Phase 6: User Interface Development – GreenHarvest

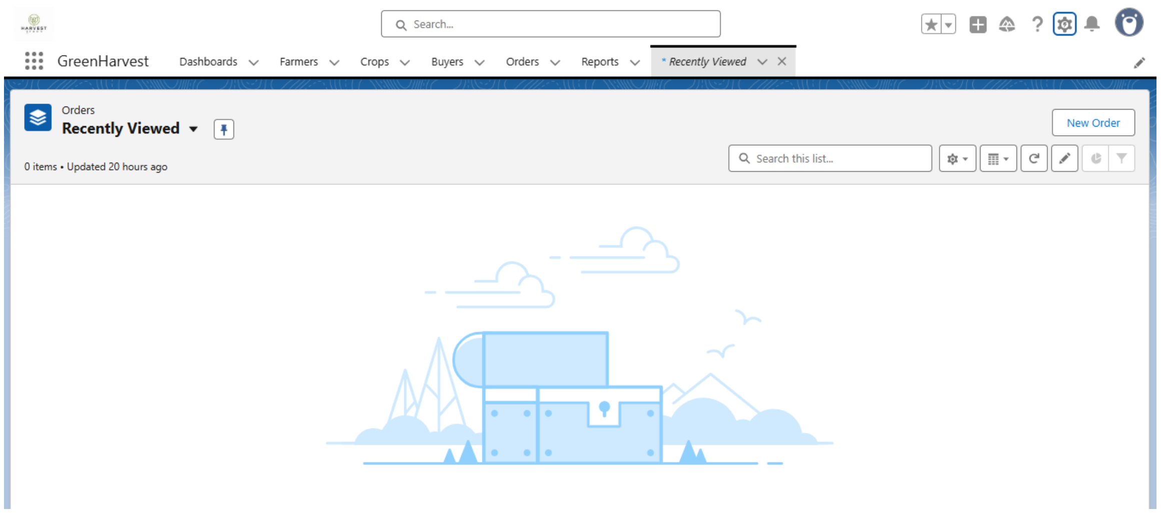
Goal

Design and implement a user-friendly Salesforce Lightning interface for *GreenHarvest*, providing role-based access for Farmers, Distributors, and Managers. The solution should enable seamless navigation and interaction with Orders, Crops, and Buyer details, ensuring efficiency, clarity, and a consistent experience across all user profiles..

1. Lightning App Builder – GreenHarvest App

- Created a Lightning App named GreenHarvest.
- Added navigation items: Farmer, Order, Crop, Buyer, Reports, Dashboards.
- Added Utility Bar items: Notes, Recent Items, Report Chart.

Steps:
1. Setup → App Manager → New Lightning App.
2. Enter App Name: GreenHarvest.
3. Added navigation items.
4. Added Utility Bar.
5. Assigned app to profiles.

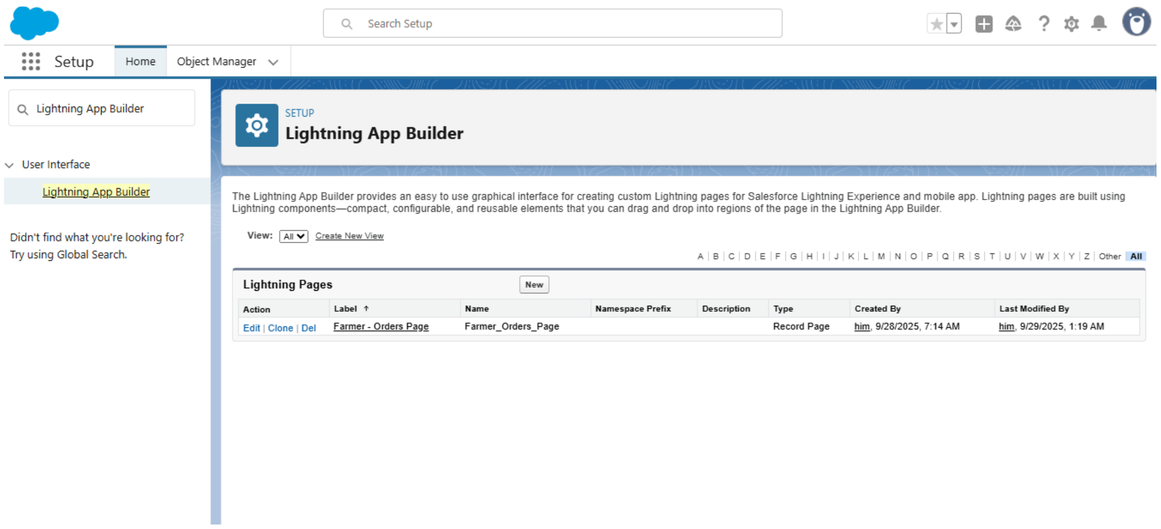


2. Record Pages – Farmer & Order Customization

- Farmer Record Page: Tabs (Details, Related Lists), Related List (Orders), Custom LWC → farmerOrdersList.
- Order Record Page: Highlights Panel, Record Detail, Custom LWC → orderWireExample.

Steps:

1. Setup → Lightning App Builder → New Record Page.
2. Selected Farmer__c and Order__c.
3. Dragged components into layout.
4. Added custom LWCs after deployment.
5. Activated pages for GreenHarvest app.



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

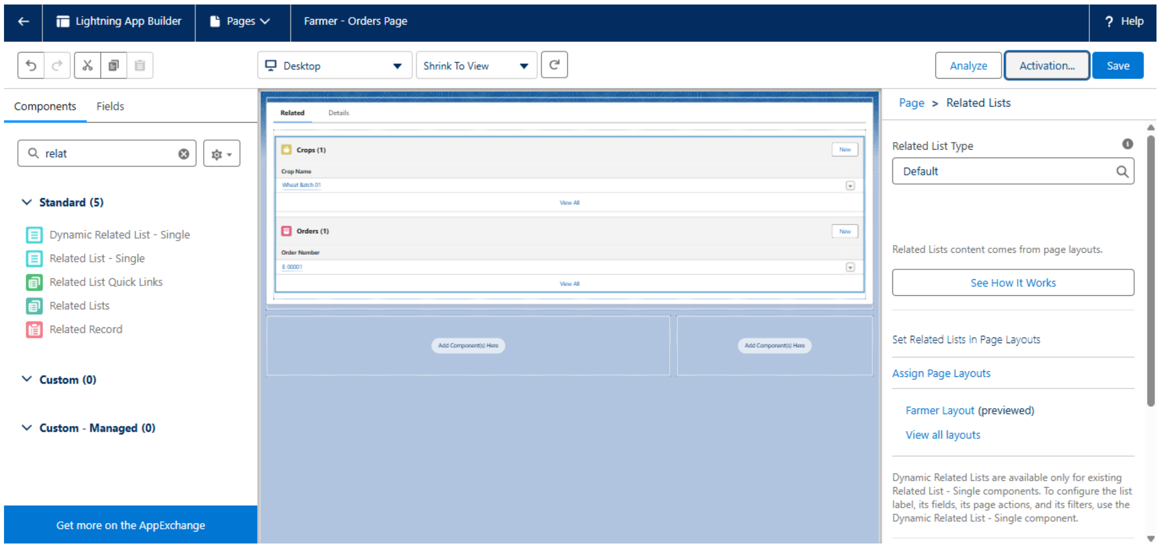
The Lightning App Builder provides an easy to use graphical interface for creating custom Lightning pages for Salesforce Lightning Experience and mobile app. Lightning pages are built using Lightning components—compact, configurable, and reusable elements that you can drag and drop into regions of the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

