PHASE 6 — UI Development

🎯 Phase 6 Goal

Create a **role-friendly, hospital-style UI** so that:

- Receptionists book appointments fast
- Doctors see their schedules
- Managers track beds & availability

◇ STEP 6.1 — Create a Dedicated Lightning App (MOST IMPORTANT)

📍 Navigation

Setup → App Manager → New Lightning App

🧩 App Details

- **App Name:** Hospital Management
- **Developer Name:** Hospital_Management
- **Description:** Hospital Appointment & Bed Management CRM

Click **Next**

🎨 App Options

- Navigation Style → **Standard**
- Disable “Enable Setup” (optional)

Click **Next**

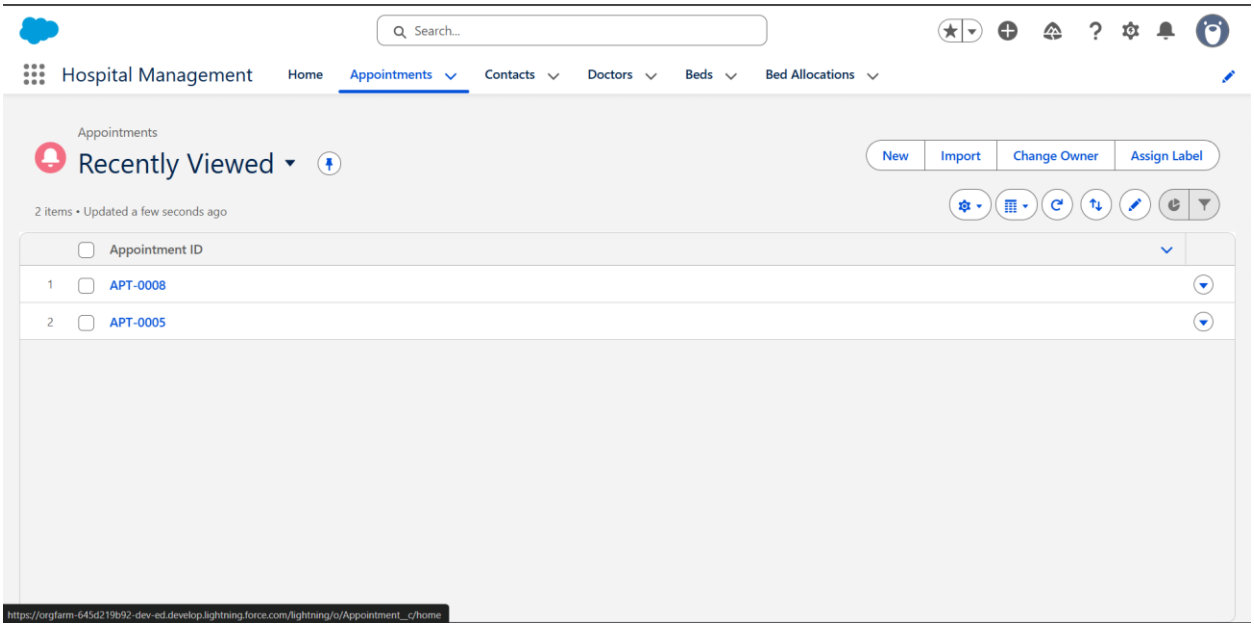
👥 Assign User Profiles

Select:

- System Administrator
- Hospital Manager
- Receptionist

Click **Save & Finish**

✅ You now have a **dedicated hospital app**



◇ STEP 6.2 — Create Custom Tabs (Core Objects)

📍 Navigation

Setup → Tabs → New

📝 Create these Tabs

1️⃣ *Doctor Tab*

- Object → `Doctor__c`
- Tab Style → Medical icon
- Visibility → Default On

2️⃣ *Appointment Tab*

- Object → `Appointment__c`
- Tab Style → Calendar icon
- Visibility → Default On

