

Phase 6: User Interface Development

Introduction

Phase 6 focused on the UI/UX layer of CLMS using Salesforce Lightning Experience.

The objective was to create user-friendly, modular, and dynamic interfaces so that Loan Officers, Credit Managers, and Branch Managers can easily interact with loan applications, EMIs, approvals, and related records.

◆ Lightning App Builder

- Built custom pages (App Page, Record Page, Home Page).
- In CLMS:
 - Loan Application Record Page → shows Loan details + EMI Schedule.
 - CLMS Home Page → shows recent items, loan reports, pending approvals.
- Outcome: Flexible layouts for different roles.

The screenshot shows the Lightning App Builder interface. At the top, there's a header with a gear icon labeled 'SETUP' and the title 'Lightning App Builder'. Below the header, a message states: '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.' There are buttons for 'View: All' and 'Create New View'. A navigation bar at the bottom includes letters from A to Z and an 'Other' link. The main area is titled 'Lightning Pages' and contains a table with three rows of data:

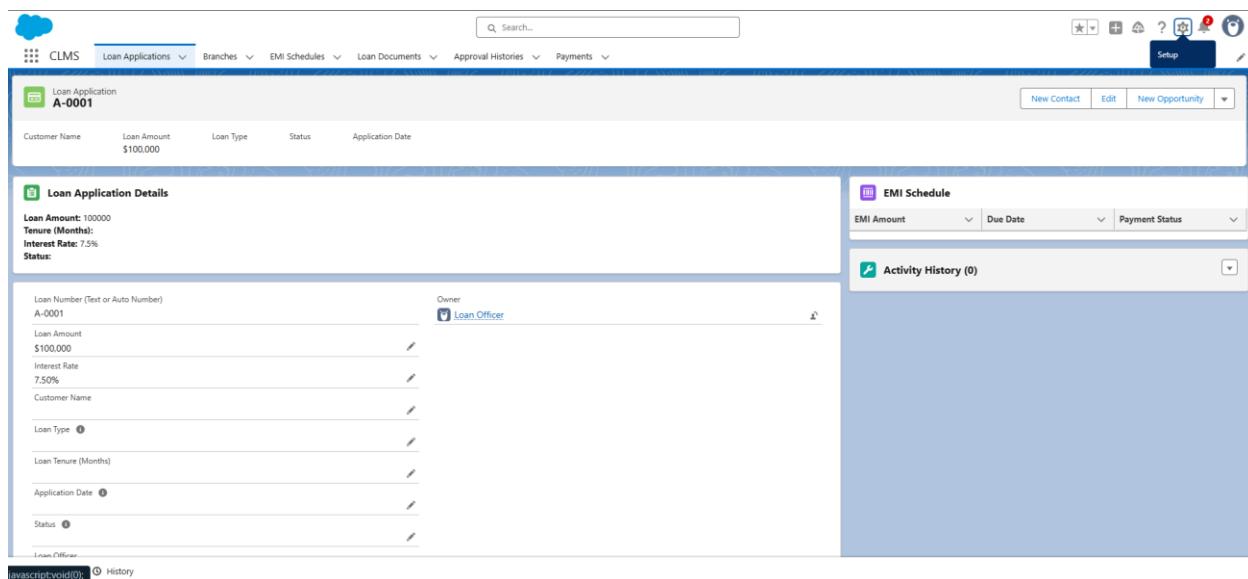
Action	Label	Name	Namespace Prefix	Description	Type	Created By	Last Modified By
Edit Clone Del	CLMS Home Page	CLMS_Home_Page			Home Page	nlk, 9/25/2025, 2:13 AM	nlk, 9/25/2025, 2:13 AM
Edit Clone Del	EMI Schedule Record Page	EMI_Schedule_Record_Page			Record Page	nlk, 9/25/2025, 1:38 AM	nlk, 9/25/2025, 1:54 AM
Edit Clone Del	Loan Application Record Page	Loan_Application_Record_Page			Record Page	nlk, 9/25/2025, 1:30 AM	nlk, 9/25/2025, 2:57 AM

◆ Record Pages

- Custom record pages created for:
 - Loan Application (loan details + EMI Viewer LWC).
 - EMI Schedule (EMI details + related Loan).
 - Loan Document (document details + loan reference).
 - Approval History (role, decision, comments).
- Outcome: Each record has its own optimized Lightning layout.

