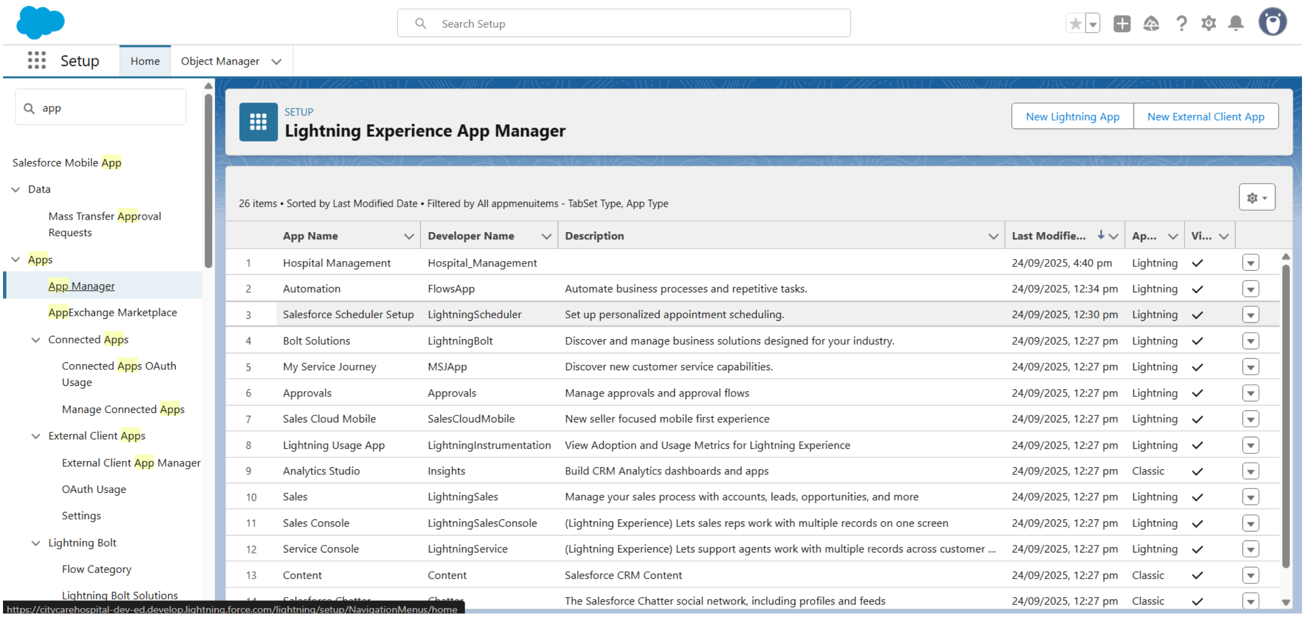
Phase 6: User Interface Development

1. Lightning App Builder

- Created the **Hospital App** to serve as the main container for all custom objects, tabs, and pages.
- This provides one unified workspace for users such as Doctors, Patients, and Receptionists.



Lightning Experience App Manager

26 items • Sorted by Last Modified Date • Filtered by All appmenuitems - TabSet Type, App Type

	App Name	Developer Name	Description	Last Modifie...	Ap...	Vi...
1	Hospital Management	Hospital_Management		24/09/2025, 4:40 pm	Lightning	✓
2	Automation	FlowsApp	Automate business processes and repetitive tasks.	24/09/2025, 12:34 pm	Lightning	✓
3	Salesforce Scheduler Setup	LightningScheduler	Set up personalized appointment scheduling.	24/09/2025, 12:30 pm	Lightning	✓
4	Bolt Solutions	LightningBolt	Discover and manage business solutions designed for your industry.	24/09/2025, 12:27 pm	Lightning	✓
5	My Service Journey	MSJApp	Discover new customer service capabilities.	24/09/2025, 12:27 pm	Lightning	✓
6	Approvals	Approvals	Manage approvals and approval flows	24/09/2025, 12:27 pm	Lightning	✓
7	Sales Cloud Mobile	SalesCloudMobile	New seller focused mobile first experience	24/09/2025, 12:27 pm	Lightning	✓
8	Lightning Usage App	LightningInstrumentation	View Adoption and Usage Metrics for Lightning Experience	24/09/2025, 12:27 pm	Lightning	✓
9	Analytics Studio	Insights	Build CRM Analytics dashboards and apps	24/09/2025, 12:27 pm	Classic	✓
10	Sales	LightningSales	Manage your sales process with accounts, leads, opportunities, and more	24/09/2025, 12:27 pm	Lightning	✓
11	Sales Console	LightningSalesConsole	(Lightning Experience) Lets sales reps work with multiple records on one screen	24/09/2025, 12:27 pm	Lightning	✓
12	Service Console	LightningService	(Lightning Experience) Lets support agents work with multiple records across customer ...	24/09/2025, 12:27 pm	Lightning	✓
13	Content	Content	Salesforce CRM Content	24/09/2025, 12:27 pm	Classic	✓
14	Salesforce Chatter	Chatter	The Salesforce Chatter social network, including profiles and feeds	24/09/2025, 12:27 pm	Classic	✓