3️⃣ *Bed Tab*

- Object → `Bed__c` *(if created earlier)*
- Tab Style → Hospital icon

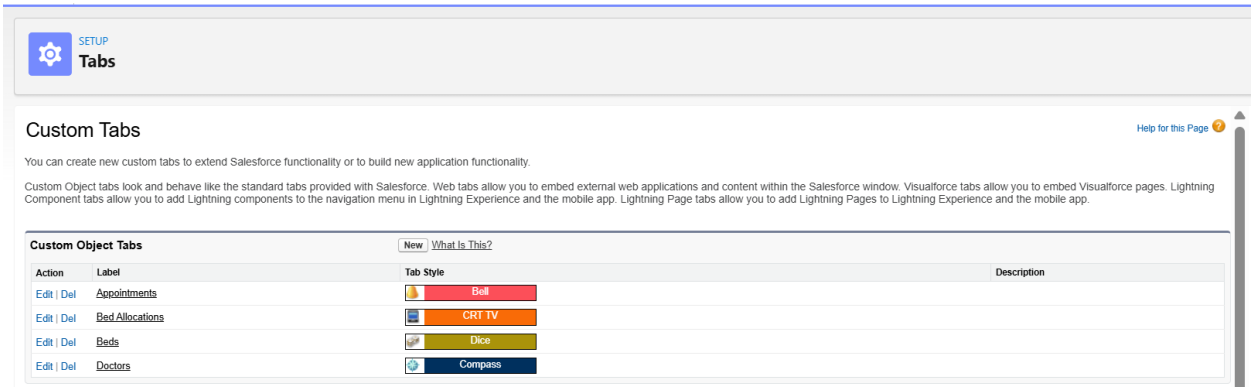
📌 Add Tabs to App

App Manager → Hospital Management → Edit → Navigation Items

Add:

- Doctors
- Appointments
- Beds
- Contacts (Patients)

Remove unnecessary tabs.



◇ STEP 6.3 — Doctor Record Page (Lightning App Builder)

Navigation

Setup → Object Manager → Doctor → Lightning Record Pages → New

Page Type

- Record Page
- Label: **Doctor Record Page**

Layout Design

Header

- Highlights Panel
    - Doctor Name
    - Availability Status
    - Specialization

Left Column

- Record Detail (Compact)

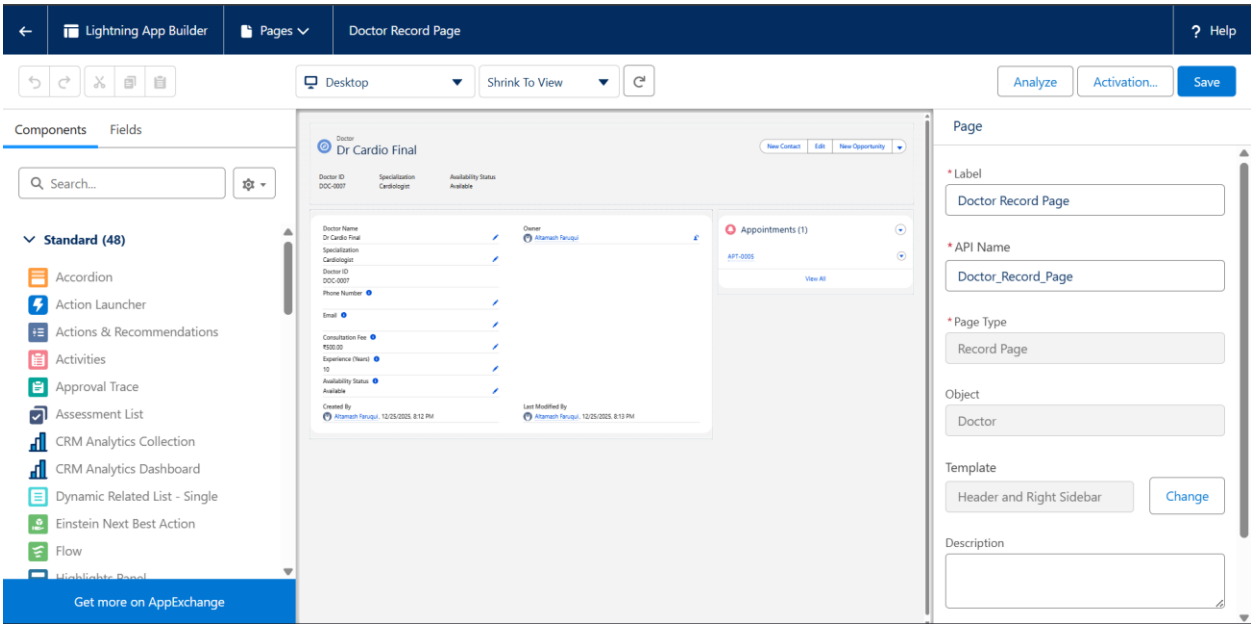
Right Column

- Related List – Single
    - Appointments

Page Assignment

- Assign to **Hospital Management App**
- Profiles:
    - Doctor
    - Hospital Manager

Save & Activate



◇ STEP 6.4 — Appointment Record Page (MOST USED)

