Project Title: Smart Pharmacy Inventory Tracker

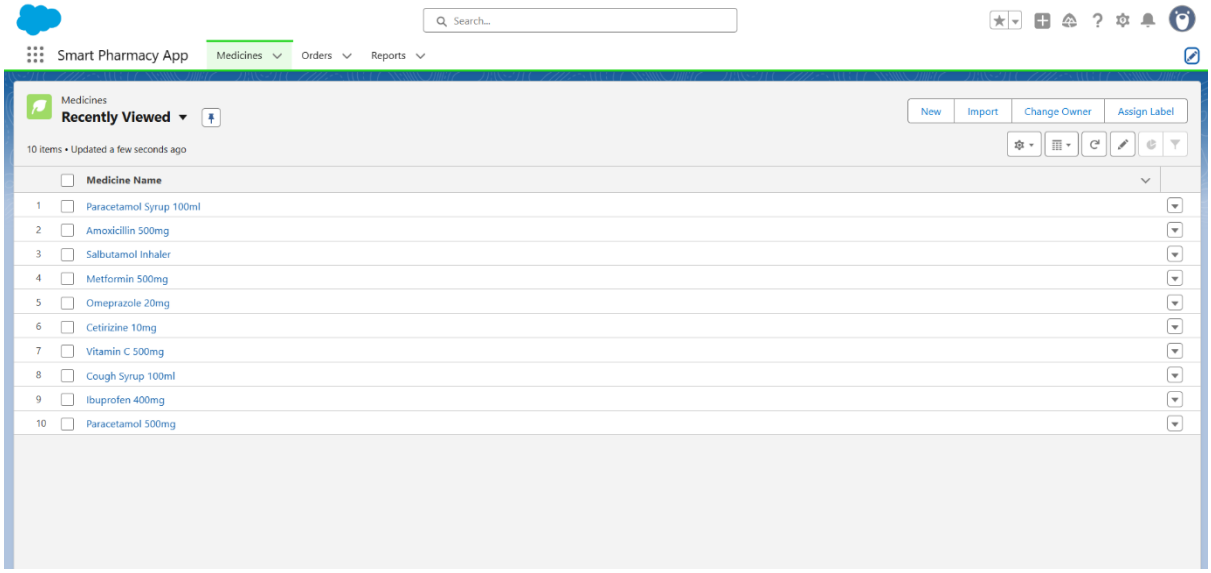
PHASE 6: User Interface Development

Executive Summary:

Phase 6 focused on delivering an intuitive and efficient user interface (UI) for the Smart Pharmacy Inventory Tracker. The goal was to make it simple for pharmacists, managers, and suppliers to interact with the system. A custom Smart Pharmacy App was created, dynamic record and home pages were designed to surface key information at a glance, and custom Lightning Web Components (LWCs) were developed for medicine stock monitoring and quick reorder actions.

Lightning App Builder:

- Purpose/Rationale**:** A dedicated Lightning App was necessary to provide a streamlined workspace for pharmacy staff, free from distractions of standard Salesforce apps.
- Implementation**:**
    - Created the Smart Pharmacy App using Lightning App Builder.
    - Added navigation items: Medicines, Suppliers, Orders, Reorder Requests, Reports, Dashboards.
    - Applied pharmacy-themed branding (pill bottle icon, blue/green color).