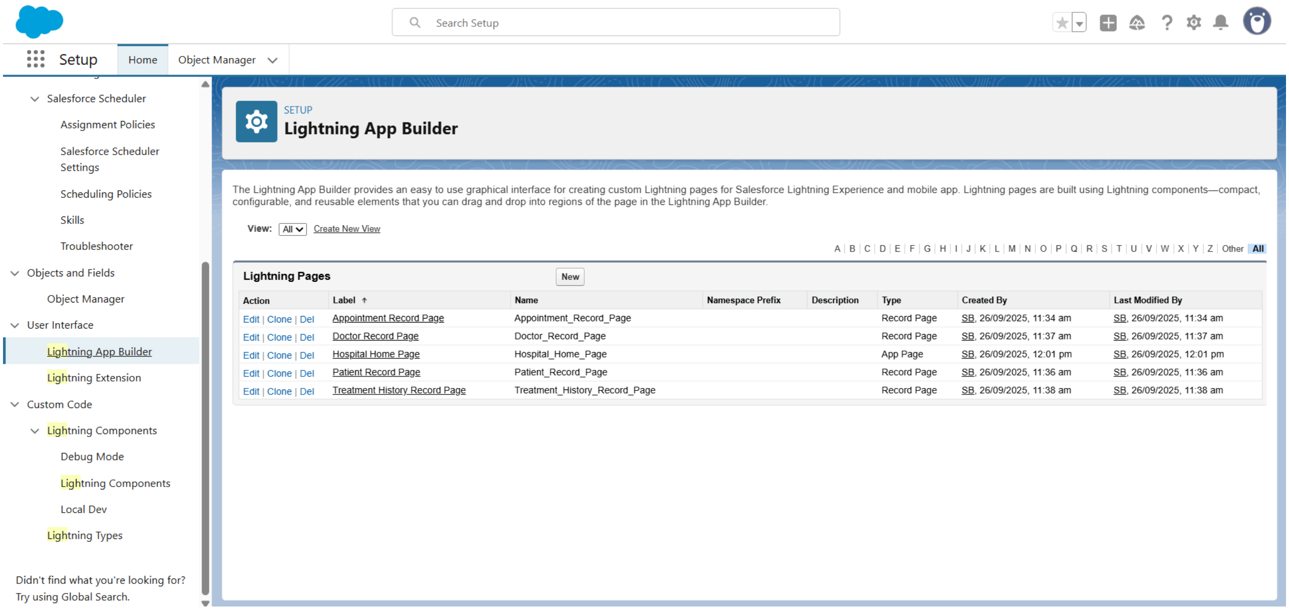
https://citycarehospital-dev-ed.develop.lightning.force.com/lightning/setup/NavigationMenus/home

2. Record Pages

- Created four record pages, each linked to its respective object:
  - Appointment Record Page → Object: `Appointment__c`
  - Patient Record Page → Object: `Patient__c`
  - Doctor Record Page → Object: `Doctor__c`

    - Treatment History Record Page → Object: `Treatment_History__c`

- Used the “Header and One Region” layout for clarity.
- Added **Highlights Panel** and **Record Detail** to display important information directly.



Lightning App Builder

The Lightning App Builder provides an easy to use graphical interface for creating custom Lightning pages for Salesforce Lightning Experience and mobile app. Lightning pages are built using Lightning components—compact, configurable, and reusable elements that you can drag and drop into regions of the page in the Lightning App Builder.