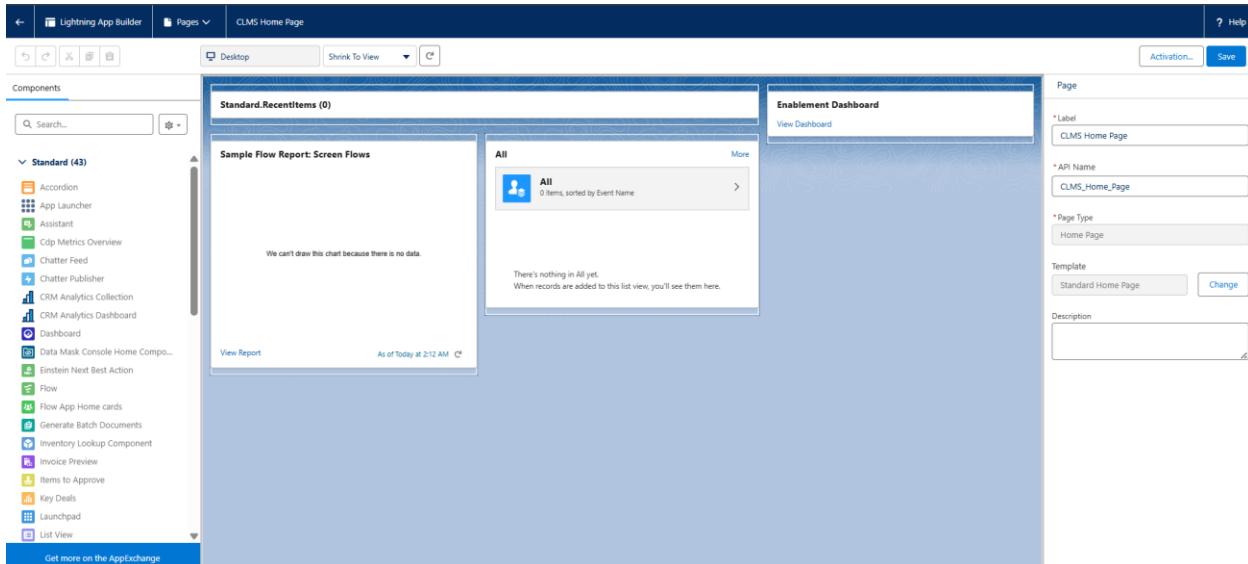
◆ Tabs

- Added CLMS navigation tabs:
 - Loan Application, EMI Schedule, Payment, Loan Document, Approval History.
- Outcome: Quick switching between CLMS objects.



◆ Home Page Layouts

- Created CLMS Home Page via App Builder.
- Components added:
 - Recent Items → quick access to loans.
 - Report Chart → Loans Approved This Month.
 - List View → Pending approvals.
- Outcome: Centralized dashboard.



◆ Utility Bar

- Added Utility Bar in CLMS App.
- Items: Notes, Recent Items.
- Outcome: Quick actions available across the app.



◆ Lightning Web Components (LWCs)

Developed LWCs for CLMS:

1. loanApplicationViewer → displays loan details.

2. emiScheduleViewer → lists EMI records + refresh option.

◆ Apex with LWC

- Added Apex methods in LoanHelper.cls:
 - calculateEMI() → EMI logic.
 - getEmiByLoan() → fetch EMI schedules.
 - Used in LWC: emiScheduleViewer uses @wire + imperative calls.
-

◆ Events in LWC

- Added event handling in LWCs:
 - Example: Refresh button triggers EMI list reload.
 - Outcome: LWCs became interactive and dynamic.
-

◆ Wire Adapters

- Used @wire with getEmiByLoan.
 - Auto-updates EMI list when Loan Application changes.
-

◆ Imperative Apex Calls

- Refresh button → calls Apex imperatively → reloads EMI list.
- LOAN APPLICATION
VIEWER

loanApplicationViewer.js-meta.xml X loanApplicationViewer.html JS loanApplicationViewer.js

force-app > main > default > lwc > loanApplicationViewer > loanApplicationViewer.js-meta.xml > ...