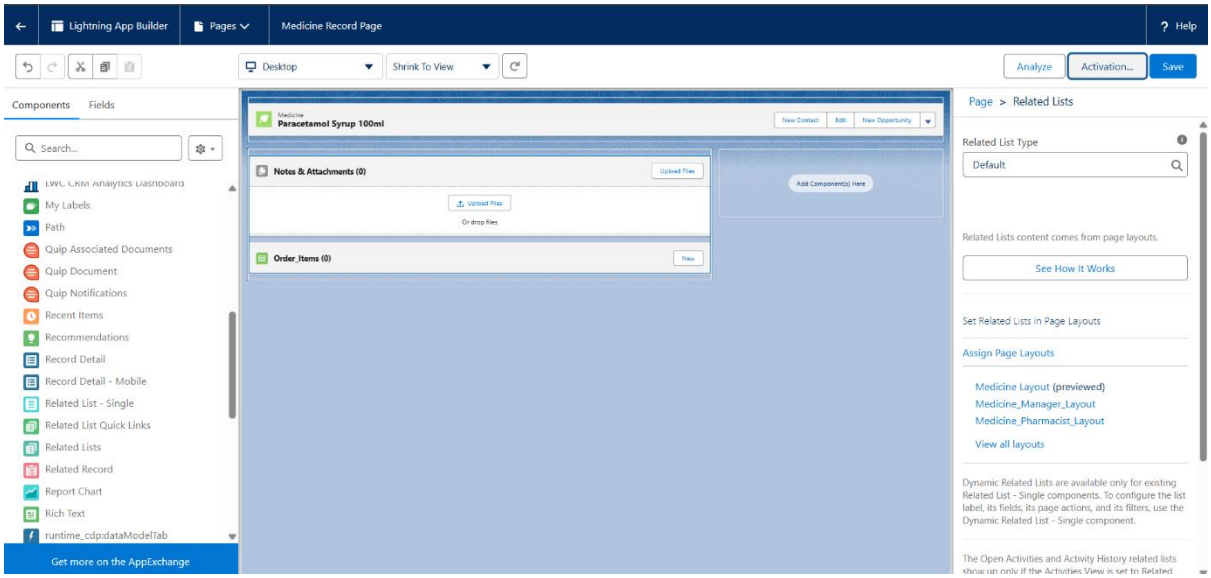


Record Pages:

 Purpose/Rationale: Default record pages are generic. A custom Medicine__c Record Page was built to highlight key fields.

 Implementation:

- Top Highlights Panel shows: Name, Quantity, Reorder Level, Expiry Date.
- Tabs component organizes Details, Related Orders, Reorder Requests.
- A custom LWC medicineDetail was placed in the right sidebar to allow pharmacists to quickly create reorder requests



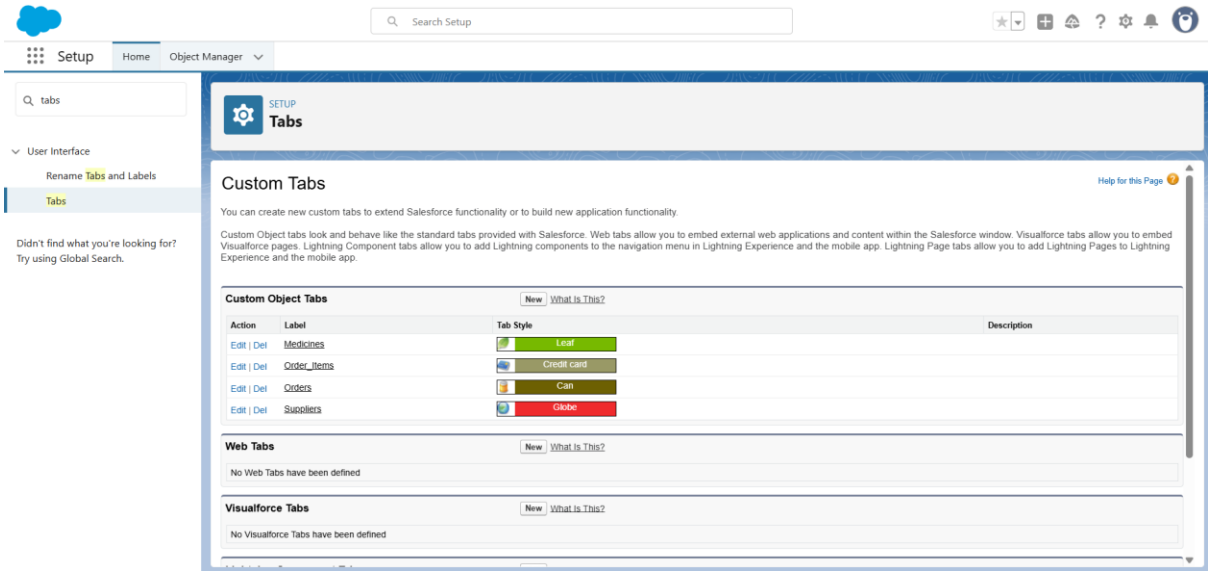
.

Tabs:

 Purpose/Rationale: Custom objects need dedicated tabs for navigation.