View: All | Create New View

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All

Lightning Pages [New]

Action	Label ↑	Name	Namespace Prefix	Description	Type	Created By	Last Modified By
Edit | Clone | Del	Farmer - Orders Page	Farmer_Orders_Page			Record Page	him, 9/28/2025, 7:14 AM	him, 9/29/2025, 1:19 AM

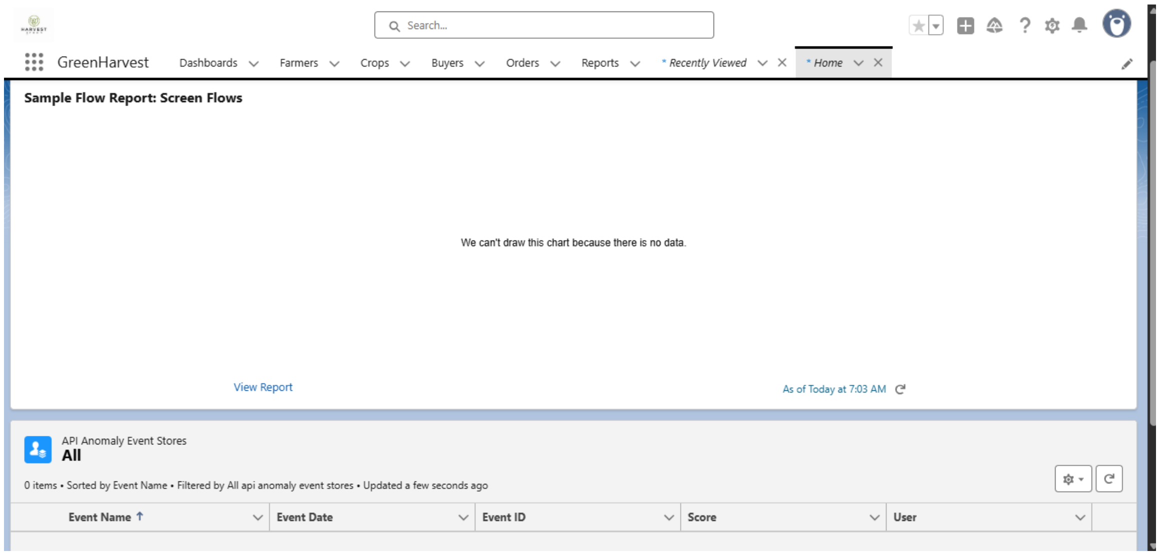


Related Lists content comes from page layouts.

Dynamic Related Lists are available only for existing Related List - Single components. To configure the list label, its fields, its page actions, and its filters, use the Dynamic Related List - Single component.

3. Home Page Layouts

- Created GreenHarvest Home Page.
- Added components: Report Chart (Orders by Status), Recent Items, List View (Pending Orders).



Sample Flow Report: Screen Flows

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

View Report

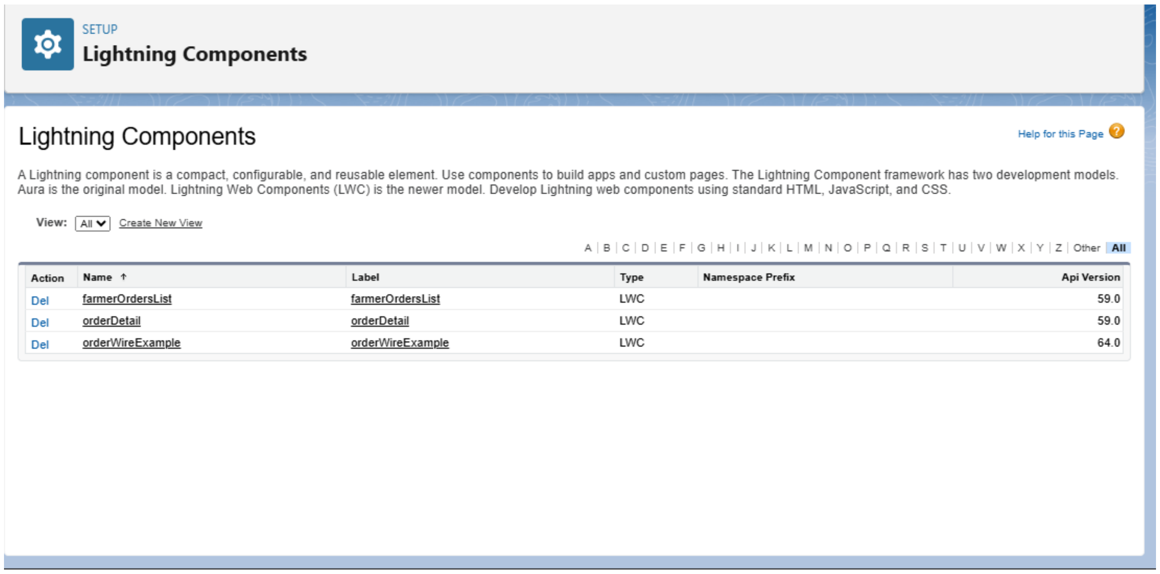
As of Today at 7:03 AM

API Anomaly Event Stores
All

0 items • Sorted by Event Name • Filtered by All api anomaly event stores • Updated a few seconds ago

Event Name	Event Date	Event ID	Score	User

4. Lightning Web Components (LWCs)


SETUP
Lightning Components

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.