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3      <apiVersion>64.0</apiVersion>
4      <isExposed>true</isExposed>
5      <targets>
6          <target>lightning__RecordPage</target>
7          <target>lightning__AppPage</target>
8      </targets>
9      <targetConfigs>
10         <targetConfig targets="lightning__RecordPage">
11             <objects>
12                 <object>Loan_Application__c</object>
13             </objects>
14         </targetConfig>
15     </targetConfigs>
16 </LightningComponentBundle>
17
```

loanApplicationViewer.js-meta.xml X loanApplicationViewer.html X JS loanApplicationViewer.js

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

```
1  <template>
2      <lightning-card title="Loan Application Details" icon-name="standard:record">
3          <div class="slds-m-around_medium">
4              <p><b>Loan Amount:</b> {loanAmount}</p>
5              <p><b>Tenure (Months):</b> {tenure}</p>
6              <p><b>Interest Rate:</b> {interestRate}%</p>
7              <p><b>Status:</b> {status}</p>
8          </div>
9      </lightning-card>
10 </template>
11
```

```

force-app > main > default > lwc > loanApplicationViewer > loanApplicationViewer.js ...
1 import { LightningElement, api, wire } from 'lwc';
2 import { getRecord } from 'lightning/uiRecordApi';
3
4 import LOAN_AMOUNT from '@salesforce/schema/Loan_Application__c.Loa...
5 import TENURE from '@salesforce/schema/Loan_Application__c.Loan_Tenure_Months__c';
6 import INTEREST_RATE from '@salesforce/schema/Loan_Application__c.Interest_Rate__c';
7 import STATUS from '@salesforce/schema/Loan_Application__c.Status__c';
8
9 export default class LoanApplicationViewer extends LightningElement {
10     @api recordId; // record Id comes from Salesforce Record Page
11
12     loanAmount;
13     tenure;
14     interestRate;
15     status;
16
17     @wire(getRecord, {
18         recordId: 'SrecordId',
19         fields: [LOAN_AMOUNT, TENURE, INTEREST_RATE, STATUS]
20     })
21     wiredLoan({ error, data }) {
22         if (data) {
23             this.loanAmount = data.fields.Loan_Amount__c.value;
24             this.tenure = data.fields.Loan_Tenure_Months__c.value;
25             this.interestRate = data.fields.Interest_Rate__c.value;