Navigation

Setup → Object Manager → Appointment → Lightning Record Pages → New

Layout Structure

- Header + Two Column

Header

- Highlights Panel
  - Status
  - Appointment Date
  - Doctor
  - Patient

Left Column

- Record Details
- Activity Timeline

Right Column

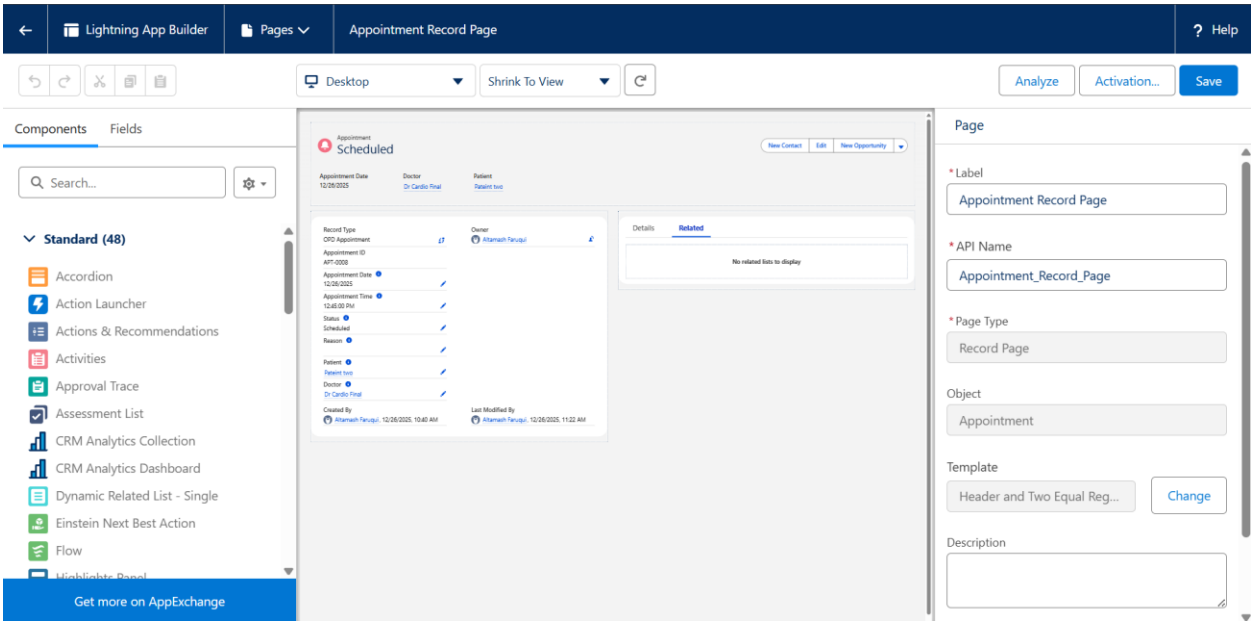
- Related Lists
  - Doctor
  - Patient

Optional Automation Component

- Add **Path** (Status)
    - Scheduled → Completed → Cancelled

Activate for:

- Receptionist
- Hospital Manager



STEP 6.5 — Home Page (Hospital Dashboard Lite)

Navigation

Setup → Lightning App Builder → New → Home Page

Components to Add

Top Section

- Assistant

- Rich Text:

“Welcome to Hospital Management System”

