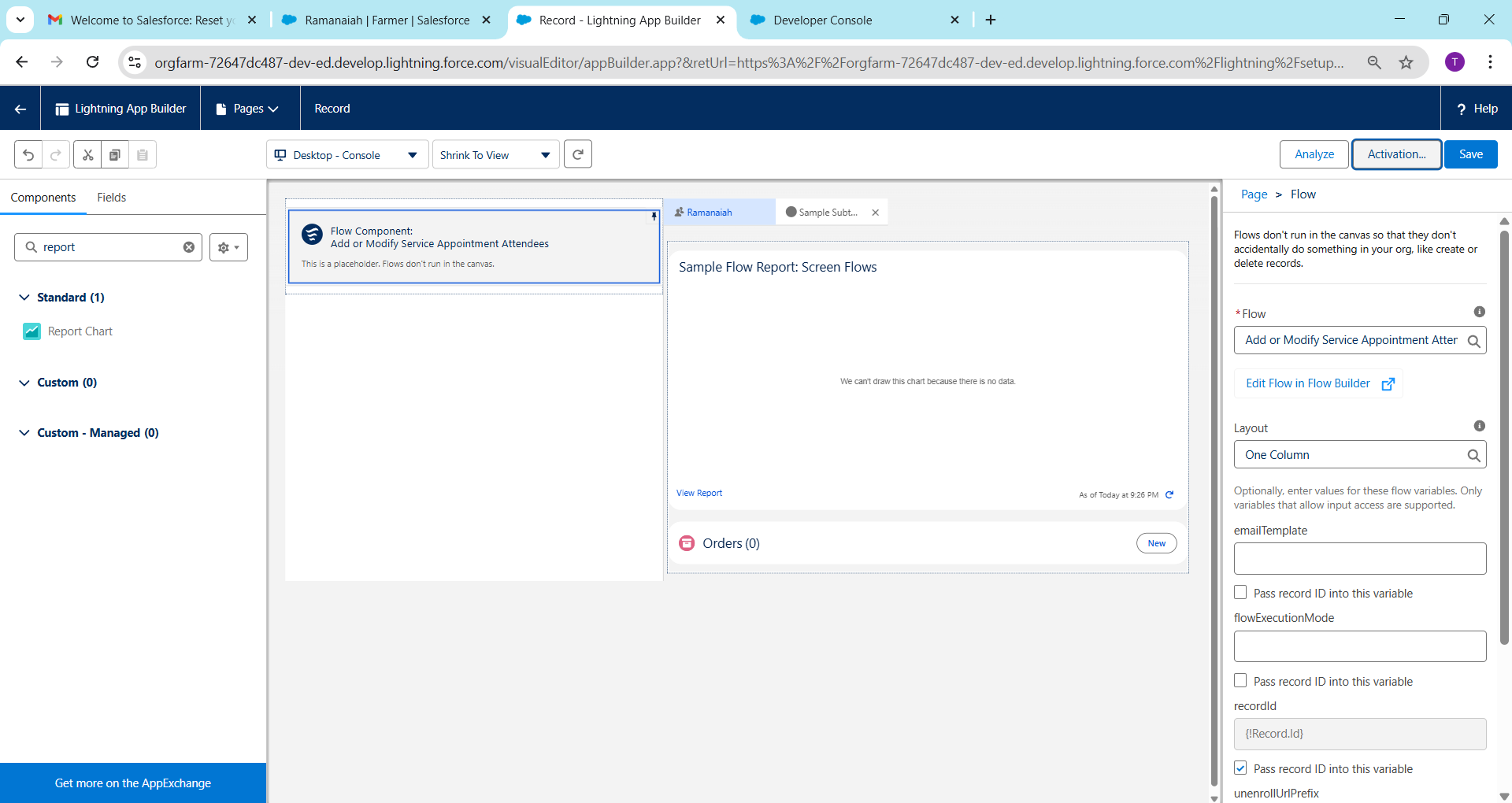
## Phase 6: User Interface (UI) Development in Salesforce

**1. Lightning App Builder**

* A **point-and-click tool** to create custom pages in Salesforce.
* You can design pages for **Apps, Record Pages, or Home Pages** without coding.
* **Components** like lists, charts, or custom LWC can be dragged onto the page.

