Phase 6: User Interface Development

Project: Customer Success & Support Management CRM
Industry: SaaS / Technology Companies

Lightning App Builder

- Used Lightning App Builder to design **Customer Success Dashboard** for the **Account object**.
- Added LWCs (Health Score, Support Queue, Onboarding Tasks) on the record page.
- Created an optional **App Page** for managers with reports, dashboards, and LWCs.

Record Pages

- Created **Customer Success Dashboard Record Page** under Account.
- Components added:
    - **HealthScoreDashboard LWC** → Top section (account health overview).
    - **SupportCasesQueue LWC** → Middle section (open support cases).
    - **OnboardingTasks LWC** → Bottom section (pending onboarding tasks).
- Activated page as **Org Default** for Account records.

Tabs

- Organized record page with Tabs for clarity:
    - **Overview Tab** → Health Score.
    - **Support Tab** → Support Cases.
    - **Onboarding Tab** → Onboarding Tasks.

Home Page Layouts

- Created optional **Customer Success Dashboard App Page**.
- Displayed reports (churn prediction, support response times, onboarding progress).
- Added LWCs for health score, cases, and tasks.
- Provided consolidated view for Customer Success managers.

Utility Bar

- Added Utility Bar to the **Customer Success CRM App**.
- Included:
    - **Recent Records**
    - **Notes**
    - **Quick Actions** (“Create Case”, “Log Usage Metric”)

LWC (Lightning Web Components)

- Developed 3 LWCs for the project:
    - **HealthScoreDashboard** → Fetches & displays Account Health Score.
    - **SupportCasesQueue** → Lists support cases with navigation to records.
    - **OnboardingTasks** → Shows pending onboarding tasks with Complete action.

Apex with LWC

- Built Apex controllers for data handling:
    - **HealthScoreController** → Returns health score for an account.
    - **SupportCasesController** → Fetches support cases for the account.
    - **OnboardingTasksController** → Fetches onboarding tasks & updates task completion.
- Used Apex with LWCs for **real-time dynamic data updates**.

Events in LWC

- Implemented event communication between LWCs:
    - Completing a task in **OnboardingTasks** → Refreshes **HealthScoreDashboard**.
    - Ensures **real-time updates** without page reload.

Wire Adapters

- Used @wire adapters for automatic data refresh:
    - Fetch Account Health Score.
    - Fetch Support Cases.
    - Fetch Onboarding Tasks.

Imperative Apex Calls

- Used where user interaction required backend updates:
    - Marking onboarding tasks as completed.
    - Updating Account health score.
    - Escalating support cases.

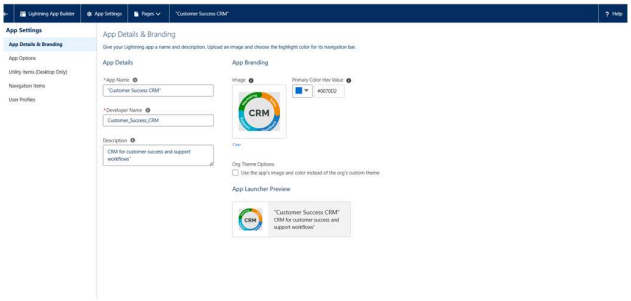
Navigation Service

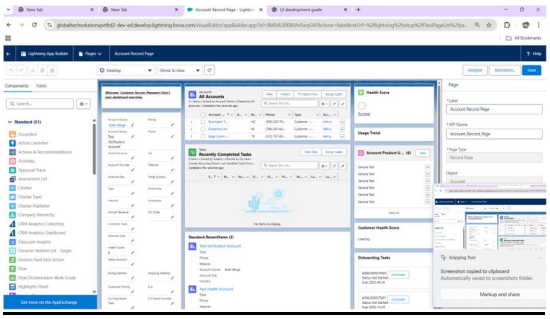
- Implemented **NavigationMixin** in LWCs:
    - From task or case list → Navigate to related Account or Case record.
    - Provides seamless navigation across Salesforce records.

✅ **Outcome**

- Built a **Customer Success Dashboard** directly on the Account record page.
- Dashboard shows **Health Score, Active Cases, Pending Onboarding Tasks**.
- Utility Bar, Tabs, and Navigation improve usability for success teams.
- Managers get both **record-level dashboards** and **App Page summaries**.
- Final result: An integrated, interactive interface for **Customer Success management**.