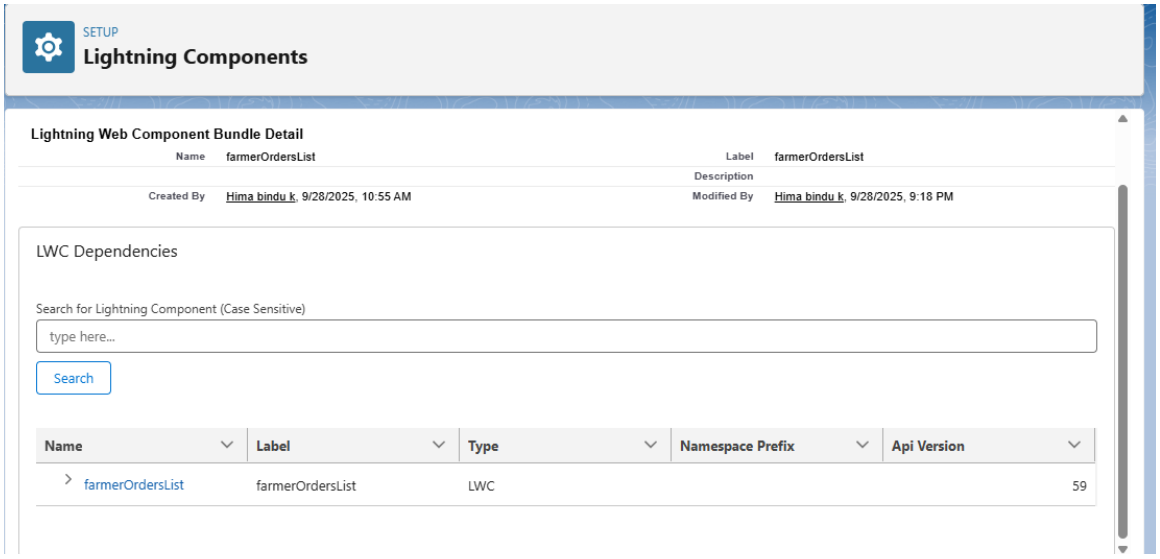
View: All | Create New View

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All

Action	Name ↑	Label	Type	Namespace Prefix	Api Version
Del	farmerOrdersList	farmerOrdersList	LWC		59.0
Del	orderDetail	orderDetail	LWC		59.0
Del	orderWireExample	orderWireExample	LWC		64.0

(a)farmerOrdersList

Displays orders linked to a Farmer.
- **Functionality:** Presents a consolidated view of all Orders related to a Farmer.
- **Placement:** Configured on the Farmer Record Page.
- **Value Add:** Simplifies order tracking for Farmers and Managers by displaying order data in a single, easy-to-access interface.


SETUP

Lightning Components

Lightning Web Component Bundle Detail

Name	farmerOrdersList	Label	farmerOrdersList
		Description	
Created By	Hima bindu k, 9/28/2025, 10:55 AM	Modified By	Hima bindu k, 9/28/2025, 9:18 PM

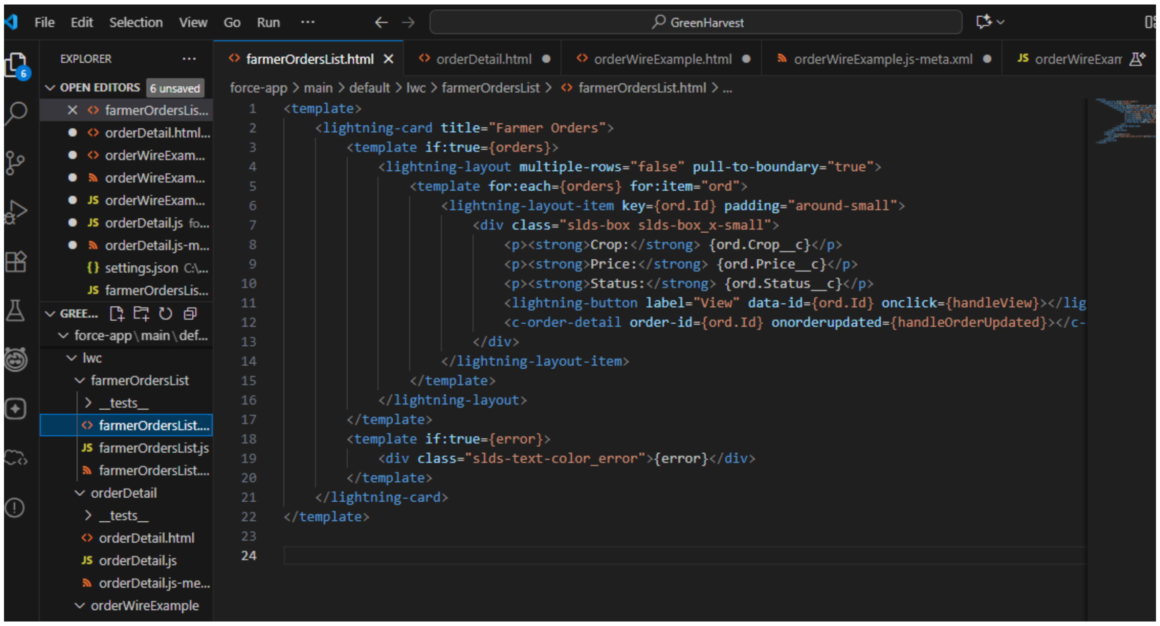
LWC Dependencies

Search for Lightning Component (Case Sensitive)

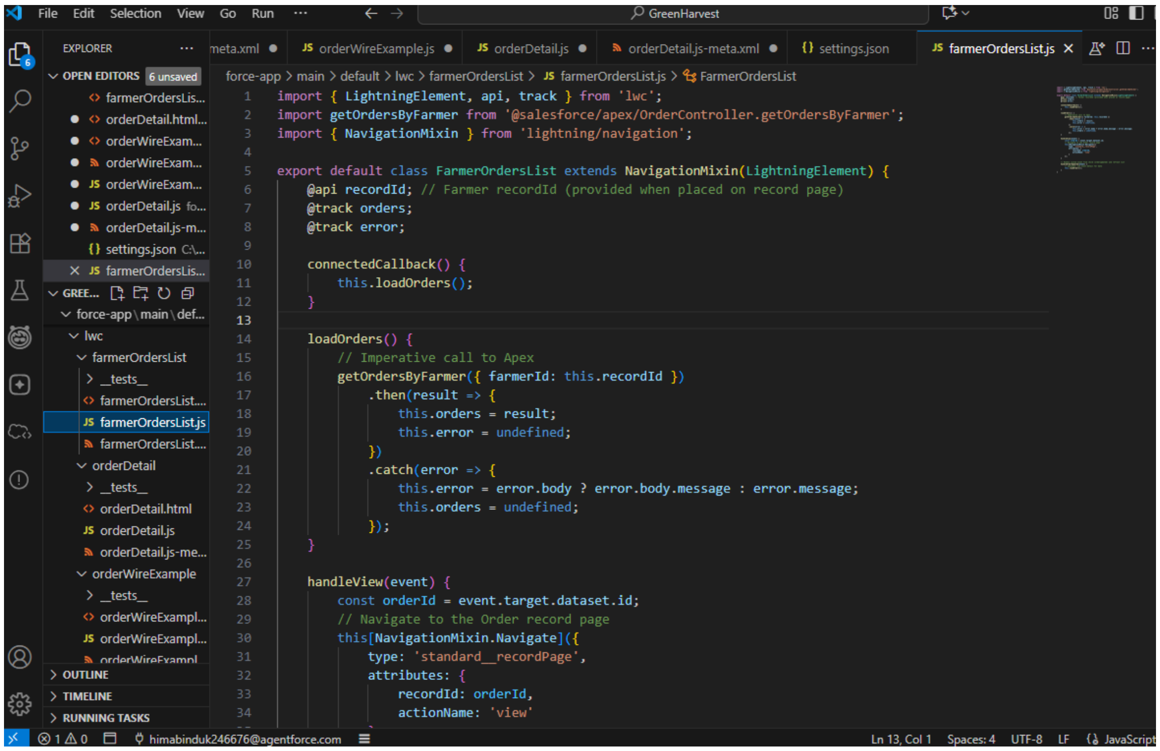
type here...