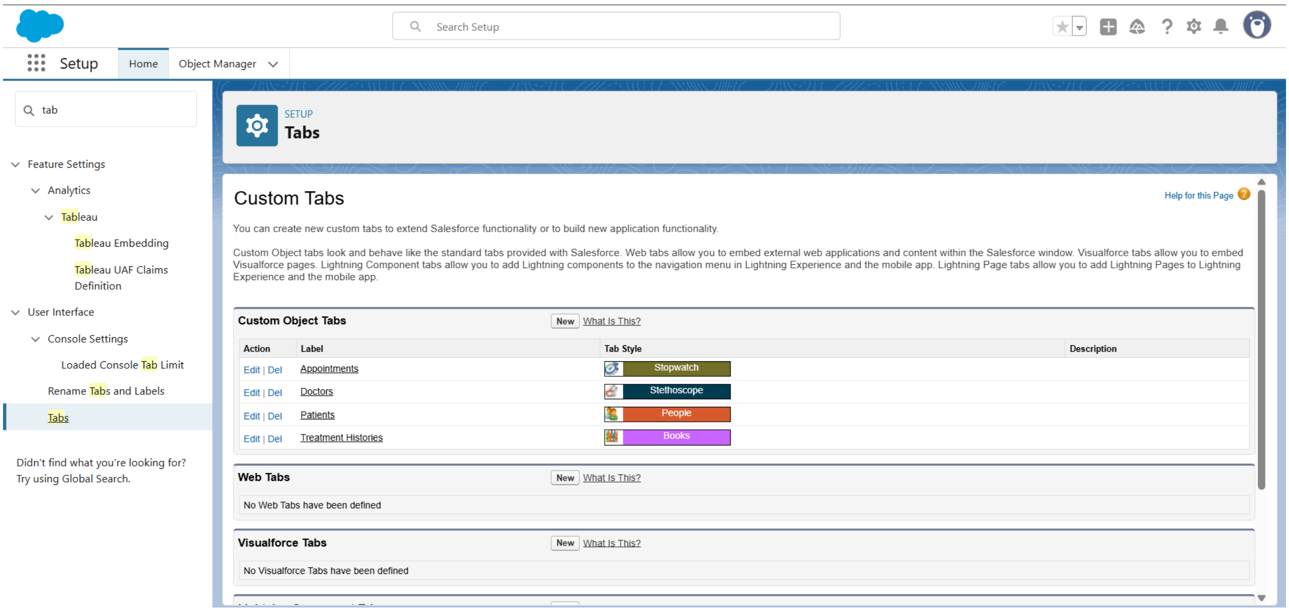
View: All Create New View

Lightning Pages

Action	Label ↑	Name	Namespace Prefix	Description	Type	Created By	Last Modified By
Edit | Clone | Del	Appointment Record Page	Appointment_Record_Page			Record Page	SB, 26/09/2025, 11:34 am	SB, 26/09/2025, 11:34 am
Edit | Clone | Del	Doctor Record Page	Doctor_Record_Page			Record Page	SB, 26/09/2025, 11:37 am	SB, 26/09/2025, 11:37 am
Edit | Clone | Del	Hospital Home Page	Hospital_Home_Page			App Page	SB, 26/09/2025, 12:01 pm	SB, 26/09/2025, 12:01 pm
Edit | Clone | Del	Patient Record Page	Patient_Record_Page			Record Page	SB, 26/09/2025, 11:36 am	SB, 26/09/2025, 11:36 am
Edit | Clone | Del	Treatment History Record Page	Treatment_History_Record_Page			Record Page	SB, 26/09/2025, 11:38 am	SB, 26/09/2025, 11:38 am

3. Tabs

- Defined **Custom Object Tabs** for easy navigation inside the Hospital App:
    - Appointments
    - Patients
    - Doctors
    - Treatment Histories
- Each tab links directly to the respective custom object.



Didn't find what you're looking for? Try using Global Search.

Custom Tabs

You can create new custom tabs to extend Salesforce functionality or to build new application functionality.

Custom Object tabs look and behave like the standard tabs provided with Salesforce. Web tabs allow you to embed external web applications and content within the Salesforce window. Visualforce tabs allow you to embed Visualforce pages. Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app. Lightning Page tabs allow you to add Lightning Pages to Lightning Experience and the mobile app.

Custom Object Tabs New What Is This?

Action	Label	Tab Style	Description
Edit | Del	Appointments	Stopwatch	
Edit | Del	Doctors	Stethoscope	
Edit | Del	Patients	People	
Edit | Del	Treatment Histories	Books	

Web Tabs New What Is This?

