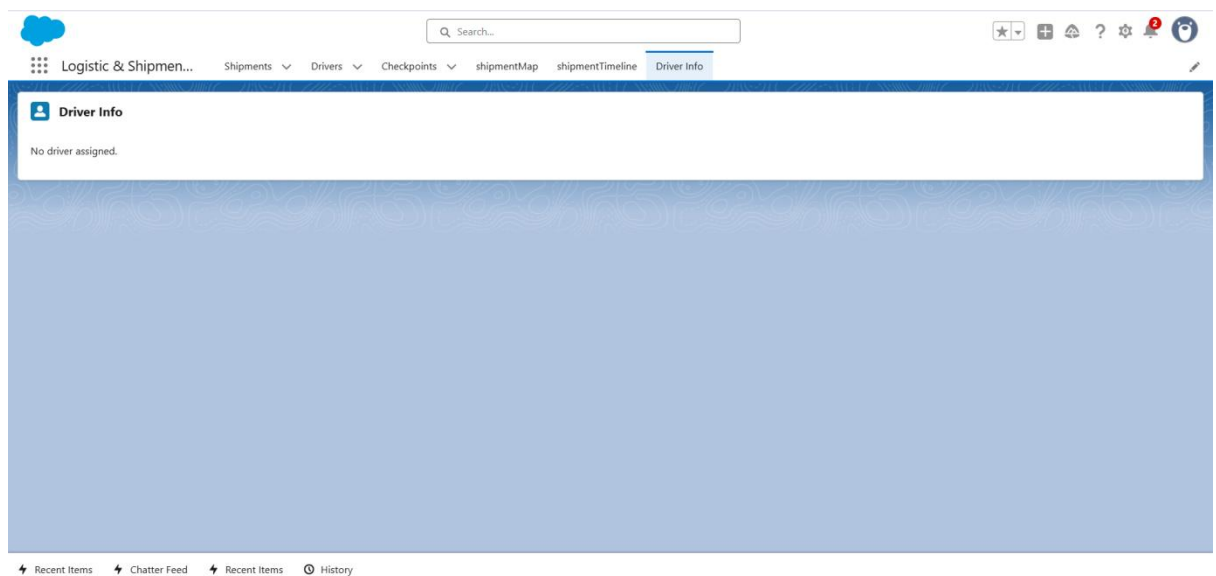


Phase 6: User Interface Development

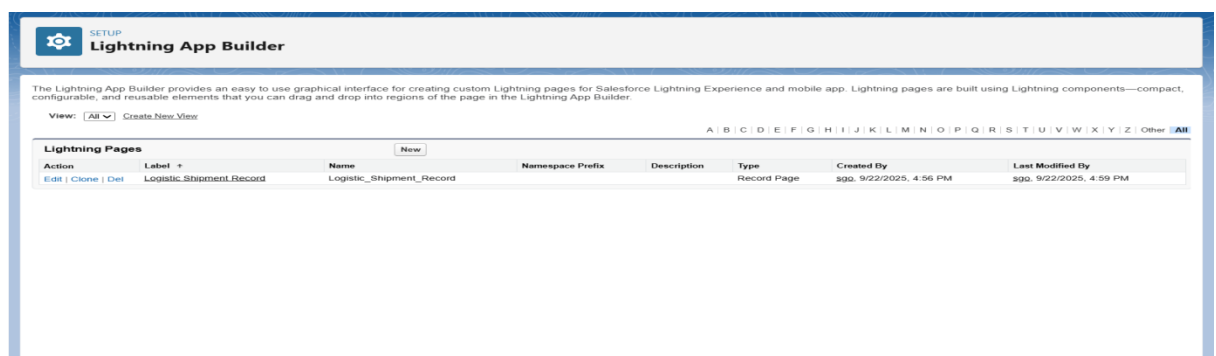
1. Lightning App Builder

- **Driver Info Tab:** Added a custom Lightning page to display driver assignment details for a shipment.
- **Dynamic Visibility:** Shows “No driver assigned” if no driver is linked to the shipment.
- **Purpose:** Provides users with quick, context-aware shipment + driver insights directly in the UI.



