Lead Management & Task Tracking System
Phase 6: User Interface Development (UI)

1. Introduction

This phase focuses on building a user-friendly interface in Salesforce for **lead management and task tracking**. A well-designed UI improves sales rep productivity, ensures accurate data entry, and provides quick access to key metrics.

Objectives:

- Create intuitive pages for leads, tasks, and dashboards.
- Use Lightning Experience for modern, responsive UI.
- Integrate custom components with business logic.

2. Lightning App Builder

- A drag-and-drop interface to build **custom apps and pages**.
- Allows combining standard, custom, and third-party components.

Examples:

- Build a **Lead Management App** with tabs for Leads, Tasks, and Reports.
- Customize pages for sales reps and managers.

3. Record Pages

- Customize the layout for **Lead** and **Task** records.
- Include standard and custom fields, related lists, and components.

Examples:

- Lead record page shows lead details, activity history, tasks, and related opportunities.
- Task page shows details, assigned rep, due date, and related lead.

4. Tabs

- Organize different objects and pages using **tabs** for easy navigation.

Examples:

- Tabs for Leads, Tasks, Opportunities, Accounts, and Reports in the app.

5. Home Page Layouts

- Customize **home page layouts** for sales reps and managers.

Examples:

- Sales rep homepage: Today's tasks, recently updated leads, pipeline summary.
- Manager homepage: Team performance charts, overdue tasks, lead conversion statistics.

6. Utility Bar

- Provides quick access to frequently used tools at the bottom of the app.

Examples:

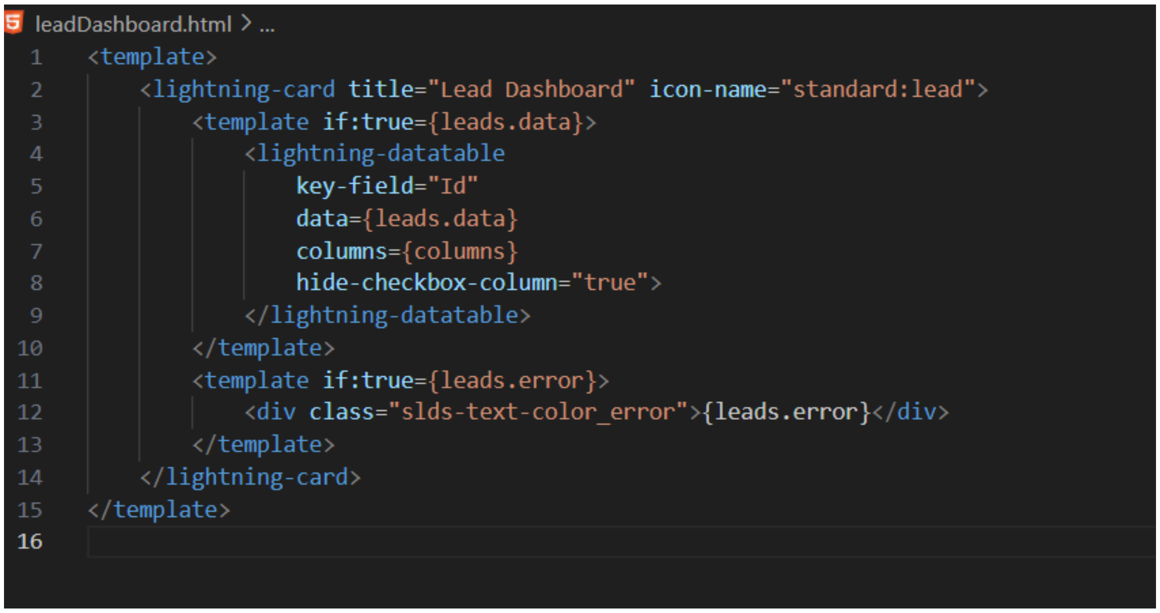
- Quick actions like "New Lead," "Create Task," "Send Email," and "Log Call."

7. LWC (Lightning Web Components)

- Modern **custom components** built using HTML, CSS, and JavaScript.
- Allows dynamic interaction with Apex and Salesforce data.