Search

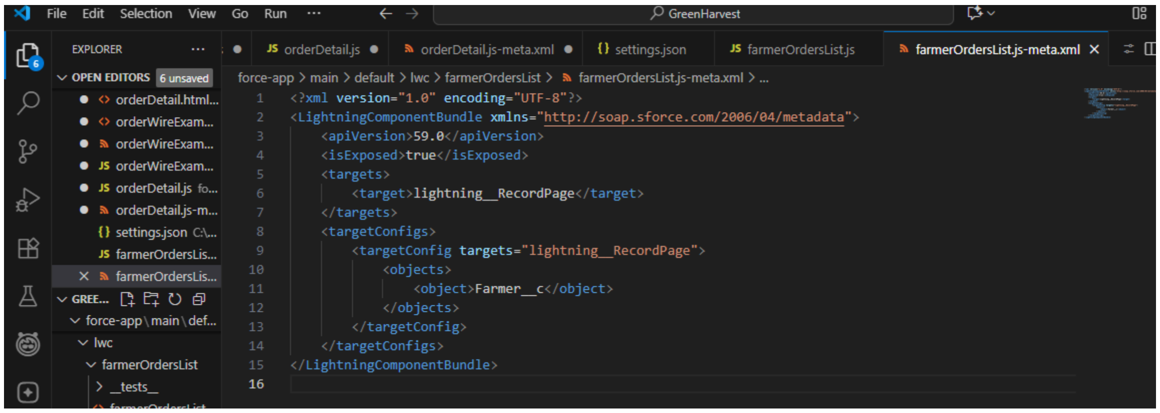
Name	Label	Type	Namespace Prefix	Api Version
farmerOrdersList	farmerOrdersList	LWC		59



```html
<template>
    <lightning-card title="Farmer Orders">
        <template if:true={orders}>
            <lightning-layout multiple-rows="false" pull-to-boundary="true">
                <template for:each={orders} for:item="ord">
                    <lightning-layout-item key={ord.Id} padding="around-small">
                        <div class="slds-box slds-box_x-small">
                            <p><strong>Crop:</strong> {ord.Crop__c}</p>
                            <p><strong>Price:</strong> {ord.Price__c}</p>
                            <p><strong>Status:</strong> {ord.Status__c}</p>
                            <lightning-button label="View" data-id={ord.Id} onclick={handleView}></lig
                            <c-order-detail order-id={ord.Id} onorderupdated={handleOrderUpdated}></c-
                        </div>
                    </lightning-layout-item>
                </template>
            </lightning-layout>
        </template>
        <template if:true={error}>
            <div class="slds-text-color_error">{error}</div>
        </template>
    </lightning-card>
</template>

```



```javascript
import { LightningElement, api, track } from 'lwc';
import getOrdersByFarmer from '@salesforce/apex/OrderController.getOrdersByFarmer';
import { NavigationMixin } from 'lightning/navigation';

export default class FarmerOrdersList extends NavigationMixin(LightningElement) {
    @api recordId; // Farmer recordId (provided when placed on record page)
    @track orders;
    @track error;

    connectedCallback() {
        this.loadOrders();
    }

    loadOrders() {
        // Imperative call to Apex
        getOrdersByFarmer({ farmerId: this.recordId })
            .then(result => {
                this.orders = result;
                this.error = undefined;
            })
            .catch(error => {
                this.error = error.body ? error.body.message : error.message;
                this.orders = undefined;
            });
    }

    handleView(event) {
        const orderId = event.target.dataset.id;
        // Navigate to the Order record page
        this[NavigationMixin.Navigate]({
            type: 'standard__recordPage',
            attributes: {
                recordId: orderId,
                actionName: 'view'
```

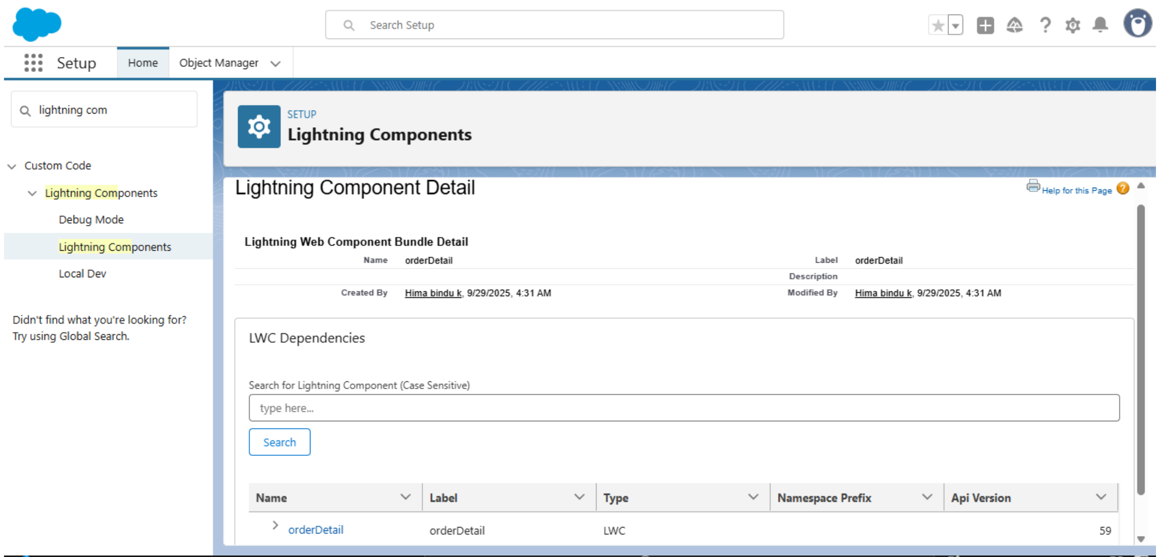


```xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
    </targets>
    <targetConfigs>
        <targetConfig targets="lightning__RecordPage">
            <objects>
                <object>Farmer__c</object>
            </objects>
        </targetConfig>
    </targetConfigs>
</LightningComponentBundle>
```

(b) orderDetail

Displays individual order detail with option to mark delivered.

- **Functionality:** Displays detailed information of a single Order with an action button to mark it as *Delivered*.
- **Placement:** Used within the `farmerOrdersList` component as a child element.
- **Value Add:** Empowers Distributors to instantly update order status while ensuring the parent list refreshes automatically through event-driven communication.