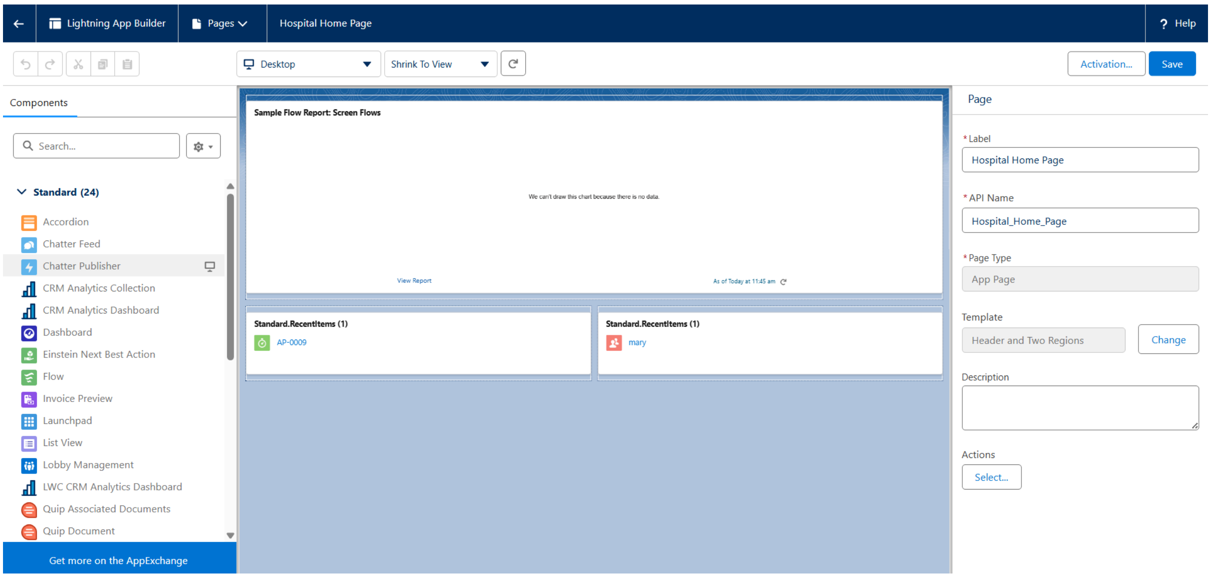
No Web Tabs have been defined

Visualforce Tabs New What Is This?

No Visualforce Tabs have been defined

4. Home Page Layout

- Created the **Hospital Home Page** using Lightning App Builder.
- Added the following components:
  - Recent Items (default component)
  - Report Chart → linked to “Appointments by Status” report for quick insights



Sample Flow Report: Screen Flows

We can't draw this chart because there is no data.

Standard.RecentItems (1)
AP-0009

Standard.RecentItems (1)
mary

* Label: Hospital Home Page

* API Name: Hospital_Home_Page

* Page Type: App Page

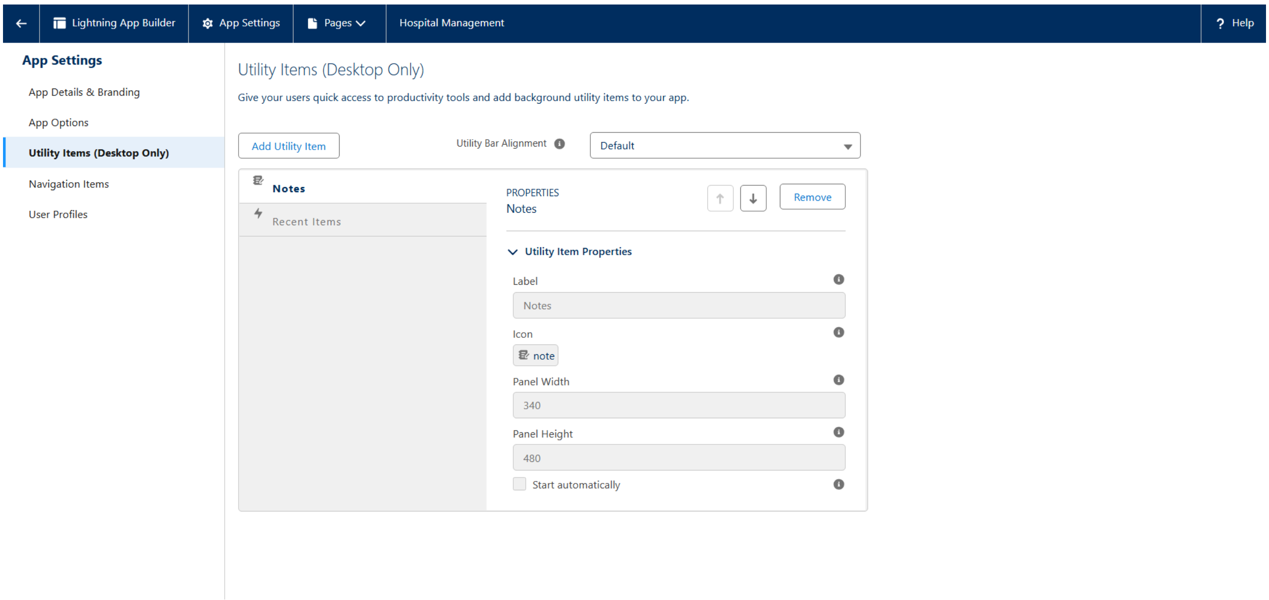
Template: Header and Two Regions

Description:

Actions:

5. Utility Bar

- Configured the **Utility Bar** for the Hospital App.
- Added utility items such as:
  - Notes
  - Recent Items



Utility Items (Desktop Only)

Give your users quick access to productivity tools and add background utility items to your app.

