Phase 6: User Interface Development (UI)

In this phase, we focused on building a clean and user-friendly interface using Salesforce Lightning tools. The goal was to provide sales, service, and partner users with simplified access to customer data, follow-ups, and quick actions through custom Lightning Apps, Record Pages, and Lightning Web Components (LWCs).

1) Lightning App Builder

Created a custom Lightning App: CRM Console

Navigation Items added:
- Leads
- Opportunities
- Cases
- Partner Engagement (custom tab)
- Community Posts (custom tab)
- Reports

☞ Users now have a dedicated workspace to handle customer, partner, and service-related tasks.

2) Record Pages

Created a custom Record Page for Account (Customer 360).

Added components:
- Highlights Panel → Shows key Account details at the top.
- Customer360 LWC → Custom component to display Account Name, Email, Phone.
- Related Lists → Opportunities, Cases, and Partner Engagements.
- Activity Timeline → Shows Tasks & follow-ups.

☞ Teams can now view all customer information in a single 360° view.

3) Tabs

Created custom object tabs for:
- Partner_Engagement__c
- Community_Post__c
- Feedback__c

Added them to the CRM Console navigation.
☞ Users can directly access partner and community data from the app.

4) Home Page Layouts

Customized Home Page for managers and executives:
- Added Reports dashboard (Pipeline by Stage, Case Summary).
- Added Recent Records for quick access.

5) Utility Bar

Configured Utility Bar in the console app (desktop only):
- Notes → Quick note-taking during customer calls.
- History → Navigate back to recently opened records.

6) LWC (Lightning Web Component)

Built Customer360 LWC in VS Code and deployed to Salesforce.

Files created:
- customer360.html
- customer360.js
- customer360.js-meta.xml

☞ Shows customer's Name, Email, Phone on Account Record Page.

Code Example (HTML):

```
<template>
  <lightning-card title="Customer 360">
    <div class="slds-p-around_medium">
      <template if:true={account}>
        <p><strong>Name:</strong> {accountName}</p>
        <p><strong>Email:</strong> {accountEmail}</p>
        <p><strong>Phone:</strong> {accountPhone}</p>
      </template>
      <template if:false={account}>
        <p>Loading...</p>
      </template>
    </div>
  </lightning-card>
</template>
```

7) Apex with LWC

Customer360 LWC uses Lightning Data Service (getRecord), so no Apex was required.

☞ Apex service classes like EnrollmentService.cls remain available for future enhancements, such as enrolling a customer directly from the UI.

8) Events in LWC

LWC prepared to handle future button events (like "Create Case" or "Assign Partner").
Currently displays data via reactive UI.

9) Wire Adapters

Used @wire(getRecord) in customer360.js to fetch Account fields (Name, Email, Phone).
Data auto-refreshes when record changes.

10) Imperative Apex Calls

Not implemented in this phase (planned for future integrations such as Payment Gateway or Partner API calls).

Example use case: Call Apex method to create Partner Engagement directly from LWC.

11) Navigation Service

Prepared to use NavigationMixin in LWC for quick navigation.
Example: Clicking a button could navigate from Customer360 to related Opportunities or Cases.

✓ Result

Phase 6 delivered a CRM Console app with:
- Custom navigation (Leads, Opportunities, Cases, Partner Engagement, Community Posts, Reports)
- Account Record Page with Customer360 view
- Quick access to Opportunities, Cases, and Partner interactions
- Utility bar for productivity

☞ Users can now manage customer and partner data in one place with minimal clicks.