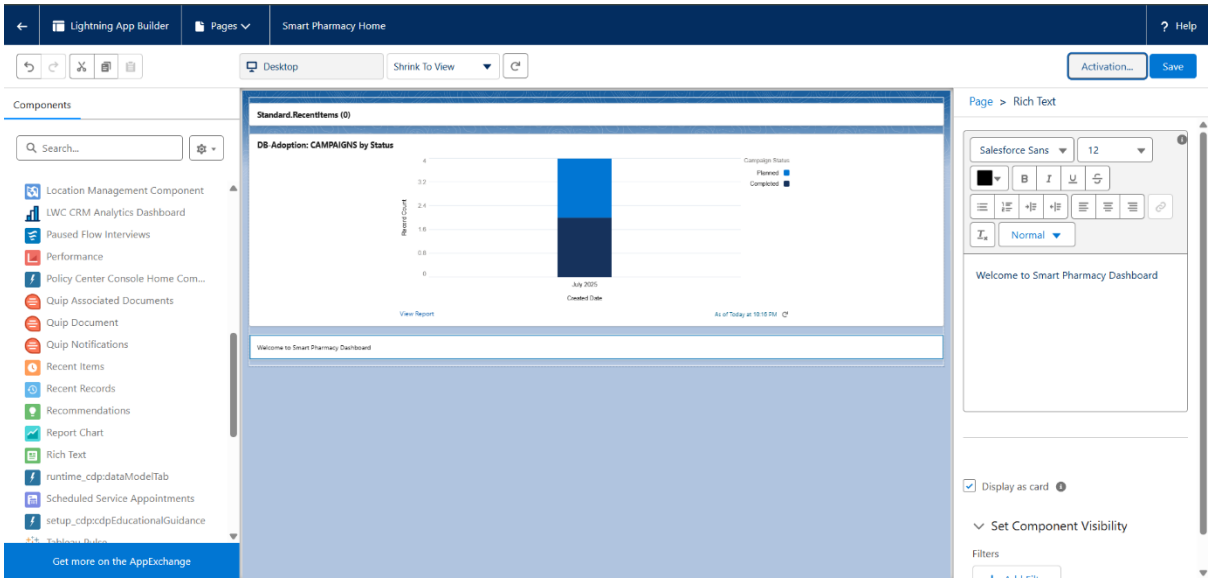
 Implementation: Created tabs for: Medicine, Supplier, Order, Reorder Request.

 Configured default list view for Medicines to show “Low Stock Medicines” by default.



Home Page Layouts:

- Purpose/Rationale: Home Page serves as the pharmacy dashboard.
- Implementation:
    - Created a custom Smart Pharmacy Home Page.
    - Added Dashboard components: Stock by Category, Expiring Medicines (Next 30 Days).
    - Embedded a List View showing medicines “Below Reorder Level”.
    - Added Rich Text for admin announcements (e.g., “Supplier audit scheduled this week”).