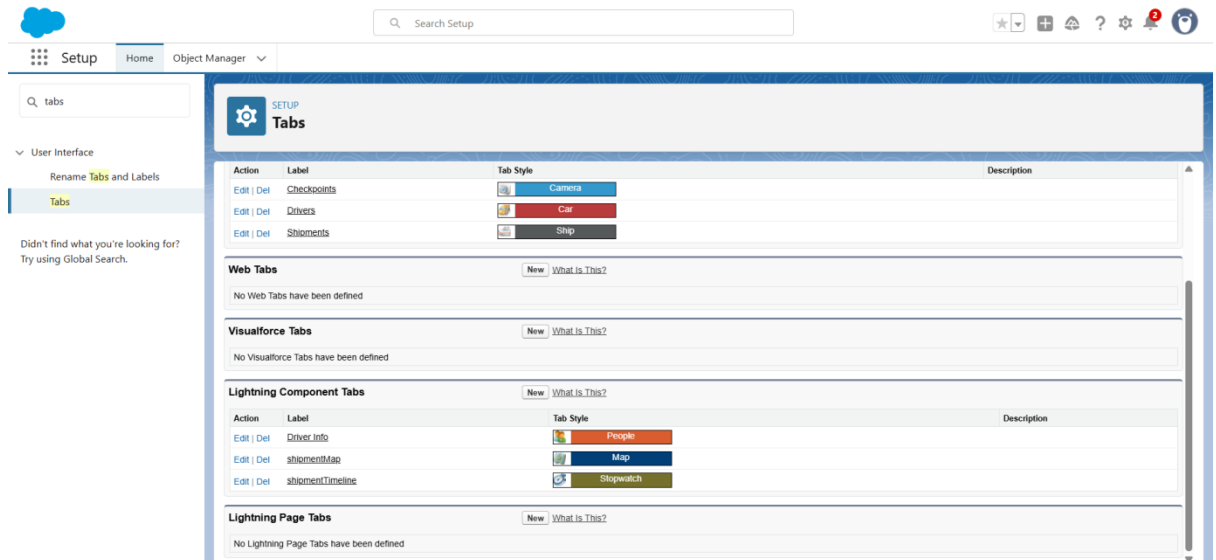
2. Record Pages

- **Created a custom Lightning Record Page (Logistic_Shipment_Record) in Lightning App Builder** to display shipment-related components in one unified layout.
- **Configured the page for extensibility**, enabling integration of dashboards and real-time components (e.g., shipment notifications, driver info) directly into the record view.