Utility Bar Alignment: Default

PROPERTIES
Notes

Utility Item Properties

Label: Notes

Icon: note

Panel Width: 340

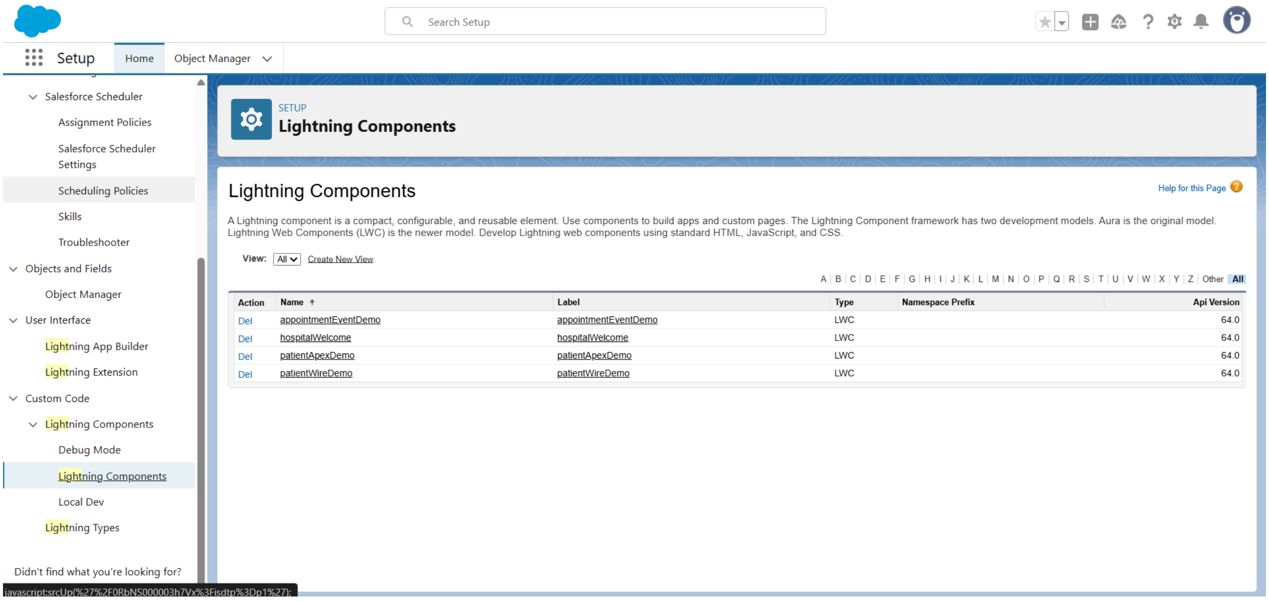
Panel Height: 480

□ Start automatically

6. Lightning Web Components (LWC)

We implemented custom **LWC components** for the Hospital App.

1. **hospitalWelcome** → Displays a welcome message.
2. **patientWireDemo** → Uses **@wire** to fetch Patient data from Apex.
3. **patientApexDemo** → Demonstrates **Imperative Apex calls** for Patients.
4. **appointmentEventDemo** → Demonstrates **LWC Events**.



Lightning Components

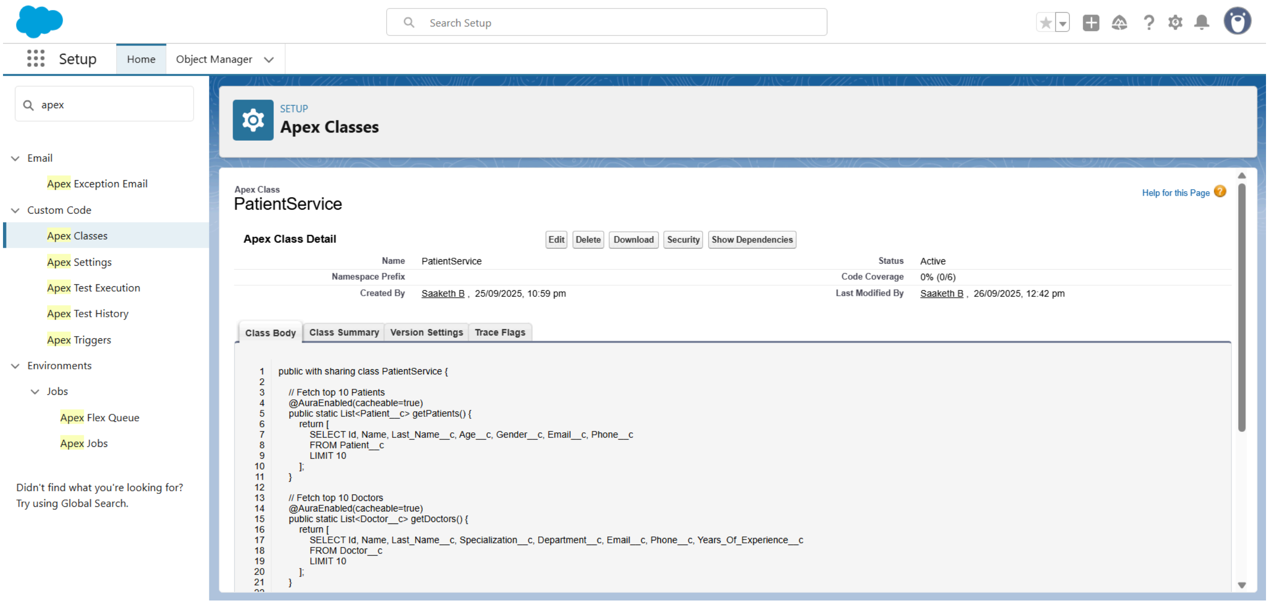
A Lightning component is a compact, configurable, and reusable element. Use components to build apps and custom pages. The Lightning Component framework has two development models. Aura is the original model. Lightning Web Components (LWC) is the newer model. Develop Lightning web components using standard HTML, JavaScript, and CSS.