Lightning Component Detail

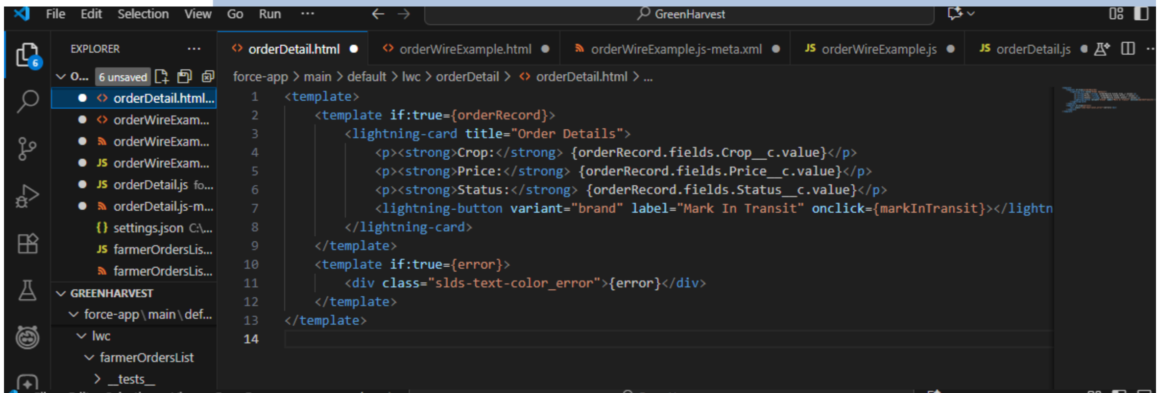
Lightning Web Component Bundle Detail

Name	orderDetail	Label	orderDetail
		Description	
Created By	Hima bindu k, 9/29/2025, 4:31 AM	Modified By	Hima bindu k, 9/29/2025, 4:31 AM

LWC Dependencies

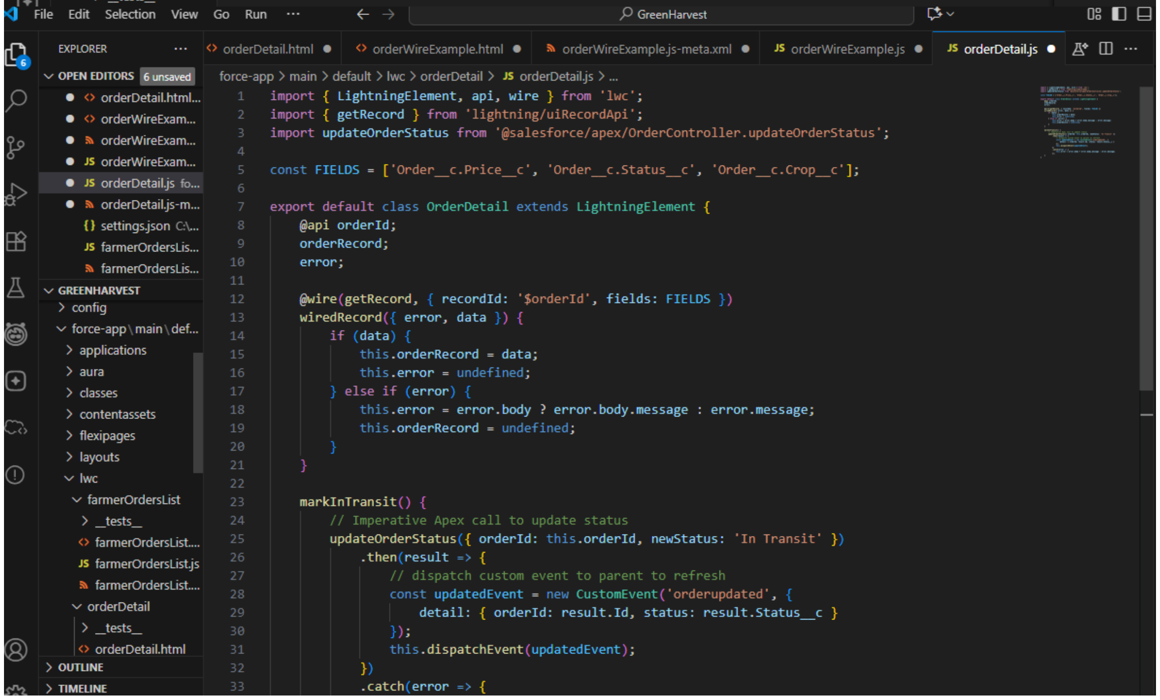
Search for Lightning Component (Case Sensitive)

Name	Label	Type	Namespace Prefix	Api Version
orderDetail	orderDetail	LWC		59



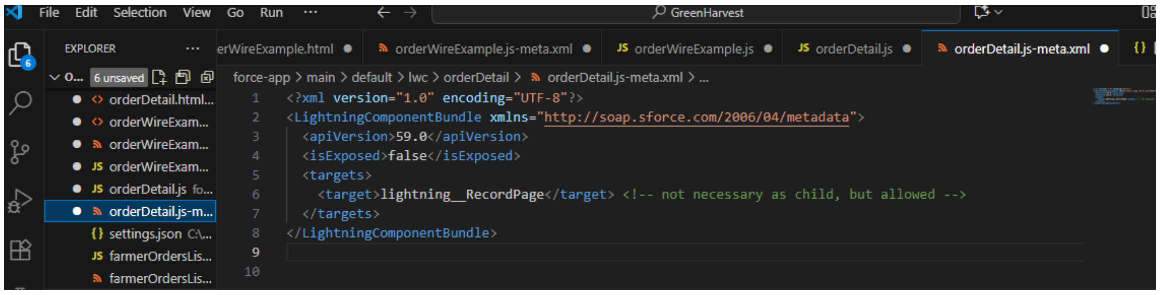
force-app > main > default > lwc > orderDetail > orderDetail.html > ...

```html
<template>
    <template if:true={orderRecord}>
        <lightning-card title="Order Details">
            <p><strong>Crop:</strong> {orderRecord.fields.Crop__c.value}</p>
            <p><strong>Price:</strong> {orderRecord.fields.Price__c.value}</p>
            <p><strong>Status:</strong> {orderRecord.fields.Status__c.value}</p>
            <lightning-button variant="brand" label="Mark In Transit" onclick={markInTransit}></lightn
        </lightning-card>
    </template>
    <template if:true={error}>
        <div class="slds-text-color_error">{error}</div>
    </template>
</template>
```



force-app > main > default > lwc > orderDetail > orderDetail.js > ...