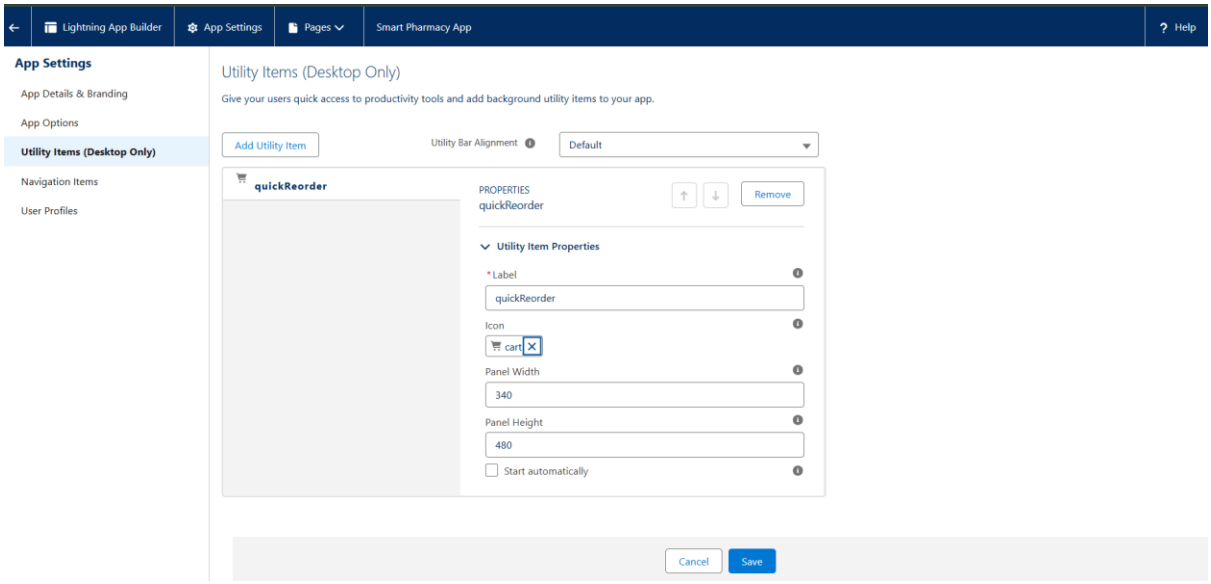


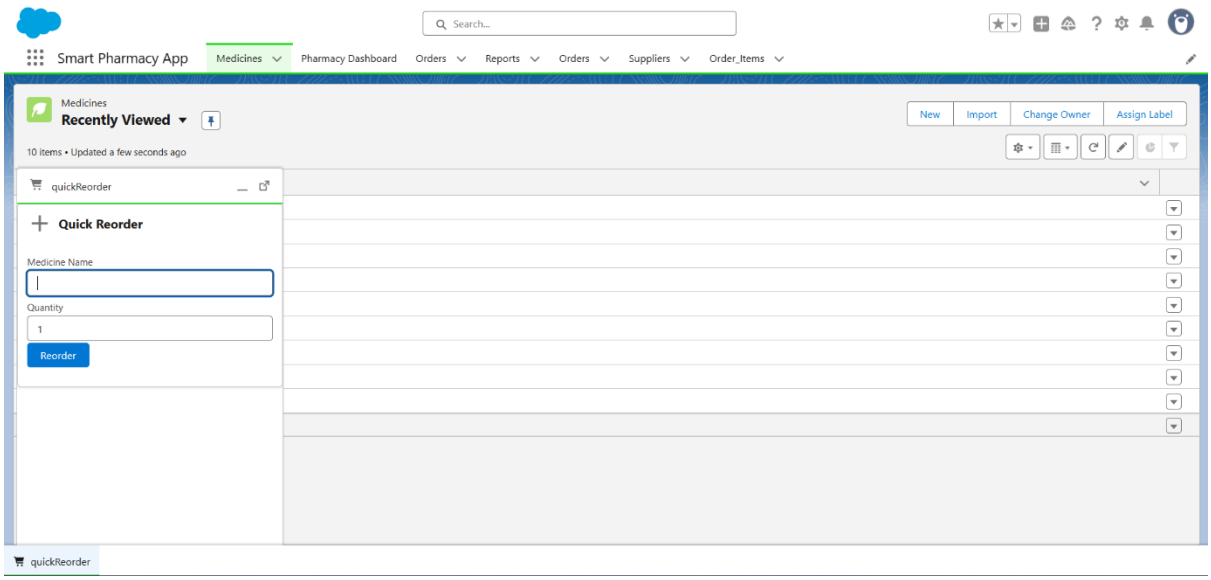
Utility Bar:

 Purpose/Rationale: Provide one-click access to quick tools across the app.

 Implementation:

- Added Recent Items (standard).
- Added custom LWC quickReorder (exposed for Utility Bar) to instantly raise a reorder request without leaving the current page.