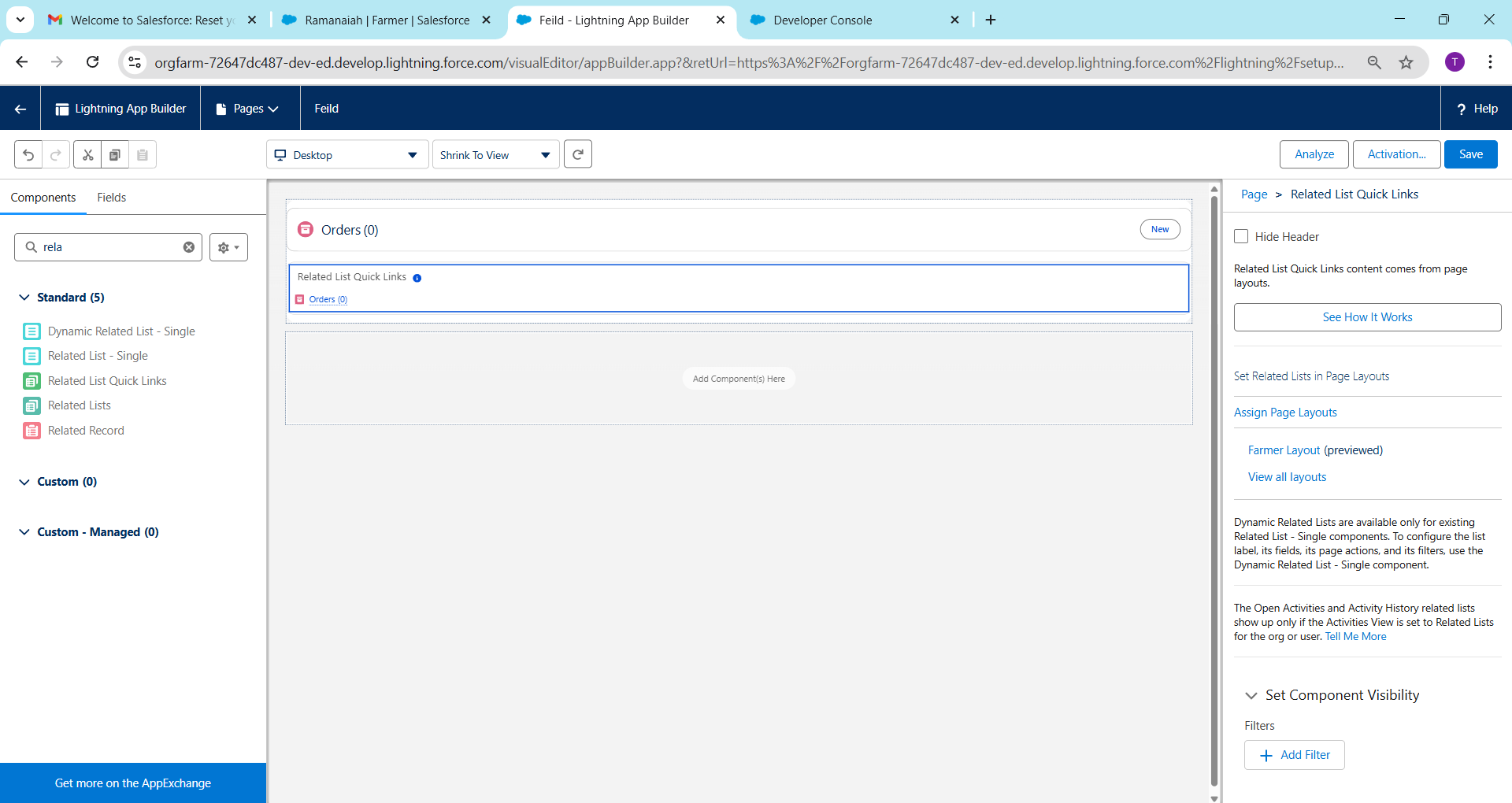
**Example:** For your Rural Market Linkage project, you could create a **dashboard page** showing farmers’ orders and buyer stats.