26             this.status = data.fields.Status__c.value;
27         } else if (error) {
28             console.error('Error fetching Loan Application:', error);
29         }
30     }
31 }
32

```

- **EMISCHEDULER**
VIEWER

CLMS Project

File Edit Selection View Go Run ...

EXPLORER

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

emiScheduleViewer.js

emiScheduleViewer.js-meta.xml

query.apex

BLACKBOX

TERMINAL

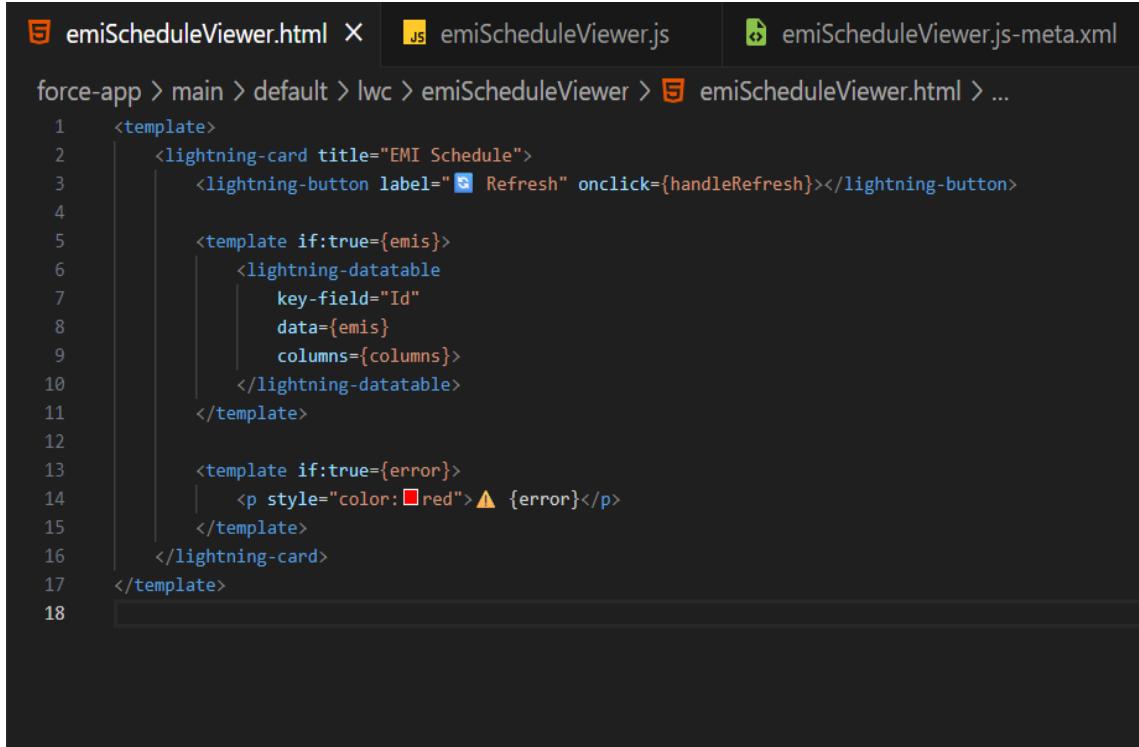
POWER SHELL

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS HISTORY

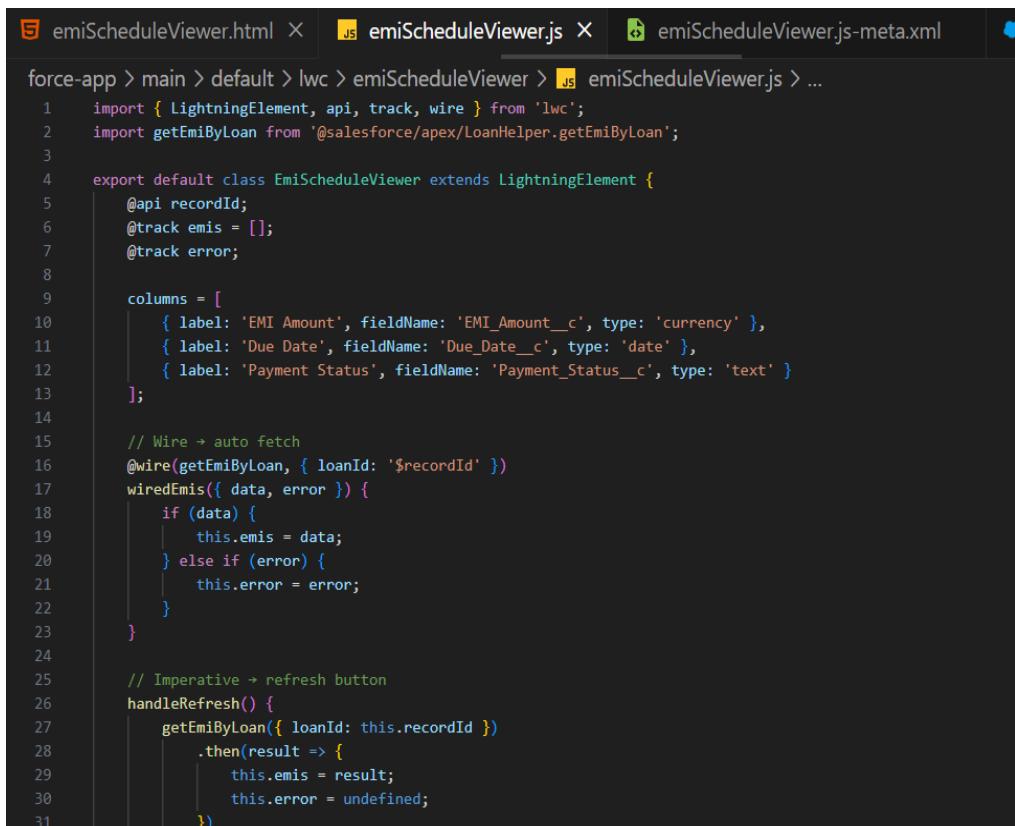
PS C:\Users\Wikihilesh\Downloads\CLMS_Project> sf project deploy start --source-dir force-app/main/default/lwc/emiScheduleViewer >>

Deployed Source

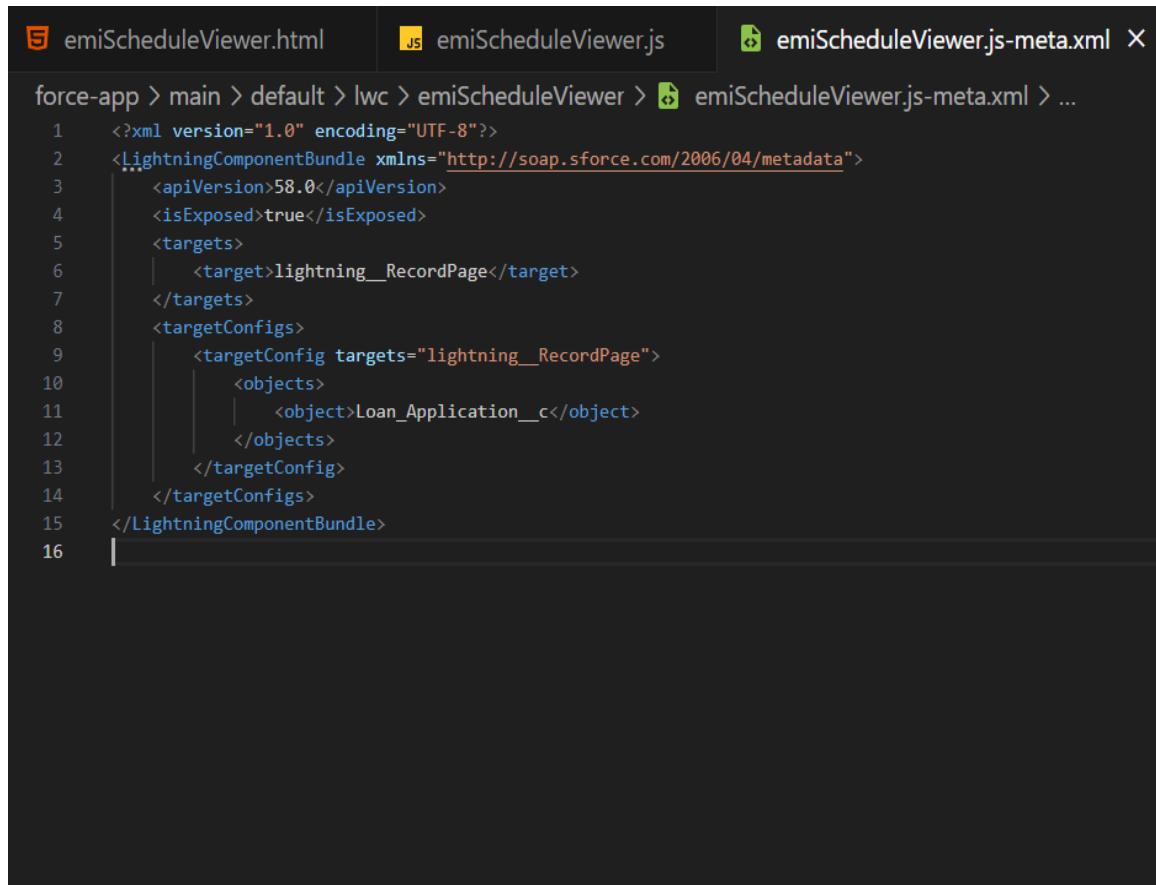
State	Name	Type	Path
Changed	emiScheduleViewer	LightningComponentBundle	force-app\main\default\lwc\emiScheduleViewer\emiScheduleViewer.html
Changed	emiScheduleViewer	LightningComponentBundle	force-app\main\default\lwc\emiScheduleViewer\emiScheduleViewer.js
Changed	emiScheduleViewer	LightningComponentBundle	force-app\main\default\lwc\emiScheduleViewer\emiScheduleViewer.js-meta.xml