View: All Create New View

Action	Name ↑	Label	Type	Namespace Prefix	Api Version
Del	appointmentEventDemo	appointmentEventDemo	LWC		64.0
Del	hospitalWelcome	hospitalWelcome	LWC		64.0
Del	patientApexDemo	patientApexDemo	LWC		64.0
Del	patientWireDemo	patientWireDemo	LWC		64.0

7. Apex with LWC

- Connected LWCs to Apex classes for dynamic data.
- Used `PatientService` Apex class to fetch Patients, Doctors, and Appointments data for LWCs.



Apex Class
PatientService

Apex Class Detail

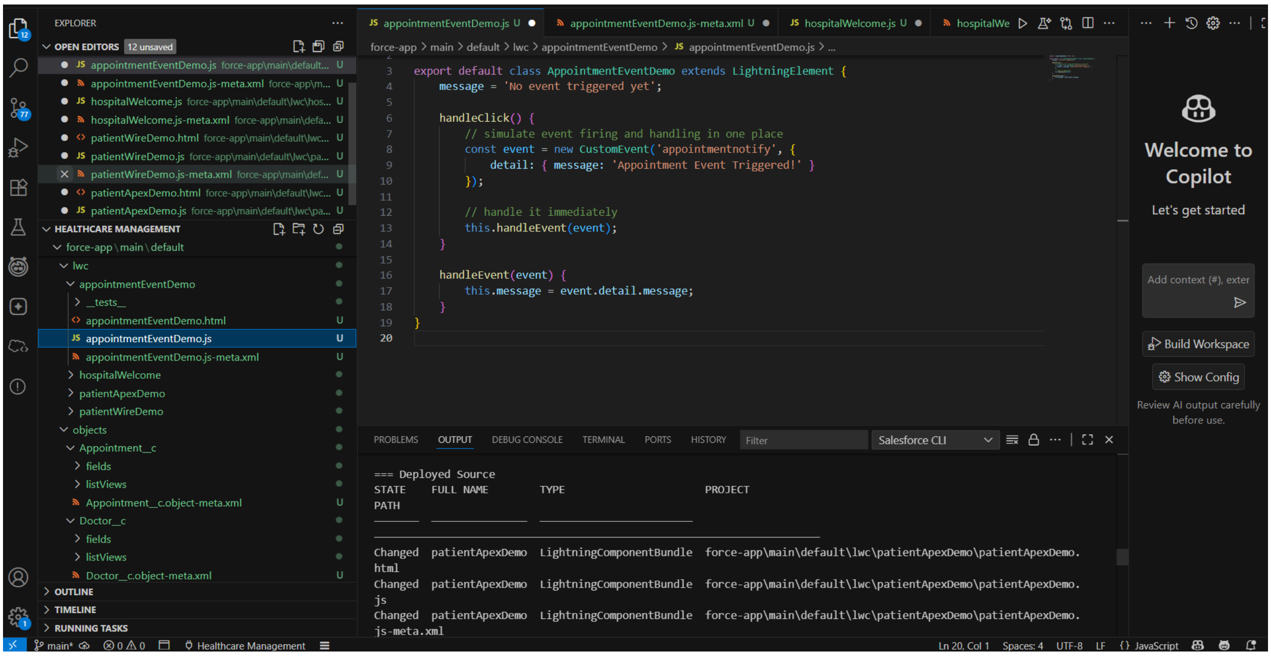
Name	PatientService	Status	Active
Namespace Prefix		Code Coverage	0% (0/6)
Created By	Saaketh B , 25/09/2025, 10:59 pm	Last Modified By	Saaketh B , 26/09/2025, 12:42 pm

```
public with sharing class PatientService {

    // Fetch top 10 Patients
    @AuraEnabled(cacheable=true)
    public static List<Patient__c> getPatients() {
        return [
            SELECT Id, Name, Last_Name__c, Age__c, Gender__c, Email__c, Phone__c
            FROM Patient__c
            LIMIT 10
        ];
    }

    // Fetch top 10 Doctors
    @AuraEnabled(cacheable=true)
    public static List<Doctor__c> getDoctors() {
        return [
            SELECT Id, Name, Last_Name__c, Specialization__c, Department__c, Email__c, Phone__c, Years_Of_Experience__c
            FROM Doctor__c
            LIMIT 10
        ];
    }
```

8. Events in LWC

- Implemented a **simple event demo**.