**2. Record Pages**

* A **record page** is the layout you see when opening a specific record (e.g., Farmer\_\_c or Order\_\_c).
* You can **customize what fields, related lists, and components appear**.
* Can be different for **desktop, mobile, or Lightning apps**.

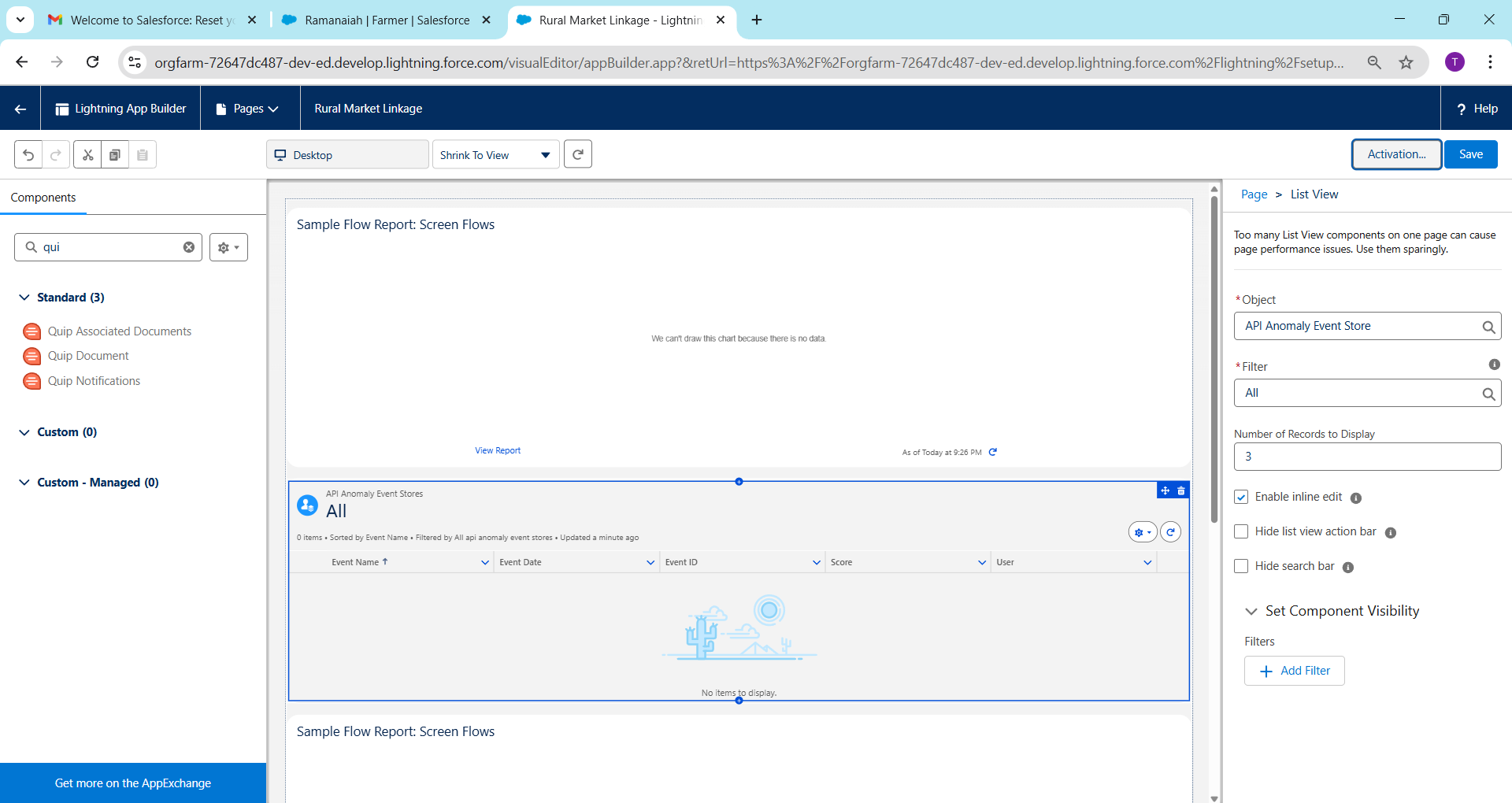
**Example:** A Farmer record page could show the farmer’s details, crops, past orders, and subsidy applications in one view.