✨ **Screenshots what I have done**

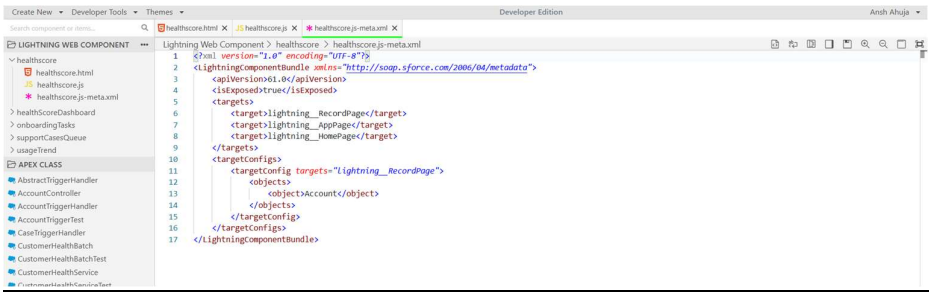

Lightning App Builder
App Settings
Pages
"Customer Success CRM"
Help
App Settings
App Details & Branding
App Options
Utility Items (Desktop Only)
Navigation Items
User Profiles
App Details & Branding
Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.
App Details
App Branding
*App Name
"Customer Success CRM"
*Developer Name
Customer_Success_CRM
Description
CRM for customer success and support workflows"
Image
Primary Color Hex Value
#0070D2
CRM
Org Theme Options
Use the app's image and color instead of the org's custom theme
App Launcher Preview
"Customer Success CRM"
CRM for customer success and support workflows"

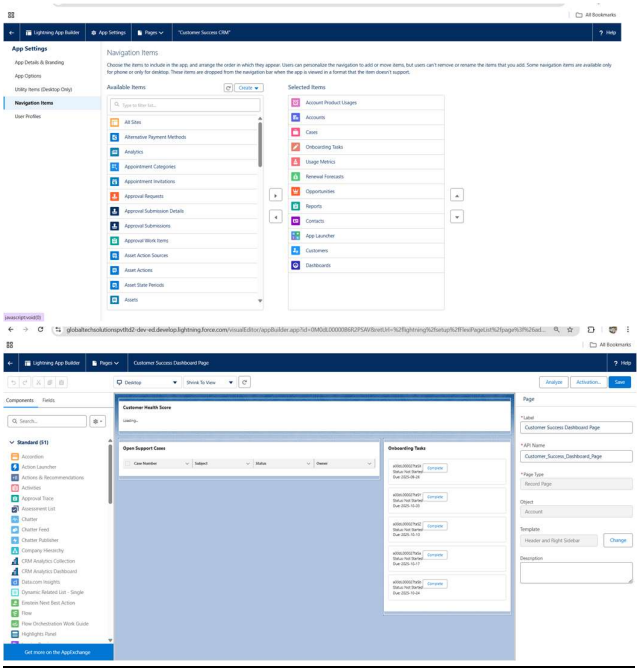



Create New
Developer Tools
Themes
Developer Edition
Ansh Ahuja
healthscore.html
LIGHTNING WEB COMPONENT
Lightning Web Component > healthscore > healthscore.html
healthscore
healthscore.html
healthscore.js
healthscore.js-meta.xml
healthScoreDashboard
onboardingTasks
supportCasesQueue
<template>
<lightning-card title="Health Score" icon-name="custom:custom63">

<lightning-progress-ring value={healthScore} size="Large" variant="active"></lightning-progress-ring>
Score: {healthScore}