- Example: A button in an LWC fires an event, and another component listens to it for interaction.



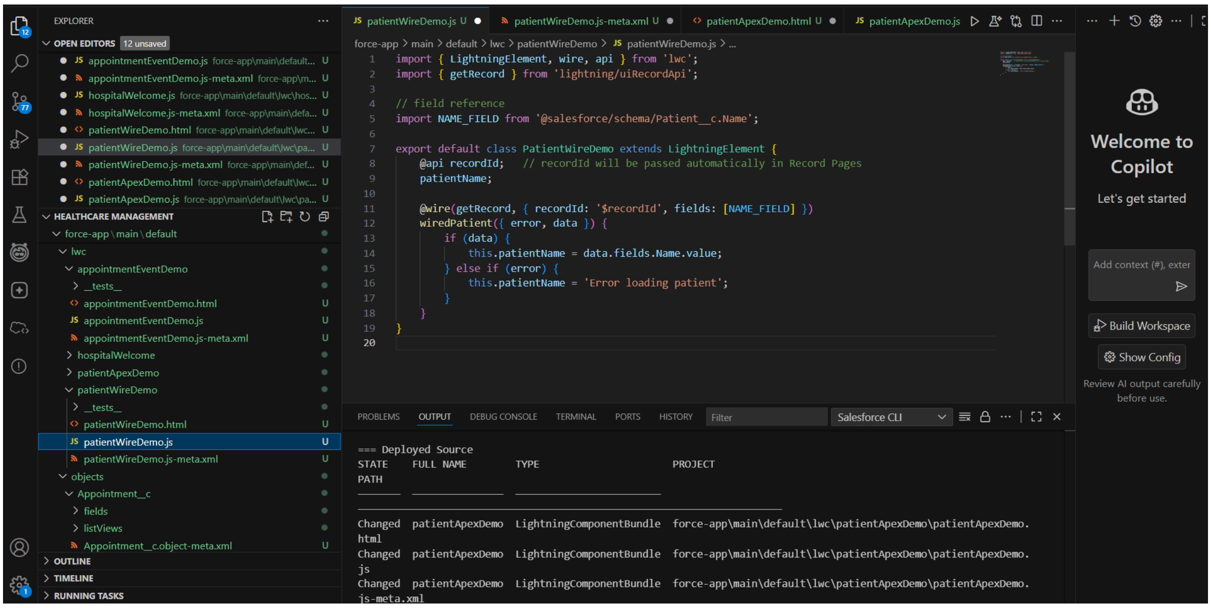
```
export default class AppointmentEventDemo extends LightningElement {
    message = 'No event triggered yet';

    handleClick() {
        // simulate event firing and handling in one place
        const event = new CustomEvent('appointmentnotify', {
            detail: { message: 'Appointment Event Triggered!' }
        });

        // handle it immediately
        this.handleEvent(event);
    }

    handleEvent(event) {
        this.message = event.detail.message;
    }
}
```

```
=== Deployed Source
STATE    FULL NAME        TYPE                         PROJECT
PATH
_______  _______________  ___________________________

_____________________________________________________________________________

Changed  patientApexDemo  LightningComponentBundle  force-app\main\default\lwc\patientApexDemo\patientApexDemo.
html
Changed  patientApexDemo  LightningComponentBundle  force-app\main\default\lwc\patientApexDemo\patientApexDemo.
js
Changed  patientApexDemo  LightningComponentBundle  force-app\main\default\lwc\patientApexDemo\patientApexDemo.
js-meta.xml
```

9. Wire Adapters

- Used **@wire decorator** in LWCs to call Apex methods and display records dynamically.

- Example: Appointment list automatically fetches data via wire.