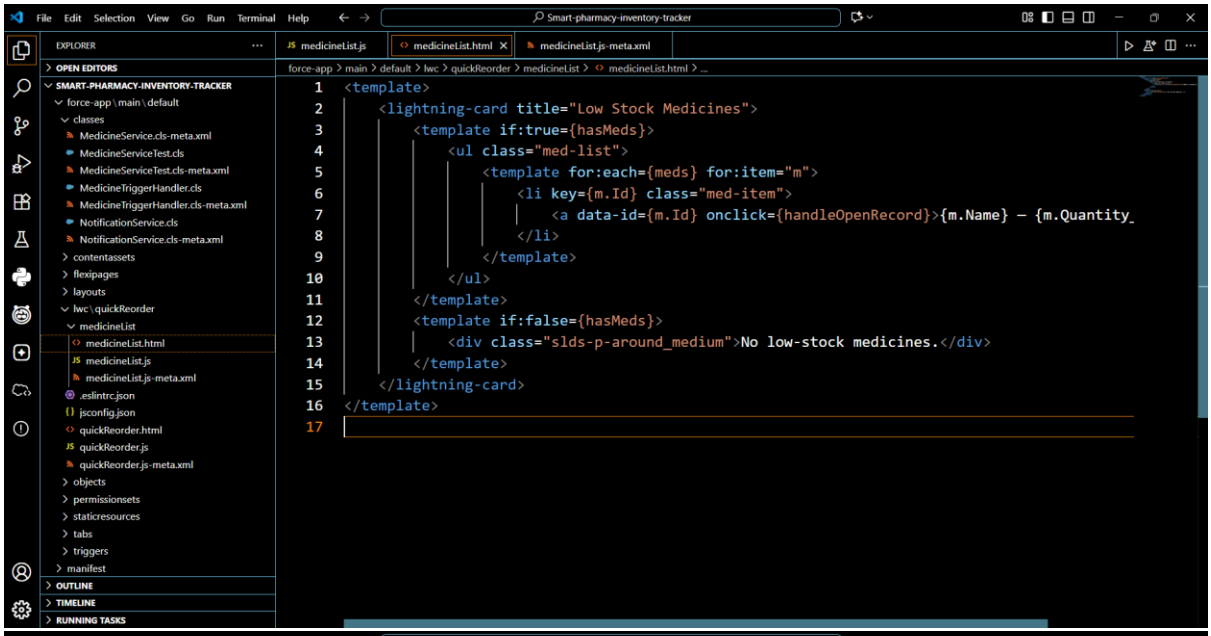


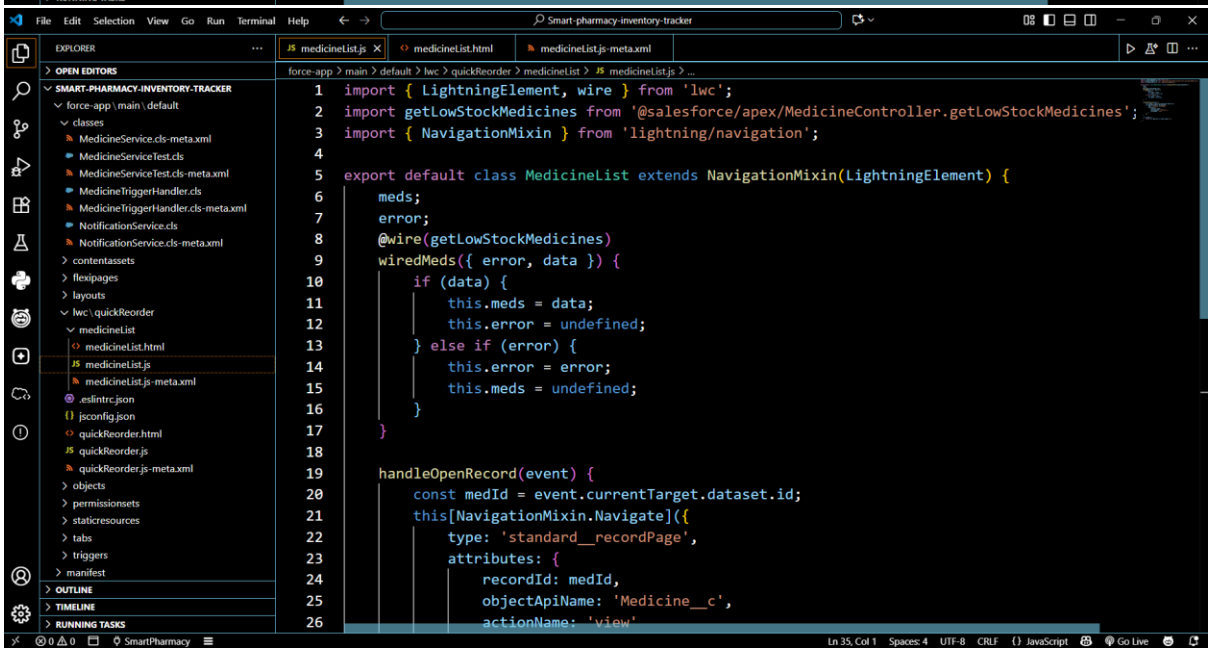
Lightning Web Components (LWC):

- Purpose/Rationale: LWCs provide fast, mobile-ready UI for custom needs.
- Implementation:
    - Built medicineList to display all low-stock medicines.
    - Built medicineDetail to show medicine details and allow reorder creation.
    - Used SLDS base components (lightning-card, lightning-input, lightning-button) for a professional and responsive design.