**3. Tabs**

* Tabs are like **menu items** in Salesforce.
* Can display **Objects, Visualforce pages, Lightning pages, or Web URLs**.
* Helps users **navigate between apps or features**.

**Example:** Create tabs for **Farmers, Buyers, Orders, and Subsidy Applications** in your app.



**4. Home Page Layouts**

* The **homepage is the first page users see** when logging in.
* Can include **charts, reports, lists, tasks, news**, and **custom components**.
* Helps users **quickly access relevant info**.

**Example:** Your home page could show **latest orders, pending subsidies, and notifications for buyers or farmers**.

**5. Utility Bar**

* A **persistent bar at the bottom of the screen** for quick actions.
* Can include **custom components, calculators, chatbots, or quick links**.
* Makes common tasks accessible **anywhere in the app**.

**Example:** A utility bar could have **“Quick Order Entry”** or **“Farmer Lookup”** for your team.

**6. LWC (Lightning Web Components)**

* Modern **JavaScript-based components** for building Salesforce UI.
* Faster and more **standard-compliant** than older Aura components.
* Can be used on **record pages, home pages, or apps**.

**Example:** Create an LWC called **“FarmerOrderSummary”** to show recent orders with dynamic updates.

**7. Apex with LWC**

* **Apex** is Salesforce’s server-side code.
* LWCs are **frontend components** and can **call Apex** to get or update data.
* Enables **complex logic** behind simple UI.

**Example:** Your LWC can call an Apex method to **fetch total crops available per farmer**.

public with sharing class FarmerController {

@AuraEnabled(cacheable=true)

public static List<Farm\_\_c> getAllCrops() {

return [SELECT Name, Quantity\_\_c FROM Farm\_\_c];

}

}

**8. Events in LWC**

* LWCs communicate using **events**.
* **Parent-child communication:** Child component can send data to parent using **Custom Events**.
* **Sibling communication:** Often goes through a **common parent** or **pub-sub model**.

**Example:** When a farmer selects a crop in one LWC, another LWC showing **available buyers** updates automatically.

**9. Wire Adapters**

* **Reactive data binding** in LWC.
* Automatically fetches **data from Salesforce objects** or Apex.
* Uses @wire decorator to bind data to the component.

import { LightningElement, wire } from 'lwc';

import getAllCrops from '@salesforce/apex/FarmerController.getAllCrops';

export default class CropList extends LightningElement {

@wire(getAllCrops) crops;

}

* **Advantage:** No need to manually refresh data.

**10. Imperative Apex Calls**

* Unlike wire adapters, **imperative calls** are made **on-demand**, usually after a user action (click, form submit).
* Gives **more control over execution and error handling**.

handleClick() {

getAllCrops()

.then(result => { this.crops = result; })

.catch(error => { this.error = error; });

}

**Example:** Fetch crops **only when a user clicks “Show Available Crops”**.

**11. Navigation Service**

* Lets LWCs **navigate programmatically** to pages, records, or apps.
* Uses NavigationMixin in JavaScript.

import { NavigationMixin } from 'lightning/navigation';

export default class NavigateToRecord extends NavigationMixin(LightningElement) {

goToFarmerRecord() {

this[NavigationMixin.Navigate]({

type: 'standard\_\_recordPage',

attributes: {

recordId: '001xx000003DGSzAAO',

objectApiName: 'Farmer\_\_c',

actionName: 'view'

}

});

}

}

**Example:** Clicking a crop in your LWC navigates to the **Farmer record page** showing orders for that crop.