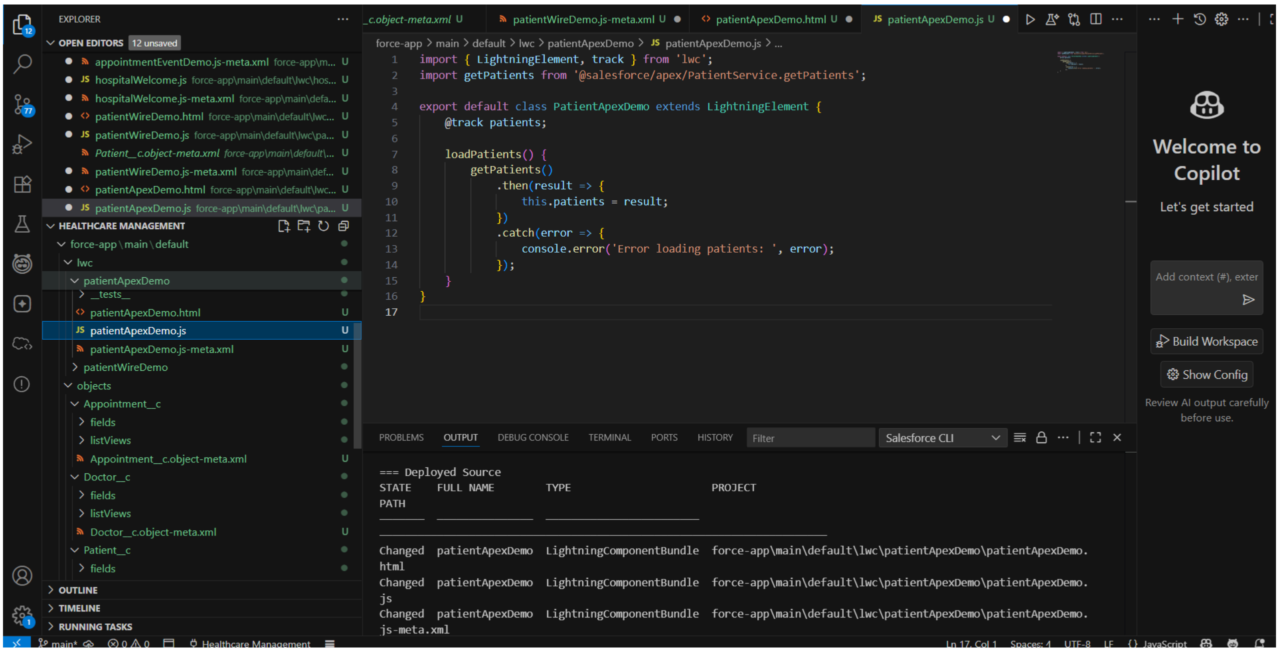


```javascript
import { LightningElement, wire, api } from 'lwc';
import { getRecord } from 'lightning/uiRecordApi';

// field reference
import NAME_FIELD from '@salesforce/schema/Patient__c.Name';

export default class PatientWireDemo extends LightningElement {
    @api recordId;   // recordId will be passed automatically in Record Pages
    patientName;

    @wire(getRecord, { recordId: '$recordId', fields: [NAME_FIELD] })
    wiredPatient({ error, data }) {
        if (data) {
            this.patientName = data.fields.Name.value;
        } else if (error) {
            this.patientName = 'Error loading patient';
        }
    }
}
```

```
=== Deployed Source
STATE    FULL NAME       TYPE                       PROJECT
PATH
───────  ──────────────  ─────────────────────────

───────────────────────────────────────────────────────────────────────────
Changed  patientApexDemo  LightningComponentBundle  force-app\main\default\lwc\patientApexDemo\patientApexDemo.
html
Changed  patientApexDemo  LightningComponentBundle  force-app\main\default\lwc\patientApexDemo\patientApexDemo.
js
Changed  patientApexDemo  LightningComponentBundle  force-app\main\default\lwc\patientApexDemo\patientApexDemo.
js-meta.xml
```

10. Imperative Apex Calls

- Added **imperative calls** in LWC for button-triggered actions.
- Example: Fetch patients on button click instead of auto-loading.



```javascript
import { LightningElement, track } from 'lwc';
import getPatients from '@salesforce/apex/PatientService.getPatients';

export default class PatientApexDemo extends LightningElement {
    @track patients;

    loadPatients() {
        getPatients()
            .then(result => {
                this.patients = result;
            })
            .catch(error => {
                console.error('Error loading patients: ', error);
            });
    }
}
```

```
=== Deployed Source
STATE    FULL NAME       TYPE                       PROJECT
PATH
───────  ──────────────  ─────────────────────────

───────────────────────────────────────────────────────────────────────────
Changed  patientApexDemo  LightningComponentBundle  force-app\main\default\lwc\patientApexDemo\patientApexDemo.
html
Changed  patientApexDemo  LightningComponentBundle  force-app\main\default\lwc\patientApexDemo\patientApexDemo.
js
Changed  patientApexDemo  LightningComponentBundle  force-app\main\default\lwc\patientApexDemo\patientApexDemo.
js-meta.xml
```

11. Navigation Service

- Used **NavigationMixin** in LWC to navigate to record pages.
- Example: Clicking on a Patient in patientList navigates to their detailed record page.