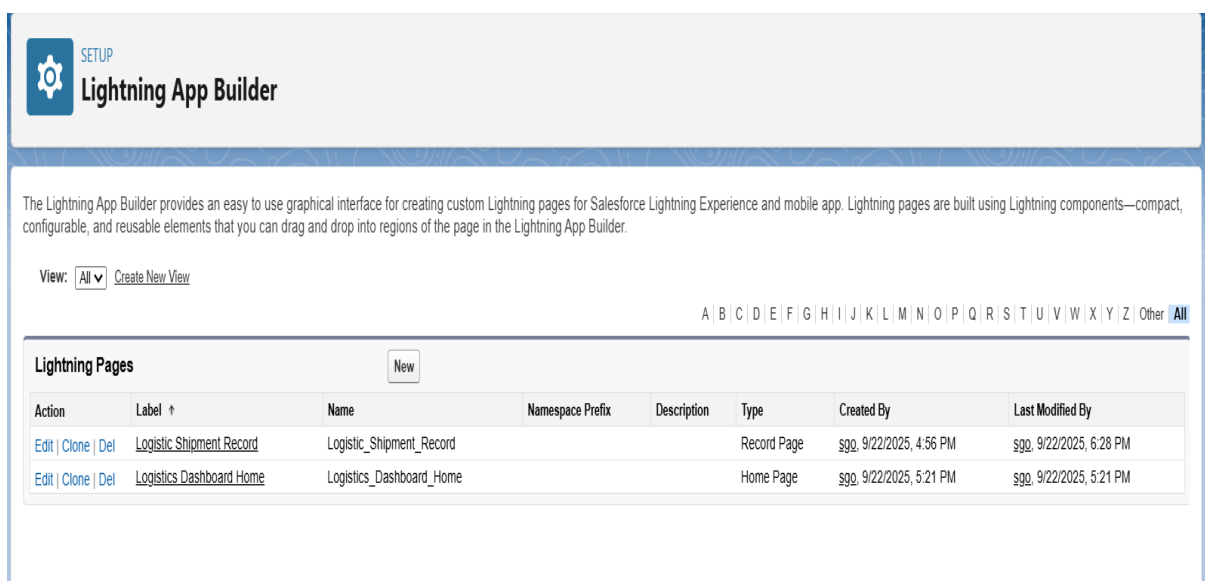
3. Tabs

- **Created Custom Tabs** for Checkpoints, Drivers, and Shipments to allow easy navigation and access to respective Salesforce objects.
- **Configured Lightning Component Tabs** like Driver Info, Shipment Map, and Shipment Timeline to display interactive, real-time data directly within the Lightning App.



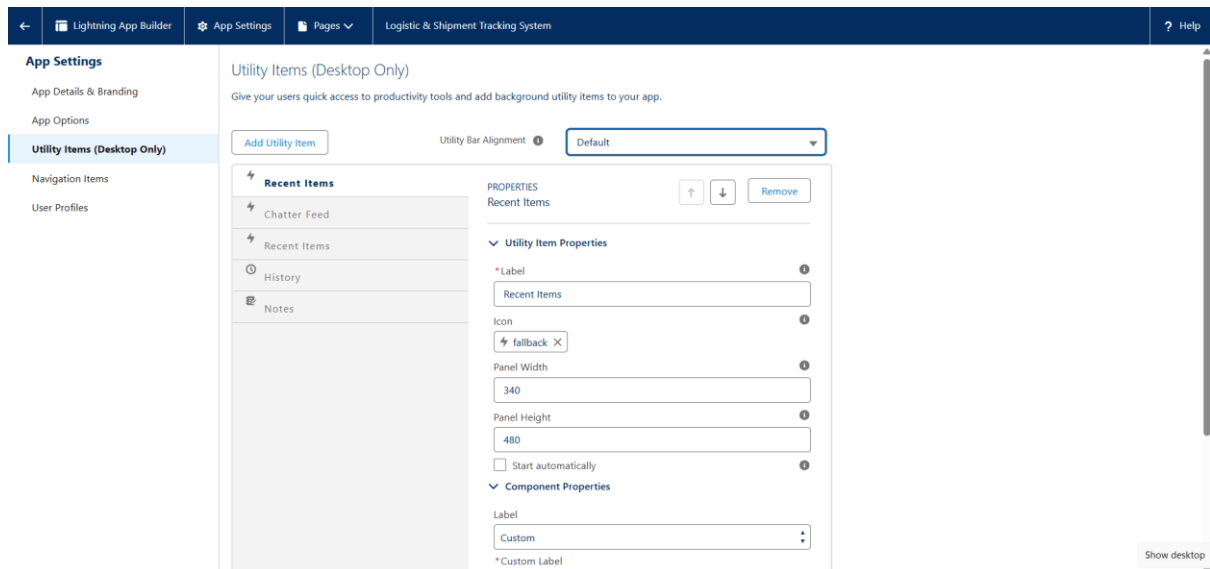
4. Home Page Layouts

- **Created a custom Home Page (Logistics Dashboard Home)** using Lightning App Builder to provide a centralized view for logistics KPIs, dashboards, and recent records.
- **Configured and activated the Home Page** so that users can directly access shipment insights, driver availability, and performance metrics upon login.