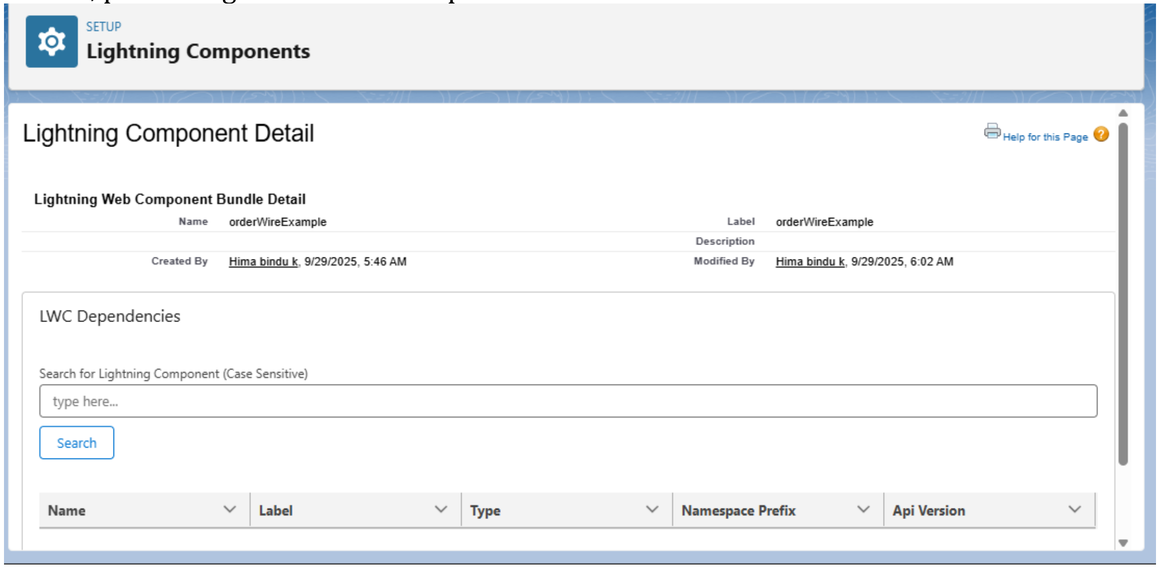
```javascript
import { LightningElement, api, wire } from 'lwc';
import { getRecord } from 'lightning/uiRecordApi';
import updateOrderStatus from '@salesforce/apex/OrderController.updateOrderStatus';

const FIELDS = ['Order__c.Price__c', 'Order__c.Status__c', 'Order__c.Crop__c'];

export default class OrderDetail extends LightningElement {
    @api orderId;
    orderRecord;
    error;

    @wire(getRecord, { recordId: '$orderId', fields: FIELDS })
    wiredRecord({ error, data }) {
        if (data) {
            this.orderRecord = data;
            this.error = undefined;
        } else if (error) {
            this.error = error.body ? error.body.message : error.message;
            this.orderRecord = undefined;
        }
    }

    markInTransit() {
        // Imperative Apex call to update status
        updateOrderStatus({ orderId: this.orderId, newStatus: 'In Transit' })
            .then(result => {
                // dispatch custom event to parent to refresh
                const updatedEvent = new CustomEvent('orderupdated', {
                    detail: { orderId: result.Id, status: result.Status__c }
                });
                this.dispatchEvent(updatedEvent);
            })
            .catch(error => {
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
  <apiVersion>59.0</apiVersion>
  <isExposed>false</isExposed>
  <targets>
    <target>lightning__RecordPage</target> <!-- not necessary as child, but allowed -->
  </targets>
</LightningComponentBundle>
```

(c) orderWireExample

Shows Order fields using Wire Adapter (uiRecordApi).
• **Functionality:** Retrieves and displays Order fields using the **Wire Adapter (uiRecordApi)**, eliminating the need for Apex.
• **Placement:** Added to the Order Record Page as a demonstration of declarative data fetching.
• **Value Add:** Serves as a best-practice example for leveraging standard Lightning Data Service, promoting low-code development and faster UI customization.



SETUP
Lightning Components

Lightning Component Detail

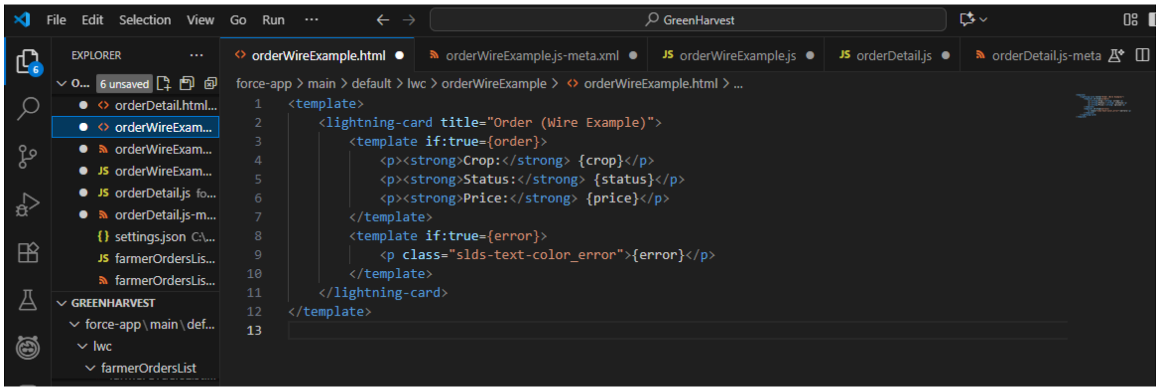
Lightning Web Component Bundle Detail

Name	orderWireExample	Label	orderWireExample
		Description	
Created By	Hima bindu k, 9/29/2025, 5:46 AM	Modified By	Hima bindu k, 9/29/2025, 6:02 AM

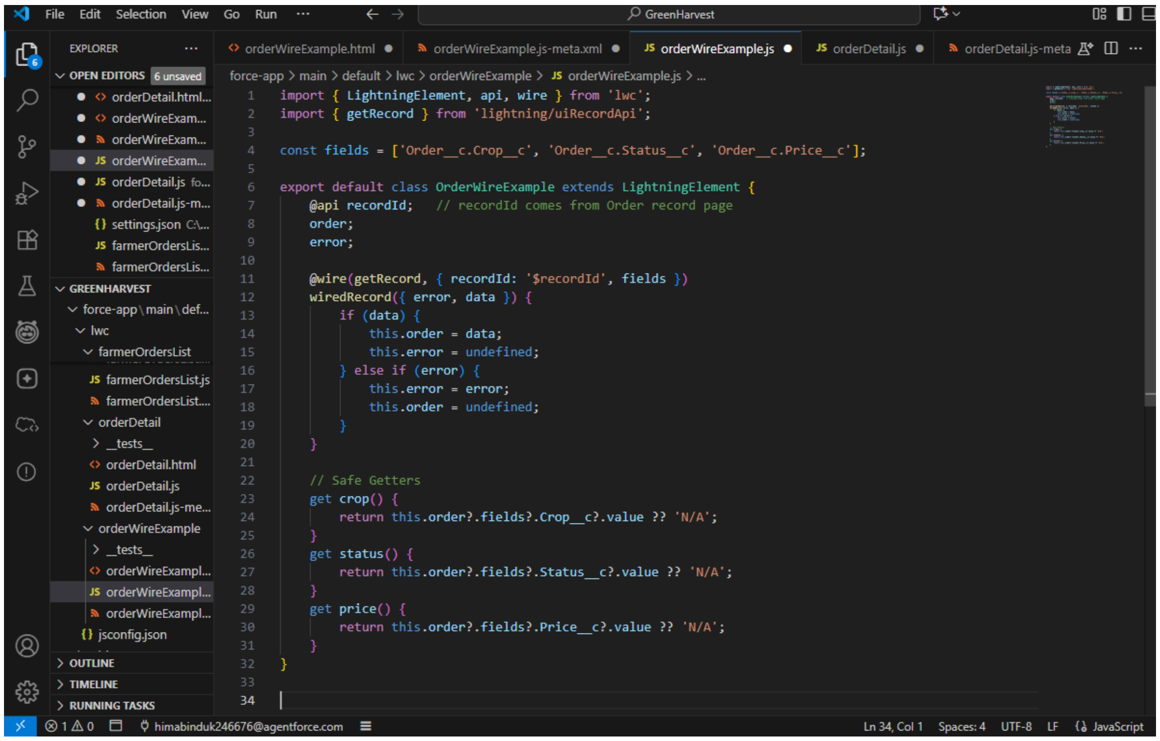
LWC Dependencies

Search for Lightning Component (Case Sensitive)