```
force-app > main > default > lwc > emiScheduleViewer > emiScheduleViewer.html > ...
1   <template>
2     <lightning-card title="EMI Schedule">
3       <lightning-button label=" Refresh" onclick={handleRefresh}></lightning-button>
4
5       <template if:true={emis}>
6         <lightning-datable
7           key-field="Id"
8           data={emis}
9           columns={columns}>
10          </lightning-datable>
11        </template>
12
13        <template if:true={error}>
14          <p style="color: red">⚠ {error}</p>
15        </template>
16      </lightning-card>
17    </template>
18  
```



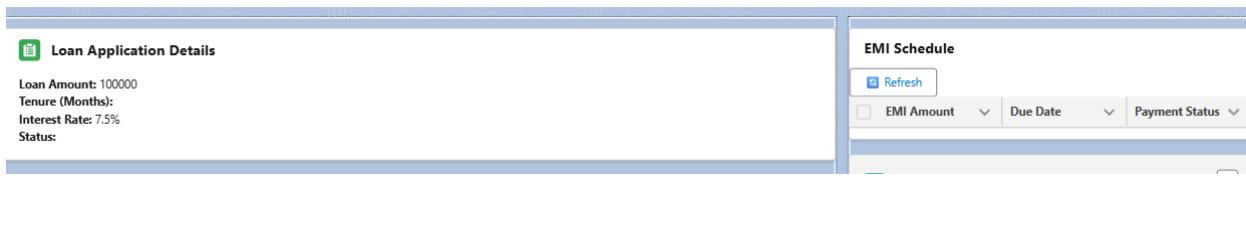
```
force-app > main > default > lwc > emiScheduleViewer > emiScheduleViewer.js > ...
1  import { LightningElement, api, track, wire } from 'lwc';
2  import getEmiByLoan from '@salesforce/apex/LoanHelper.getEmiByLoan';
3
4  export default class EmiScheduleViewer extends LightningElement {
5    @api recordId;
6    @track emis = [];
7    @track error;
8
9    columns = [
10      { label: 'EMI Amount', fieldName: 'EMI_Amount__c', type: 'currency' },
11      { label: 'Due Date', fieldName: 'Due_Date__c', type: 'date' },
12      { label: 'Payment Status', fieldName: 'Payment_Status__c', type: 'text' }
13    ];
14
15    // Wire + auto fetch
16    @wire(getEmiByLoan, { loanId: '$recordId' })
17    wiredEmis({ data, error }) {
18      if (data) {
19        this.emis = data;
20      } else if (error) {
21        this.error = error;
22      }
23    }
24
25    // Imperative → refresh button
26    handleRefresh() {
27      getEmiByLoan({ loanId: this.recordId })
28        .then(result => {
29          this.emis = result;
30          this.error = undefined;
31        })
32    }
33  }
```



```

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      <targetConfigs>
9          <targetConfig targets="lightning_RecordPage">
10             <objects>
11                 <object>Loan_Application__c</object>
12             </objects>
13         </targetConfig>
14     </targetConfigs>
15 </LightningComponentBundle>
16

```



◆ Navigation Service

- Added record navigation inside LWCs.
- Example: From EMI Schedule → back to Loan Application.

3. Outcomes

- Built modular Lightning Pages for CLMS.

- Created custom LWCs integrated with Apex.
 - Enhanced usability with Home Page + Utility Bar.
 - Added auto-refresh (wire) and manual refresh (imperative Apex).
 - Implemented navigation & events for smooth user experience.
-

4. Conclusion

Phase 6 delivered a modern Lightning Experience UI for CLMS.

Users can now work with loans, EMIs, and approvals in a single consolidated workspace.

With LWCs and Apex integration, the UI is interactive, scalable, and user-friend