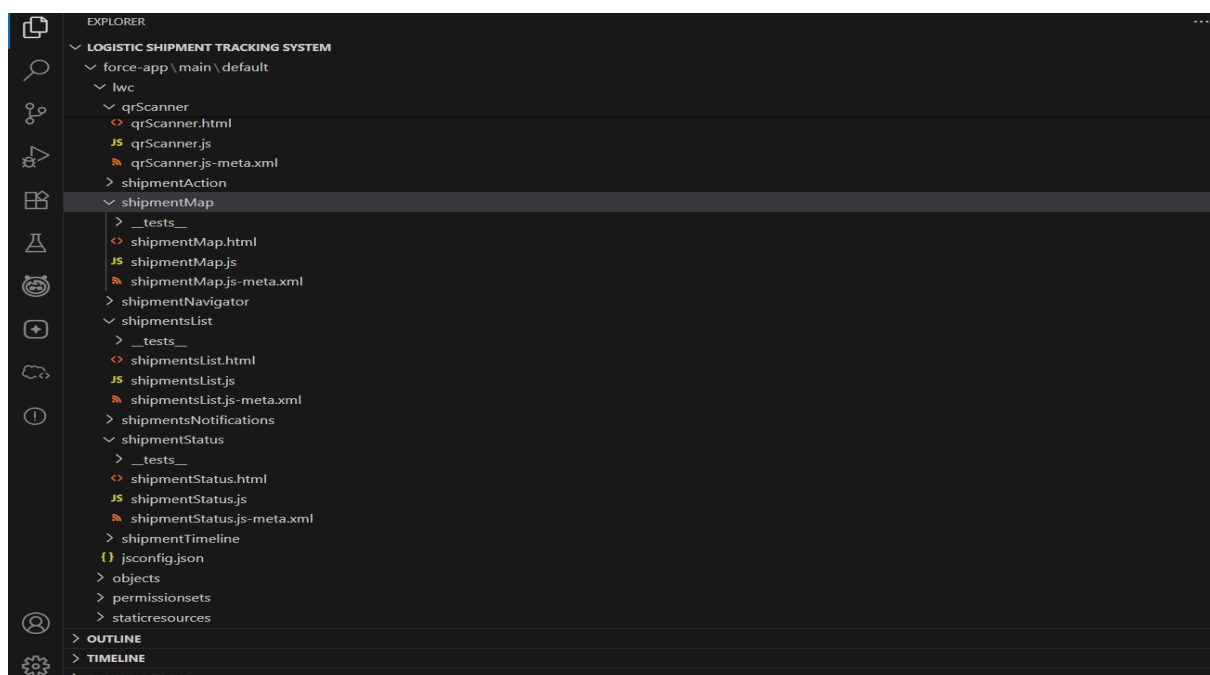
5. Utility Bar

- Configured **Utility Items** (Recent Items, Chatter Feed, History, Notes) for quick access within the app.
- Customized properties (panel width, height, labels) to optimize the user experience in the **Logistic & Shipment Tracking System App**.



6. LWC (Lightning Web Components)

- Developed multiple **Lightning Web Components (LWCs)** such as qrScanner, shipmentMap, shipmentNavigator, shipmentsList, shipmentsNotifications, shipmentStatus, and shipmentTimeline.
- Each LWC is designed to handle a specific functionality (e.g., **QR code scanning, live shipment tracking, status updates, timeline visualization**) for the **Logistic Shipment Tracking System**.