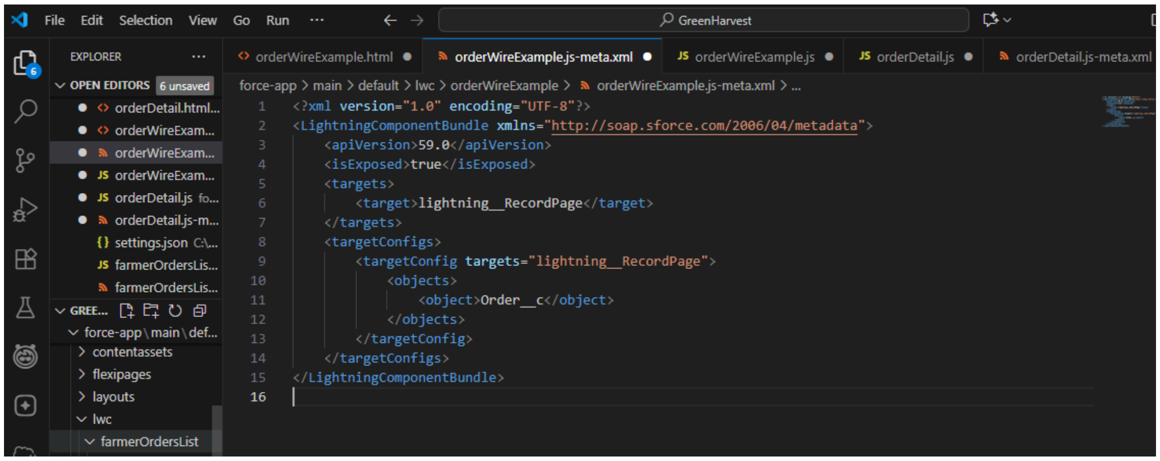
Name	Label	Type	Namespace Prefix	Api Version



```html
<template>
    <lightning-card title="Order (Wire Example)">
        <template if:true={order}>
            <p><strong>Crop:</strong> {crop}</p>
            <p><strong>Status:</strong> {status}</p>
            <p><strong>Price:</strong> {price}</p>
        </template>
        <template if:true={error}>
            <p class="slds-text-color_error">{error}</p>
        </template>
    </lightning-card>
</template>
```



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

const fields = ['Order__c.Crop__c', 'Order__c.Status__c', 'Order__c.Price__c'];

export default class OrderWireExample extends LightningElement {
    @api recordId;   // recordId comes from Order record page
    order;
    error;

    @wire(getRecord, { recordId: '$recordId', fields })
    wiredRecord({ error, data }) {
        if (data) {
            this.order = data;
            this.error = undefined;
        } else if (error) {
            this.error = error;
            this.order = undefined;
        }
    }

    // Safe Getters
    get crop() {
        return this.order?.fields?.Crop__c?.value ?? 'N/A';
    }
    get status() {
        return this.order?.fields?.Status__c?.value ?? 'N/A';
    }
    get price() {
        return this.order?.fields?.Price__c?.value ?? 'N/A';
    }
}

```



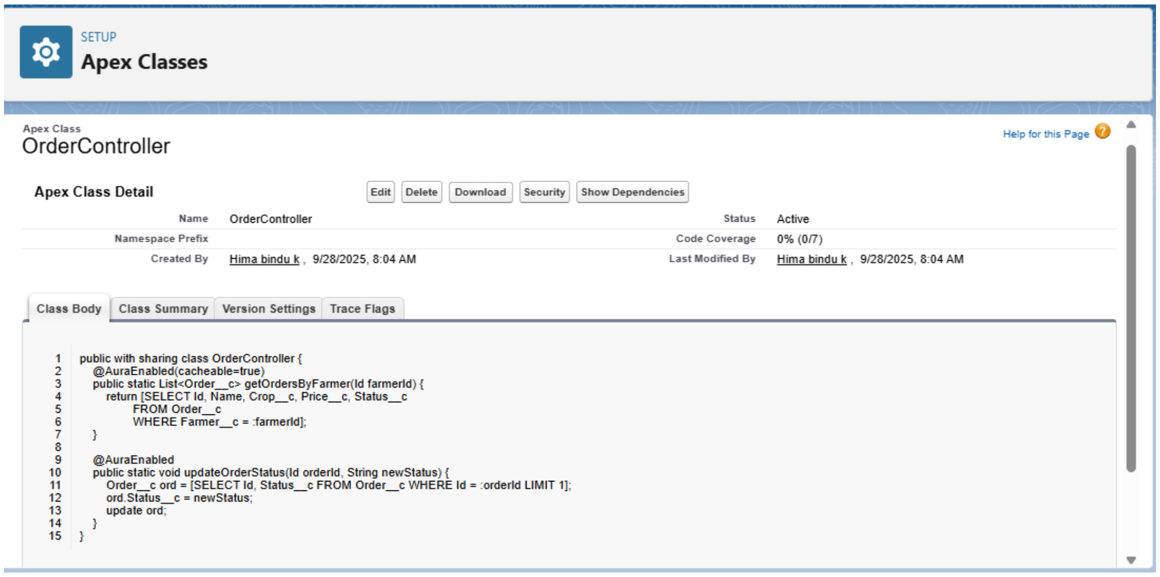
```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
    </targets>
    <targetConfigs>
        <targetConfig targets="lightning__RecordPage">
            <objects>
                <object>Order__c</object>
            </objects>
        </targetConfig>
    </targetConfigs>
</LightningComponentBundle>
```

5. Apex with LWC Integration

- Created Apex Class OrderController with methods:
  getOrdersByFarmer(Id farmerId) → returns Farmer's Orders.
  updateOrderStatus(Id orderId, String newStatus) → updates Order status.

Steps:
1. Setup → Developer Console → New Apex Class.
2. Added SOQL queries to fetch and update Orders.

3. Granted access via profiles.



Apex Class
OrderController

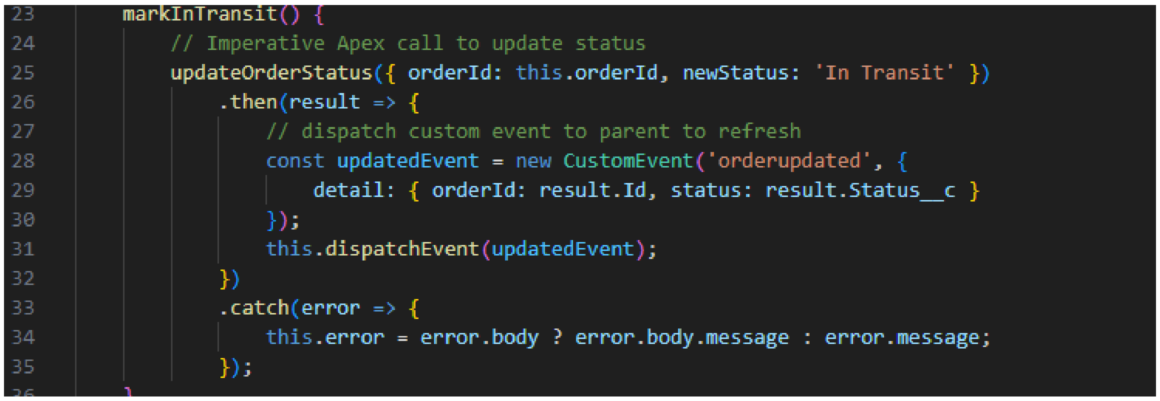
Apex Class Detail

Name	OrderController	Status	Active
Namespace Prefix		Code Coverage	0% (0/7)
Created By	Hima bindu k , 9/28/2025, 8:04 AM	Last Modified By	Hima bindu k , 9/28/2025, 8:04 AM