Middle Section

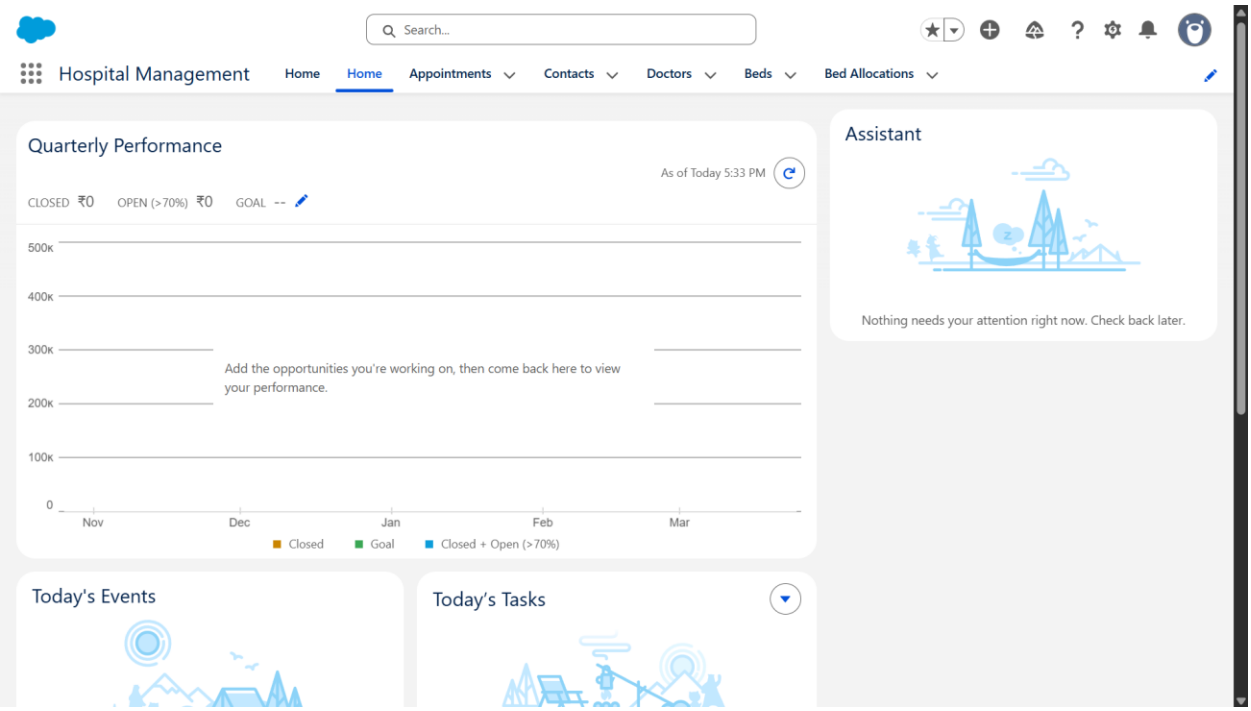
- Report Chart:
    - Today’s Appointments
    - Available Beds

Bottom Section

- To-Do List
- Recent Appointments

Assign Home Page

- App: Hospital Management
- Profiles:
    - Receptionist
    - Hospital Manager



◇ STEP 6.6 — Utility Bar (POWER FEATURE)

Navigation

App Manager → Hospital Management → Edit → Utility Items

Add Utilities

1 Notes

Quick patient notes

2 Recent Items

Fast access to Doctors/Appointments

6️⃣ Lightning Web Components (LWC)

🎯 Purpose

Build **reusable**, **responsive**, and **high-performance** UI components in Salesforce.

🧱 Structure

File	Description
`.js`	Component logic
`.html`	UI template
`.js-meta.xml`	Metadata & exposure settings

🛠️ Steps

1. Open **VS Code** → SFDX Project
2. Create **Lightning Web Component**
3. Implement business logic & styling
4. Deploy using **SFDX: Deploy**

📝 Notes

- LWCs can be used in:
  - Lightning App Builder
  - Record Pages
  - Tabs
  - Utility Bar

7️⃣ Apex with LWC

🎯 Purpose

Fetch or manipulate **Salesforce data** using Apex from LWC.

🛠️ Steps

1. Write Apex methods using `@AuraEnabled`
2. Call Apex from LWC using:
    a. **Wire Adapter** → Reactive data
    b. **Imperative Call** → User-triggered actions
3. Handle **responses & errors** in LWC JavaScript

8️⃣ Events in LWC

🎯 Purpose

Enable **communication between components**.

🔁 Types

Event Type	Use Case
Custom Events	Child → Parent communication
Lightning Message Service	Cross-component communication
Standard DOM Events	Click, Change, Input

🛠️ Steps

1. Create and dispatch `CustomEvent` in child component
2. Listen in parent using `on<eventname>` attribute

9️⃣ Wire Adapters

🎯 Purpose

Retrieve Salesforce data **reactively**.

📌 Examples

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

📝 Notes

- Automatically refreshes UI when data changes
- Best suited for **read-only operations**

◇ Imperative Apex Calls

🎯 Purpose

Call Apex methods **on demand** (user actions).

🛠️ Steps

1. Import Apex method in LWC JS
2. Call method inside a function (e.g., button click)
3. Handle Promise:

```
myApexMethod({ param1: value })
  .then(result => {
    // handle result
  })
  .catch(error => {
    // handle error
  });
```

📝 Notes

- Best for **user-triggered actions**
- Provides **explicit control** over execution

☑ Best Practices

✓ Keep LWCs **modular and reusable**

✓ Always handle **errors** in Apex calls

✓ Use **Profiles / Permission Sets** for access control

✓ Test Home & Record Pages across multiple profiles

✓ Deploy using **Change Sets or SFDX** for version control

★ Main Focus Areas

I mainly concentrated on:

- **Lightning App Builder**
- **Record Pages**
- **Tabs**
- **Home Page Layouts**
- **Utility Bar**