7. Apex with LWC

- **Apex Controller with LWC :** The `ShipmentController.cls` Apex class exposes methods like `updateShipmentStatus`, `assignDriver`, `sendShipmentNotification`, and `getShipmentsByStatus`. These methods allow LWC components to interact with Salesforce data (e.g., update shipment status, assign drivers, or fetch shipments based on conditions).
- **LWC Components Using Apex :** Custom LWC components (`shipmentMap`, `shipmentStatus`, `shipmentTimeline`, etc.) call the Apex methods using `@AuraEnabled` methods. Example: The `shipmentStatus` LWC retrieves shipment data via `getShipmentsByStatus` and displays the live shipment progress in the UI, creating a seamless integration between backend (Apex) and frontend (LWC).

```
force-app > main > default > classes > ShipmentController.cls > ...
1 public with sharing class ShipmentController {
2
3     // Update status of a shipment
4     public void updateShipmentStatus(Id shipmentId, String newStatus) {
5         Shipment__c shipment = [SELECT Id, Status__c FROM Shipment__c WHERE Id = :shipmentId LIMIT 1];
6         shipment.Status__c = newStatus;
7         update shipment;
8     }
9
10    // Assign a driver to a shipment
11    public void assignDriver(Id shipmentId, Id driverId) {
12        Shipment__c shipment = [SELECT Id, Assigned_Driver__c FROM Shipment__c WHERE Id = :shipmentId LIMIT 1];
13        shipment.Assigned_Driver__c = driverId;
14        update shipment;
15    }
16
17    // Send shipment notification (example method, just logs)
18    public void sendShipmentNotification(Id shipmentId) {
19        Shipment__c shipment = [SELECT Id, Status__c FROM Shipment__c WHERE Id = :shipmentId LIMIT 1];
20        System.debug('Notification: Shipment ' + shipment.Id + ' is now ' + shipment.Status__c);
21        // You can add real email or platform event logic here
22    }
23
24    // Retrieve shipments by status
25    public List<Shipment__c> getShipmentsByStatus(String status) {
26        return [SELECT Id, Status__c, Assigned_Driver__c, Checkpoint__c
27                FROM Shipment__c
28                WHERE Status__c = :status];
29    }
30 }
```

```
force-app > main > default > classes > ShipmentControllerTest.cls > ShipmentControllerTest > testAssignDriver() : void
1 @isTest
2 private class ShipmentControllerTest {
3
4     @isTest
5     static void testUpdateShipmentStatus() {
6         // Arrange: Create a shipment
7         Shipment__c s = new Shipment__c(Status__c = 'Pending');
8         insert s;
9
10        ShipmentController controller = new ShipmentController();
11
12        // Act: Update status
13        Test.startTest();
14        controller.updateShipmentStatus(s.Id, 'Dispatched');
15        Test.stopTest();
16
17        // Assert: Check update
18        Shipment__c updated = [SELECT Status__c FROM Shipment__c WHERE Id = :s.Id];
19        System.assertEquals('Dispatched', updated.Status__c, 'Shipment status should update');
20    }
21
22    @isTest
23    static void testAssignDriver() {
24        // Arrange: Shipment + fake driver Id (using Contact here, replace with real Driver__c if exists)
25        Shipment__c s = new Shipment__c(Status__c = 'Pending');
26        insert s;
27
28        Contact driver = new Contact(LastName = 'Driver Test');
29        insert driver;
30
31        ShipmentController controller = new ShipmentController();
32
33        // Act: Assign driver
34        Test.startTest();
35        controller.assignDriver(s.Id, driver.Id);
36        Test.stopTest();
37    }
38 }
```