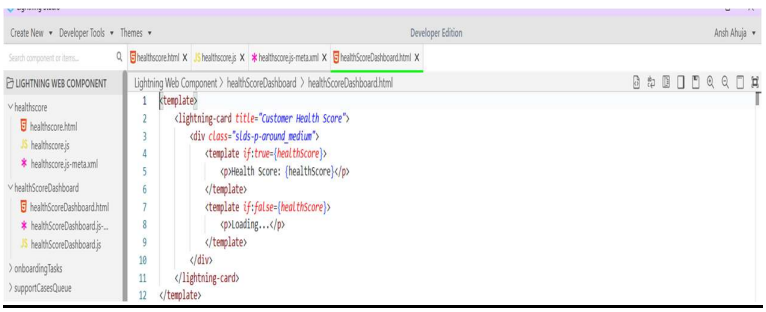
</lightning-card>
</template>


Create New
Developer Tools
Themes
Developer Edition
Ansh Ahuja
healthscore.html
healthscore.js
healthscore.js-meta.xml
LIGHTNING WEB COMPONENT
Lightning Web Component > healthscore > healthscore.js-meta.xml
healthscore
healthscore.html
healthscore.js
healthscore.js-meta.xml
healthScoreDashboard
onboardingTasks
supportCasesQueue
usageTrend
APEX CLASS
AbstractTriggerHandler
AccountController
AccountTriggerHandler
AccountTriggerTest
CaseTriggerHandler
CustomerHealthBatch
CustomerHealthBatchTest
CustomerHealthService
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
<apiVersion>61.0</apiVersion>
<isExposed>true</isExposed>
<targets>
<target>lightning__RecordPage</target>
<target>lightning__AppPage</target>
<target>lightning__HomePage</target>
</targets>
<targetConfigs>
<targetConfig targets="lightning__RecordPage">
<objects>
<object>Account</object>
</objects>
</targetConfig>
</targetConfigs>
</LightningComponentBundle>


Lightning App Builder
App Settings
Pages
"Customer Success CRM"
Help
App Settings
App Details & Branding
App Options
Utility Items (Desktop Only)
Navigation Items
User Profiles
Navigation Items
Choose the items to include in the app, and arrange the order in which they appear. Users can personalize the navigation to add or move items, but users can't remove or rename the items that you add. Some navigation items are available only for phone or only for desktop. These items are dropped from the navigation bar when the app is viewed in a format that the item doesn't support.
Available Items
Create
Selected Items
All Sites
Alternative Payment Methods
Analytics
Appointment Categories
Appointment Invitations
Approval Requests
Approval Submission Details
Approval Submissions
Approval Work Items
Asset Action Sources
Asset Actions
Asset State Periods
Assets
Account Product Usages
Accounts
Cases
Onboarding Tasks
Usage Metrics
Renewal Forecasts
Opportunities
Reports
Contacts
App Launcher
Customers
Dashboards

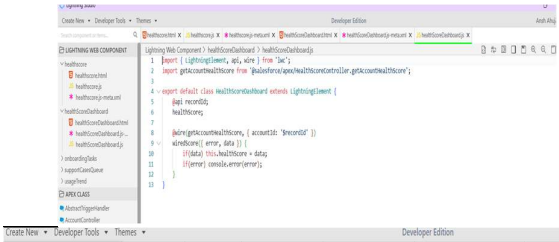
Lightning App Builder
Pages
Customer Success Dashboard Page
Help
Components
Fields
Customer Health Score
Open Support Cases
Onboarding Tasks
Page
Label
Customer Success Dashboard Page
API Name
Customer_Success_Dashboard_Page
Page Type
Record Page
Object
Account
Template
Header and Right Sidebar
Change
Description
Get more on the AppExchange


Create New
Developer Tools
Themes
Developer Edition
Ansh Ahuja
healthscore.html
healthscore.js
LIGHTNING WEB COMPONENT
Lightning Web Component 〉 healthscore 〉 healthscore.js
healthscore
healthscore.html
healthscore.js
healthscore.js-meta.xml
healthScoreDashboard
onboardingTasks
supportCasesQueue
usageTrend
APEX CLASS
AbstractTriggerHandler
AccountController
AccountTriggerHandler
AccountTriggerTest
CaseTriggerHandler
import { LightningElement, wire, api } from 'lwc';
import getHealthScore from '@salesforce/apex/AccountController.getHealthScore';
export default class HealthScore extends LightningElement {
@api recordId;
healthScore;
@wire(getHealthScore, { accountId: '$recordId' })
wiredScore({ error, data }) {
if (data) {
this.healthScore = data;
} else if (error) {
console.error(error);
}
}
}

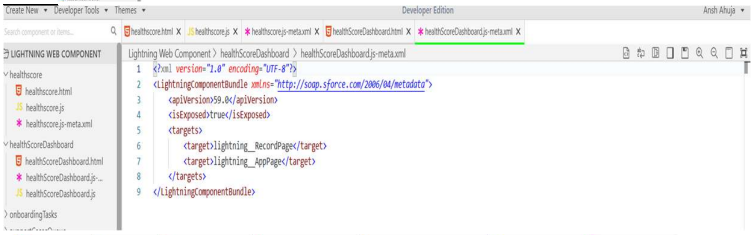

Create New
Developer Tools
Themes
Developer Edition
Ansh Ahuja
healthscore.html
healthscore.js
healthscore.js-meta.xml
healthScoreDashboard.html
LIGHTNING WEB COMPONENT
Lightning Web Component 〉 healthScoreDashboard 〉 healthScoreDashboard.html
healthscore
healthscore.html
healthscore.js
healthscore.js-meta.xml
healthScoreDashboard
healthScoreDashboard.html
healthScoreDashboard.js-...
healthScoreDashboard.js
onboardingTasks
supportCasesQueue
<template>
<lightning-card title="Customer Health Score">

