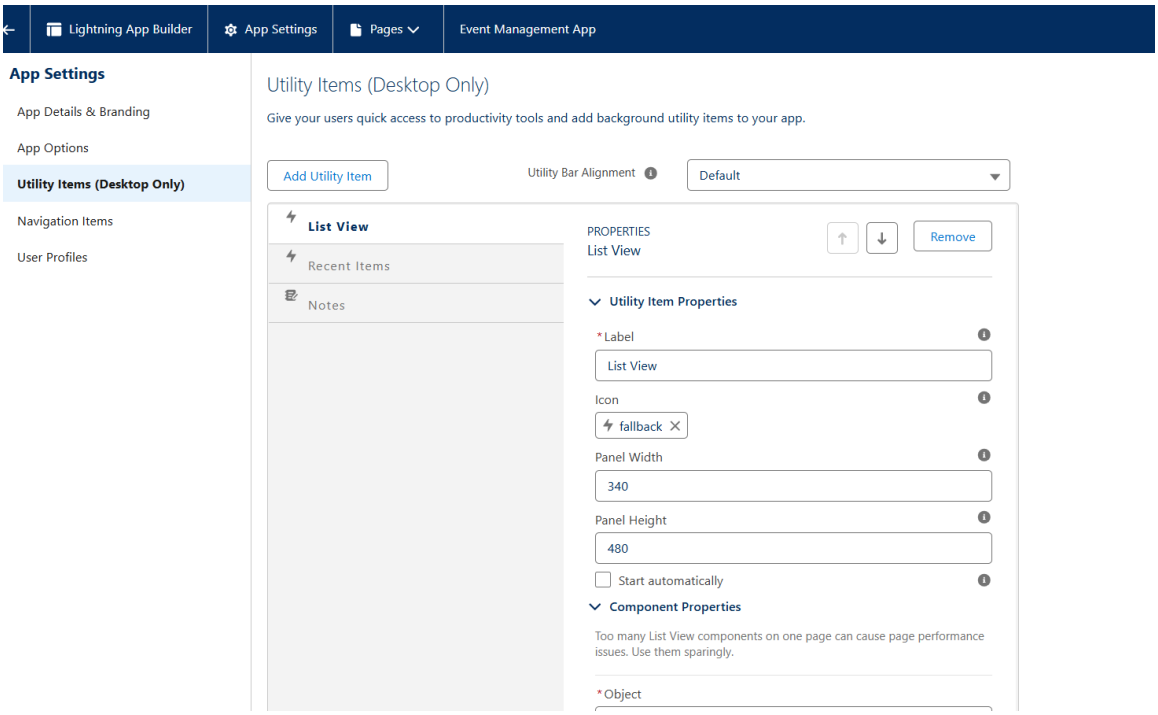
5️⃣ Utility Bar

Purpose: Quick access to tools at the bottom of the app.

Steps:

- Open App Manager → Edit App → Utility Bar.
- Click Add Utility Item.
- Choose component (e.g., Notes, Custom LWC, Flows).
- Configure Label, Icon, Size.

- Save & verify in app.



6️⃣ Lightning Web Components (LWC)

Purpose: Build reusable, responsive UI components.

Structure:

- .js → Component logic
- .html → Template
- .js-meta.xml → Metadata

Steps:

- Open VS Code → SFDX project → Create LWC.
- Implement logic & styling.
- Deploy using SFDX: Deploy.

Notes:

- LWCs can be used in App Builder, Record Pages, Tabs, or Utility Bar.

7️⃣ Apex with LWC

Purpose: Fetch or manipulate Salesforce data using Apex.

Steps:

- Write @AuraEnabled Apex methods.
- Call methods in LWC using Wire Adapter (reactive) or Imperative Call (on user action).
- Handle responses & errors in LWC JS.

8️⃣ Events in LWC

Purpose: Handle component communication.

Types:

- Custom Events → Child → Parent
- Lightning Message Service → Cross-component communication
- Standard DOM events → Click, Change, Input

Steps:

- Create & dispatch CustomEvent in child component.
- Listen in parent with on<eventname> attribute.

9️⃣ Wire Adapters

Purpose: Retrieve Salesforce data reactively.

Examples:

- @wire(getRecord, { recordId, fields })
- @wire(getObjectInfo, { objectApiName })

Notes:

- Wire automatically updates UI when data changes.
- Mostly for read-only operations.

🔟 Imperative Apex Calls

Purpose: Call Apex methods on-demand.

Steps:

- Import Apex method in LWC JS.
- Call method inside a function (e.g., button click).
- Handle Promise: myApexMethod({ param1: value }).then(result => {/* handle result */}).catch(error => {/* handle error */});

Notes:

- Good for user-triggered actions.

- Gives explicit control over when the call happens.

✅Best Practices

Keep LWCs modular and reusable.

Always handle errors in Apex calls.

Use Profiles / Permission Sets to control access.

Test Home Page and Record Page layouts for multiple profiles.

Deploy using Change Sets or SFDX for version control.

I mainly Concentrated on
• Lightning App Builder
• Record Pages
• Tabs
• Home Page Layouts
• Utility Bar