Example Code: medicineList.html

```
<template>
  <lightning-card title="Low Stock Medicines">
    <template if:true={meds}>
      <ul>
        <template for:each={meds} for:item="m">
          <li key={m.Id}>
            <a data-id={m.Id} onclick={handleOpenRecord}>{m.Name} — {m.Quantity__c}</a>
          </li>
        </template>
      </ul>
    </template>
    <template if:false={meds}>
      <p class="slds-p-around_medium">No low-stock medicines.</p>
    </template>
  </lightning-card>
</template>
```



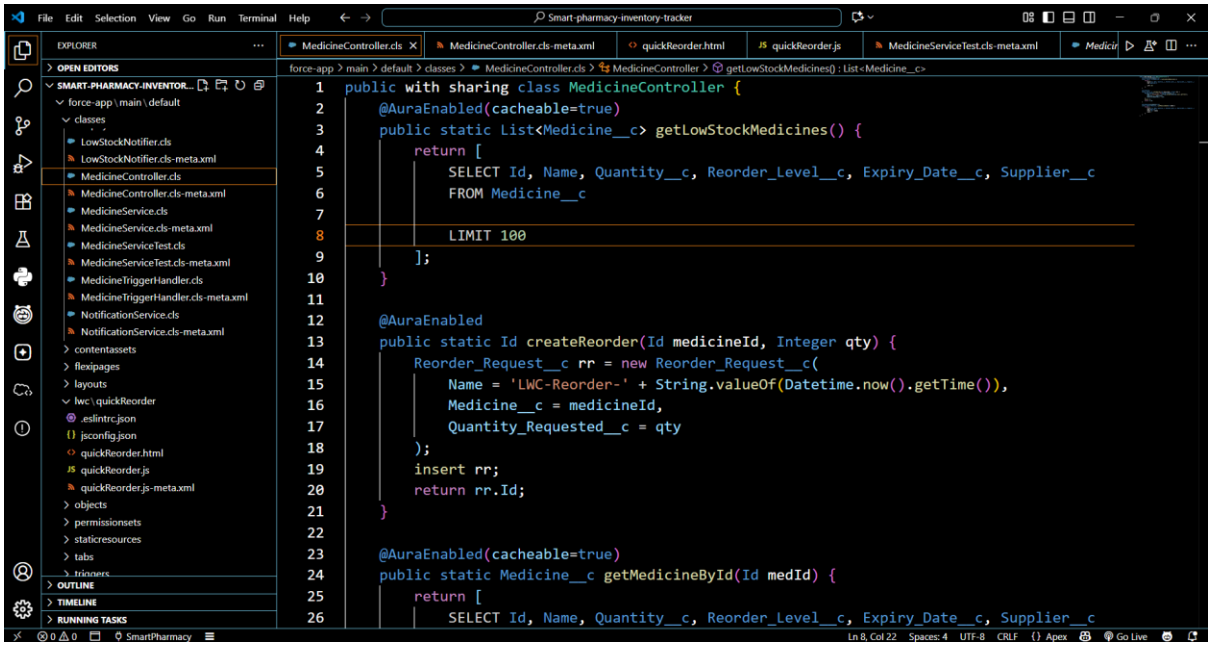


Apex with LWC:

 Purpose/Rationale: Apex was required to fetch medicine records and perform DML (creating reorder requests).

 Implementation:

- MedicineController.cls exposes methods getLowStockMedicines() and createReorder().
- @AuraEnabled annotation allows LWC to call these securely.
- LWC medicineDetail calls createReorder imperatively to insert a Reorder_Request__c record.



Events in LWC:

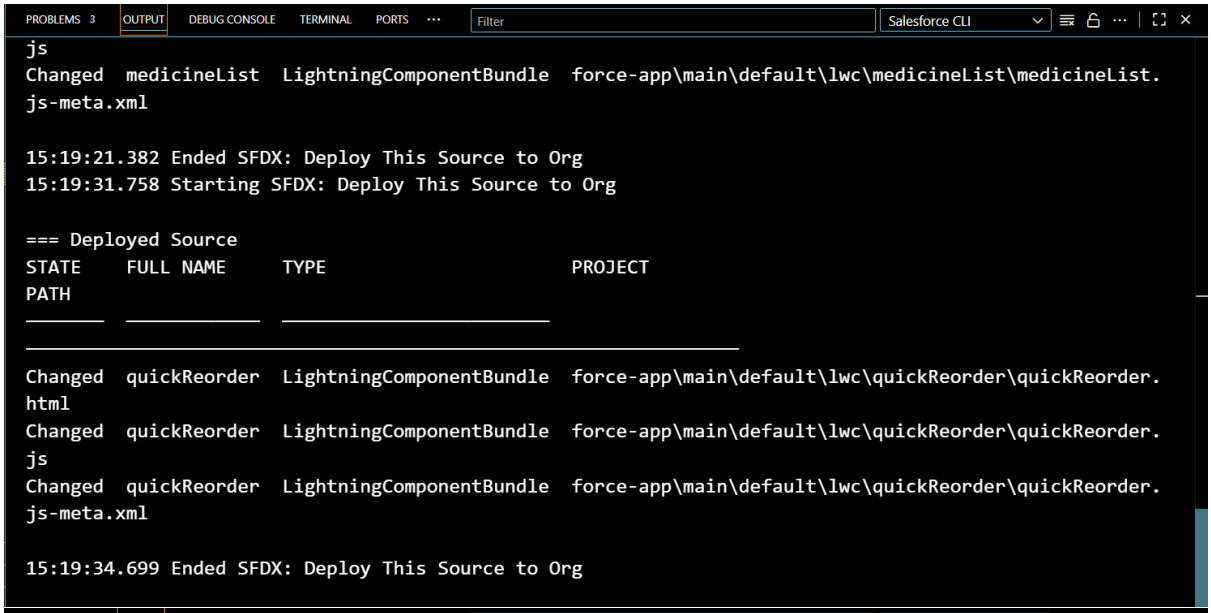
- Purpose/Rationale: Provide user feedback when actions succeed.
- Implementation:
    - On reorder success, a ShowToastEvent is dispatched with message “Reorder created successfully.”
    - A custom event reordered is emitted from medicineDetail to refresh the parent list component.

Deploying LWCs and Apex Controller to Org:

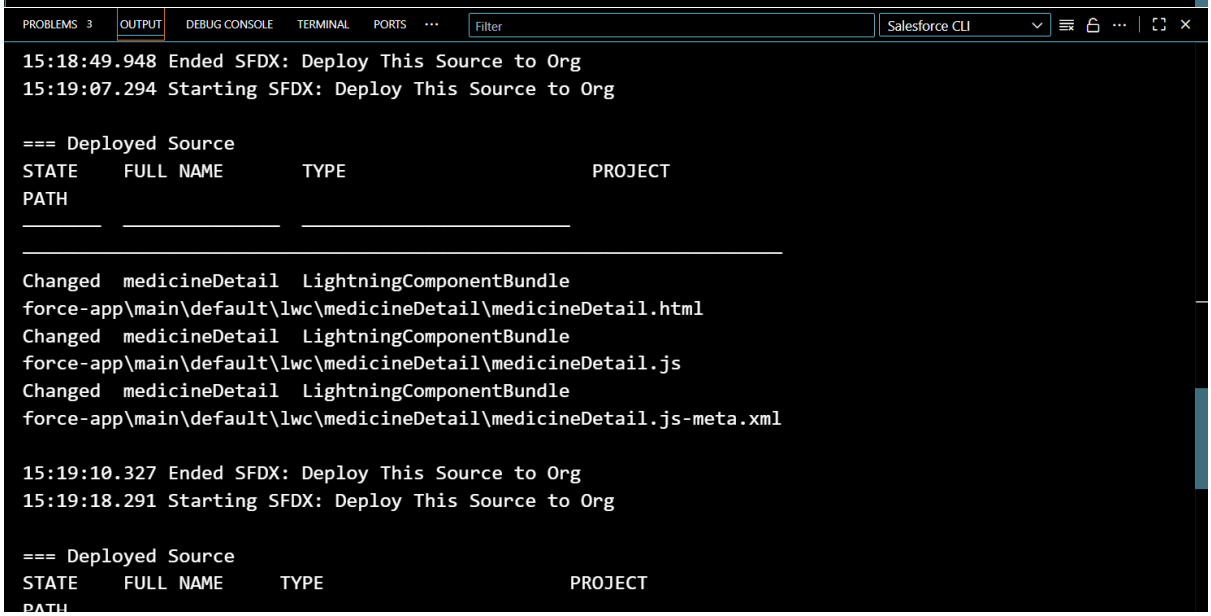
 Purpose/Rationale: After developing LWCs and Apex controllers locally in VS Code, they need to be deployed to the Salesforce org to be used by end users.