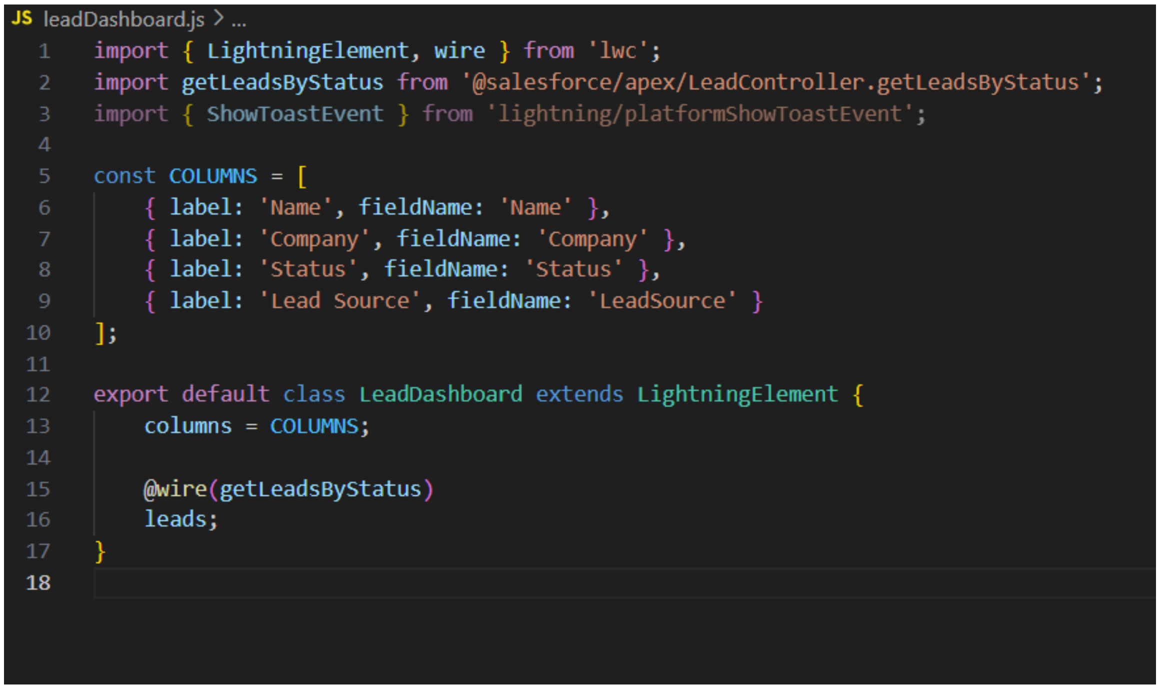
Examples:

- Custom Lead Dashboard showing high-priority leads with color-coded status.
- Task list component with filters for due dates and priority.



leadDashboard.html > ...

```html
<template>
    <lightning-card title="Lead Dashboard" icon-name="standard:lead">
        <template if:true={leads.data}>
            <lightning-datatable
                key-field="Id"
                data={leads.data}
                columns={columns}
                hide-checkbox-column="true">
            </lightning-datatable>
        </template>
        <template if:true={leads.error}>
            <div class="slds-text-color_error">{leads.error}</div>
        </template>
    </lightning-card>
</template>
```

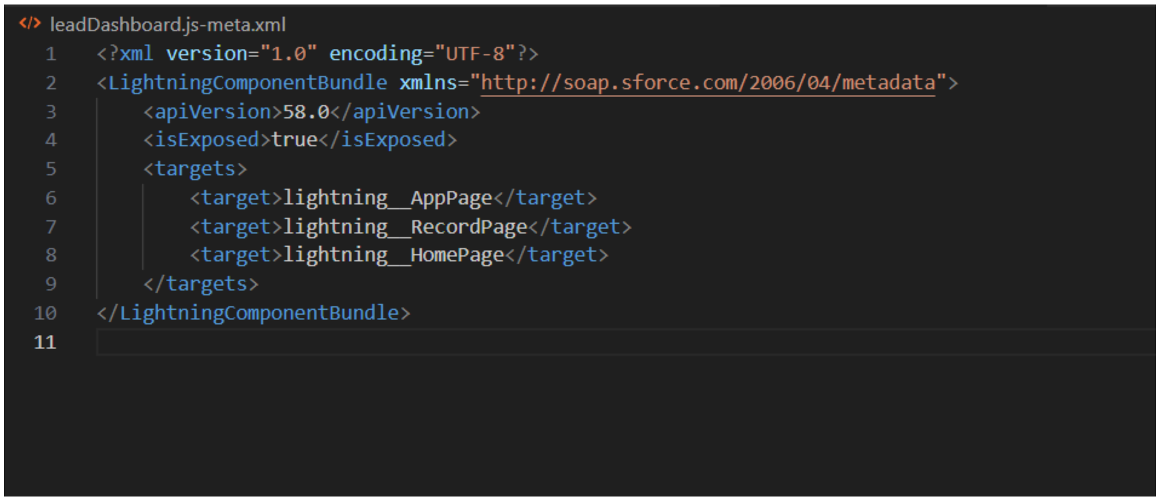

JS leadDashboard.js > ...

```javascript
import { LightningElement, wire } from 'lwc';
import getLeadsByStatus from '@salesforce/apex/LeadController.getLeadsByStatus';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';

const COLUMNS = [
    { label: 'Name', fieldName: 'Name' },
    { label: 'Company', fieldName: 'Company' },
    { label: 'Status', fieldName: 'Status' },
    { label: 'Lead Source', fieldName: 'LeadSource' }
];

export default class LeadDashboard extends LightningElement {
    columns = COLUMNS;

    @wire(getLeadsByStatus)
    leads;
}
```


leadDashboard.js-meta.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>58.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```