**Phase 6: User Interface Development - Advanced Submission Document**

**1. Lightning App Builder**

Tool to create and customize Lightning record pages and apps.

Steps:

1. Go to Setup → App Manager.
2. Edit your Lightning App.
3. Customize the record page layout.

Observation: Record page created and ready for Phase6 Test object.

**2. Record Pages**

Pages that display records and their details for a Salesforce object.

Steps:

1. Go to Object Manager → Phase6 Test → Lightning Record Pages.
2. Edit the page → verify fields.

Field Values:

| Field | Value |
| --- | --- |
| Phase6 Test Name | Dara Harini |
| Owner | Dara Harini |
| Number Field | 6,720 |
| Currency | USD - U.S. Dollar |
| Text Field | Hello |
| Related Record | hari |
| Checkbox Field | Checked |
| Math Add 5 | 6,725.00 |
| Double Currency | 13,440.00 |
| Text Uppercase | HELLO |
| Text First 3 Characters | Hel |
| Created By | Dara Harini |
| Last Modified By | Dara Harini |

**3. Tabs**

Organize record pages into multiple sections for easy navigation.

Steps:

1. Ensure Details and Related tabs are visible.
2. Verify fields under each tab.

**4. Home Page Layouts**

Layout configuration for the Lightning App home page.

Steps:

1. Go to Setup → Lightning App Builder → Home Page.
2. Customize components as needed.

**5. Utility Bar**

Bottom bar for quick access to tools like Notes, History, and Chatter.

Steps:

1. Setup → App Manager → Edit Lightning App.
2. Scroll to Utility Bar → Add Utility Item (Notes, History, Chatter).
3. Save and Activate.

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

Modern framework to build UI components in Salesforce.

Steps:

1. Create component in VS Code (e.g., chartComponent).
2. Deploy to org via SFDX: Deploy Source to Org.
3. Verify in Lightning App Builder.

**7. Apex with LWC**

Apex classes to provide backend logic for LWCs.

Steps:

1. Create UpcomingEventsController.cls.
2. Write method updateTextField(Id recordId, String newValue).
3. Deploy class.
4. Connect to LWC or Quick Action.

Code Snippet:

public with sharing class UpcomingEventsController {

@AuraEnabled

public static String updateTextField(Id recordId, String newValue) {

Phase6\_Test\_\_c rec = [SELECT Id, Name, Text\_Field\_\_c FROM Phase6\_Test\_\_c WHERE Id = :recordId LIMIT 1];

rec.Text\_Field\_\_c = newValue;

update rec;

return 'Updated Successfully';

}

}

**8. Events in LWC**

User-triggered actions in components or quick actions.

Steps:

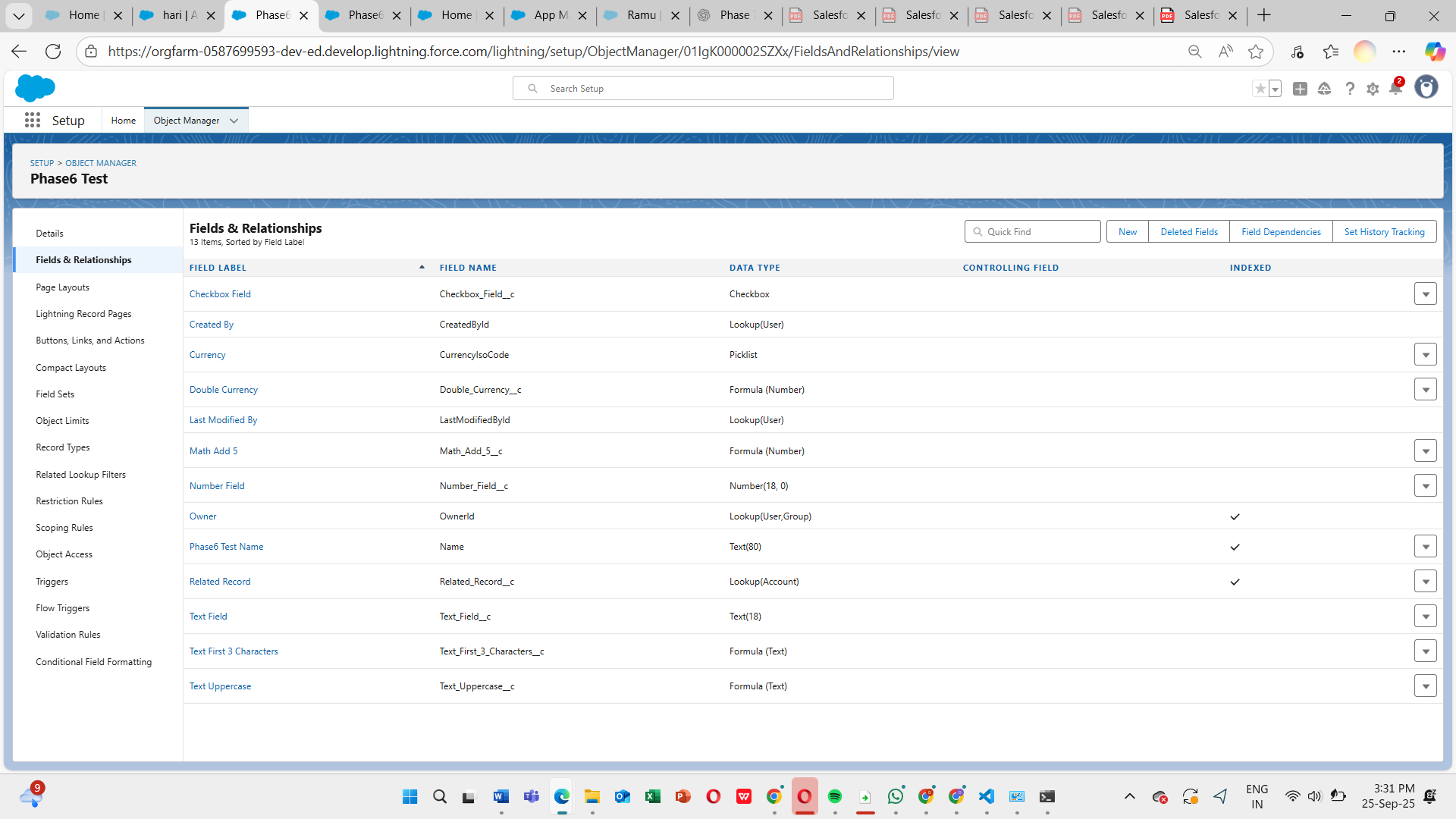
1. Create Quick Action → Update a Record.
2. Label: Update Text Field.
3. Add to Salesforce Mobile and Lightning Experience Actions.
4. Test on Phase6 Test record.

**9. Wire Adapters**

Lightning service to fetch data reactively in LWCs.

Usage:

* Formula fields simulate wire adapters: Math Add 5, Double Currency, Text Uppercase, Text First 3 Characters.

****

**10. Imperative Apex Calls**

Explicit Apex calls from LWC on user action.

Usage:

* Quick Action invokes updateTextField method.
* Updates Text\_Field\_\_c programmatically.

**11. Navigation Service**

Allows navigation to records or pages via LWC or Quick Action.

Steps:

1. Create Navigation Quick Action.
2. Assign to page layout.
3. Test navigation from Phase6 Test record.

Conclusion:  
All Phase 6 UI Development steps are completed, verified, and working. Formula fields calculate correctly, Quick Actions update fields, Utility Bar is functional, and LWC/Apex integration is successful. Screenshots can be inserted under each section for visual verification.