 Procedure:

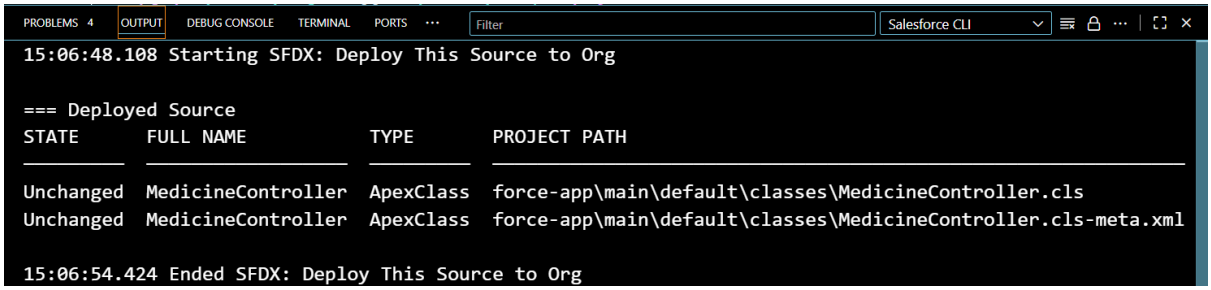
1. Open VS Code project connected to your Salesforce org.
2. Right-click on the LWC folder (e.g., medicineList or medicineDetail) → SFDX: Deploy Source to Org.
3. Repeat the same for Apex controller classes (MedicineController.cls).
4. Confirm successful deployment in the Output panel in VS Code.



```
PROBLEMS 3  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  ...  Filter  Salesforce CLI

js
Changed  medicineList  LightningComponentBundle  force-app\main\default\lwc\medicineList\medicineList.
js-meta.xml

15:19:21.382 Ended SFDX: Deploy This Source to Org
15:19:31.758 Starting SFDX: Deploy This Source to Org

=== Deployed Source
STATE    FULL NAME     TYPE                      PROJECT
PATH
_______  ____________  ________________________
_____________________________________________________________

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

15:19:34.699 Ended SFDX: Deploy This Source to Org
```



```
PROBLEMS 3  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  ...  Filter  Salesforce CLI

15:18:49.948 Ended SFDX: Deploy This Source to Org
15:19:07.294 Starting SFDX: Deploy This Source to Org

=== Deployed Source
STATE    FULL NAME       TYPE                      PROJECT
PATH
_______  _____________  ________________________
_____________________________________________________________

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

15:19:10.327 Ended SFDX: Deploy This Source to Org
15:19:18.291 Starting SFDX: Deploy This Source to Org

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



```
PROBLEMS 4  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  ...  Filter  Salesforce CLI

15:06:48.108 Starting SFDX: Deploy This Source to Org

=== Deployed Source
STATE      FULL NAME           TYPE       PROJECT PATH
_________  __________________  _________  ____________________________________________________________

Unchanged  MedicineController  ApexClass  force-app\main\default\classes\MedicineController.cls
Unchanged  MedicineController  ApexClass  force-app\main\default\classes\MedicineController.cls-meta.xml

15:06:54.424 Ended SFDX: Deploy This Source to Org
```

Wire Adapters, Imperative Calls, Events & Navigation Service Combined:

All these were combined in the medicineDetail LWC:

- @wire for fetching medicine details.
- Imperative Apex to create reorder.
- Events (ShowToastEvent, custom events) for feedback.
- Navigation Service to redirect to the reorder record.

Example Code: medicineDetail.js

```
import {LightningElement, api, wire} from 'lwc';
import getRecord} from 'lightning/uiRecordApi';
import createReorder from '@salesforce/apex/MedicineController.createReorder';
import {ShowToastEvent} from 'lightning/platformShowToastEvent';
import {NavigationMixin} from 'lightning/navigation';

import NAME_FIELD from '@salesforce/schema/Medicine__c.Name';

export default class MedicineDetail extends NavigationMixin(LightningElement) {
    @api recordId;
    @wire(getRecord, { recordId: '$recordId', fields: [NAME_FIELD] })
    medicine;

    handleCreateReorder() {
        createReorder({ medicineId: this.recordId, qty: 10 })
            .then(result => {
                this.dispatchEvent(new ShowToastEvent({
                    title: 'Success',
                    message: 'Reorder created successfully',
                    variant: 'success'
                }));
                this[NavigationMixin.Navigate]({
                    type: 'standard__recordPage',
```

```
                attributes: {
                    recordId: result,
                    objectApiName: 'Reorder_Request__c',
                    actionName: 'view'
                }
            });
        })
        .catch(error => {
            this.dispatchEvent(new ShowToastEvent({
                title: 'Error',
                message: error.body.message,
                variant: 'error'
            }));
        });
    }
}
```