```
public with sharing class OrderController {
    @AuraEnabled(cacheable=true)
    public static List<Order__c> getOrdersByFarmer(Id farmerId) {
        return [SELECT Id, Name, Crop__c, Price__c, Status__c
                FROM Order__c
                WHERE Farmer__c = :farmerId];
    }

    @AuraEnabled
    public static void updateOrderStatus(Id orderId, String newStatus) {
        Order__c ord = [SELECT Id, Status__c FROM Order__c WHERE Id = :orderId LIMIT 1];
        ord.Status__c = newStatus;
        update ord;
    }
}
```

6. Events in LWC

- Child component (orderDetail) dispatches event: this.dispatchEvent(new CustomEvent('orderupdated')).
- Parent (farmerOrdersList) listens: <c-order-detail onorderupdated={handleOrderUpdated}></c-order-detail>.
- Parent handler refreshes Orders list automatically.



```
    markInTransit() {
        // Imperative Apex call to update status
        updateOrderStatus({ orderId: this.orderId, newStatus: 'In Transit' })
            .then(result => {
                // dispatch custom event to parent to refresh
                const updatedEvent = new CustomEvent('orderupdated', {
                    detail: { orderId: result.Id, status: result.Status__c }
                });
                this.dispatchEvent(updatedEvent);
            })
            .catch(error => {
                this.error = error.body ? error.body.message : error.message;
            });
```

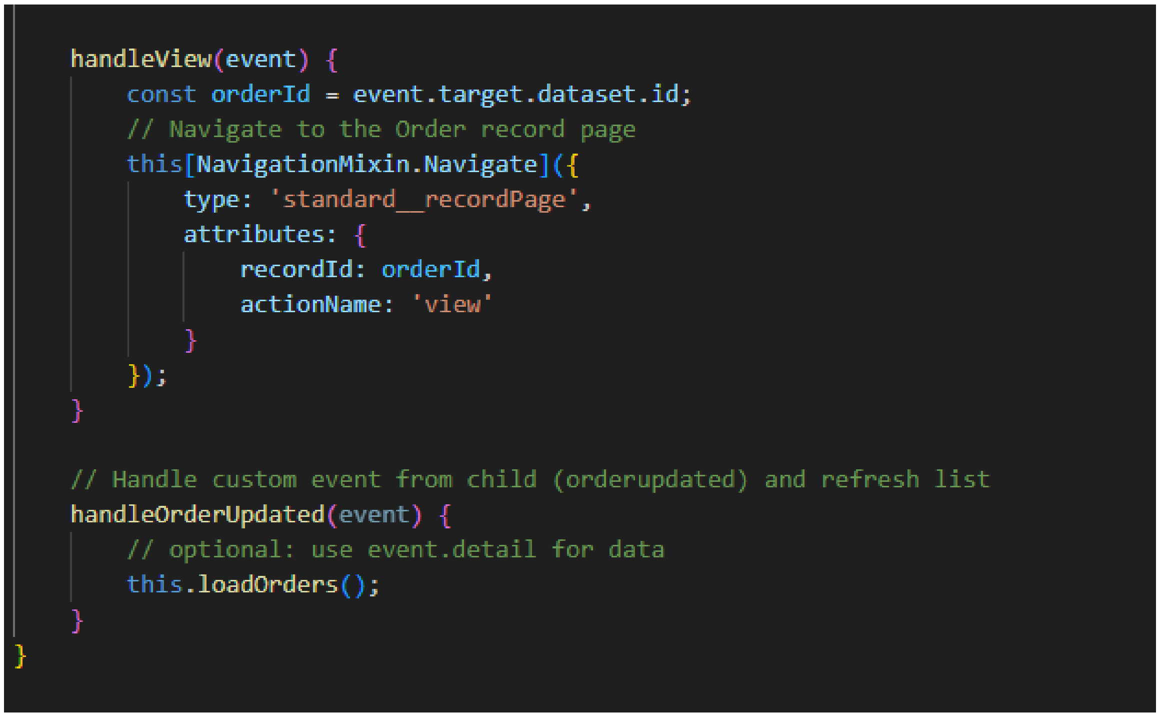
7. Wire Adapters & Imperative Calls

- Wire Adapter: orderWireExample uses @wire(getRecord) to fetch Crop, Status, Price.
- Imperative Call: farmerOrdersList calls Apex getOrdersByFarmer imperatively.

[Screenshot Placeholder: OrderWireExample Output]
[Screenshot Placeholder: Imperative Call Code in VS Code]

8. Navigation Service

- farmerOrdersList uses NavigationMixin.Navigate to view Order record page.



```
    handleView(event) {
        const orderId = event.target.dataset.id;
        // Navigate to the Order record page
        this[NavigationMixin.Navigate]({
            type: 'standard__recordPage',
            attributes: {
                recordId: orderId,
                actionName: 'view'
            }
        });
    }

    // Handle custom event from child (orderupdated) and refresh list
    handleOrderUpdated(event) {
        // optional: use event.detail for data
        this.loadOrders();
    }
}
```

The GreenHarvest Lightning interface now supports:
- App & Page Customization.
- LWC-based UI with Apex integration.
- Parent-Child Event handling.
- Wire Adapters, Imperative Calls, and Navigation Service.

This phase ensures an interactive, real-time experience for GreenHarvest users.