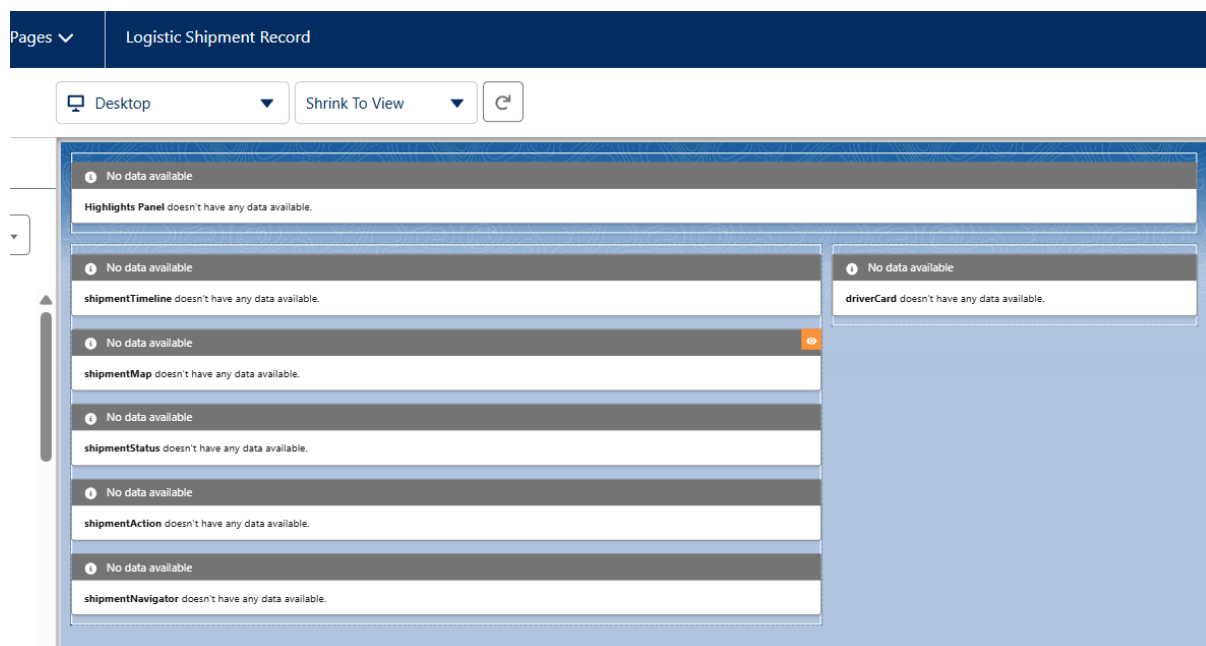
8. Wire Adapters

- Use of Wire Adapters in LWC

- Lightning Web Components like `shipmentTimeline`, `shipmentMap`, `shipmentStatus`, and `driverCard` use **@wire adapters** to fetch real-time data from Salesforce objects without explicitly writing Apex calls.
- Example: `@wire(getRecord, { recordId: '$recordId', fields: [...] })` retrieves live shipment or driver details directly from the database.

- Integration in Lightning Page

- These LWC components are embedded into the **Logistic Shipment Record Page** through the Lightning App Builder.
- When a shipment record is opened, wire adapters automatically fetch and refresh data such as status, assigned driver, checkpoints, and timeline — ensuring up-to-date tracking information.



9. Imperative Apex Calls

- **Manual Control of Apex Execution**
- In the `shipmentAction.js` component, the method `handleMarkDelivered()` calls an Apex method **imperatively** instead of using `@wire`. This allows more flexibility, as the call runs **only when triggered** (e.g., button click), not automatically.
- **Error Handling with Toast Messages**
- If the Apex call fails, a `ShowToastEvent` is dispatched to display a user-friendly error notification. Example shown in the screenshot:
- This improves user experience by giving real-time feedback when something goes wrong.

```

force-app > main > default > lwc > shipmentAction > JS shipmentAction.js > ...
5   export default class ShipmentAction extends LightningElement {
9       handleMarkDelivered() {
11          .then(() => {
19              });
20          })
21          .catch(error => {
22              // Error toast
23              this.dispatchEvent(
24                  new ShowToastEvent({
25                      title: 'Error',
26                      message: error.body ? error.body.message : error.message,
27                      variant: 'error'
28                  })
29              );
30          });
31      }
32  }
33

force-app > main > default > lwc > shipmentAction > <> shipmentAction.html > ...
1   <template>
2       <lightning-card title="Shipment Actions">
3           <div class="slds-p-around_medium">
4               <lightning-button
5                   label="Mark Delivered"
6                   onclick={handleMarkDelivered}>
7               </lightning-button>
8           </div>
9       </lightning-card>
10  </template>
11

force-app > main > default > lwc > shipmentAction > 📄 shipmentAction.js-meta.xml
1   <?xml version="1.0" encoding="UTF-8"?>
2   <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3       <apiVersion>58.0</apiVersion>
4       <isExposed>true</isExposed>
5       <targets>
6           <target>lightning__RecordPage</target>
7       </targets>
8   </LightningComponentBundle>
9

```

10. Navigation Service

- Exposed LWC to **Record Page**, **App Page**, and **Home Page** for flexible use.
- Used **Navigation Service** to redirect users to records, lists, dashboards, and external pages.
- Improved **user experience** with quick, context-based navigation.