<template if:true={healthScore}>
Health Score: {healthScore}
</template>
<template if:false={healthScore}>
Loading...
</template>

</lightning-card>
</template>

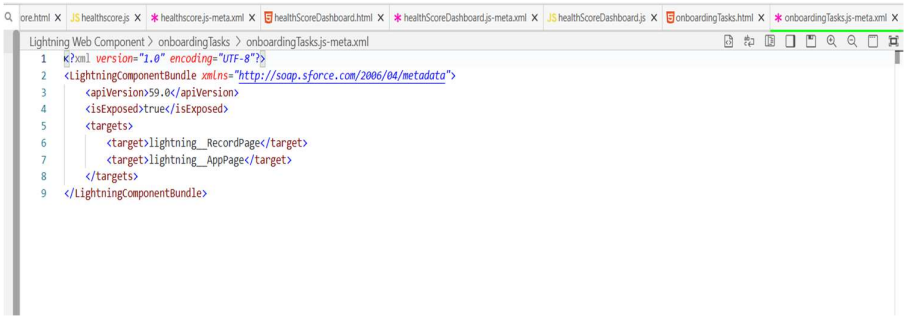


```
import { LightningElement, api, wire } from 'lwc';
import getAccountHealthScore from '@salesforce/apex/HealthScoreController.getAccountHealthScore';

export default class HealthScoreDashboard extends LightningElement {
    @api recordId;
    healthScore;

    @wire(getAccountHealthScore, { accountId: '$recordId' })
    wiredScore({ error, data }) {
        if(data) this.healthScore = data;
        if(error) console.error(error);
    }
}
```

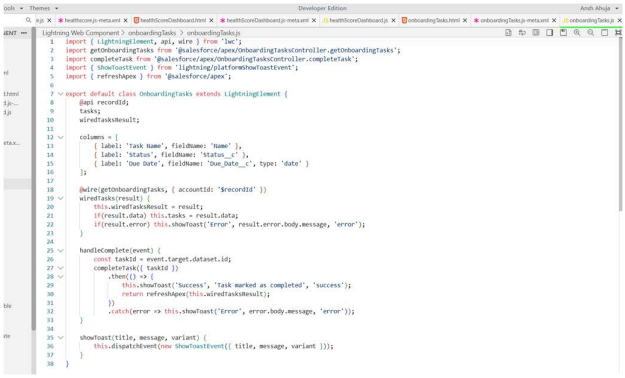


```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
    </targets>
</LightningComponentBundle>
```

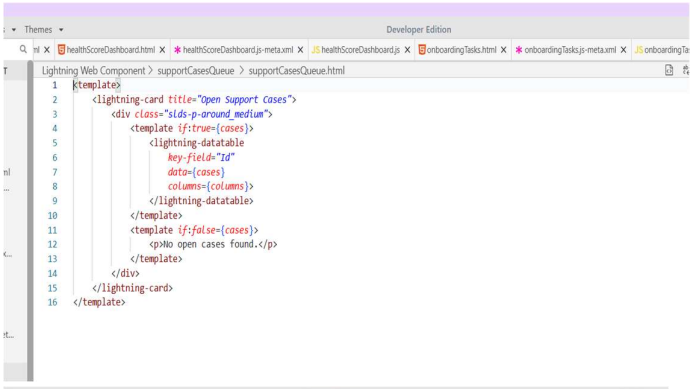
```
<template>
    <lightning-card title="Onboarding Tasks">
        <div class="slds-p-around_medium">
            <template if:true={tasks}>
                <template for:each={tasks} for:item="task">
                    <div key={task.Id} class="slds-box slds-m-bottom_small slds-grid slds-align-spread">
                        <div>
                            <p>{task.Name}</p>
                            <p>Status: {task.Status__c}</p>
                            <p>Due: {task.Due_Date__c}</p>
                        </div>
                        <lightning-button label="Complete" data-id={task.Id} onclick={handleComplete}></lightning-button>
                    </div>
                </template>
            </template>
            <template if:false={tasks}>
                <p>No pending onboarding tasks.</p>
            </template>
        </div>
    </lightning-card>
</template>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
    </targets>
</LightningComponentBundle>
```



```
Lightning Web Component > onboardingTasks > onboardingTasks.js
import { LightningElement, api, wire } from 'lwc';
import getOnboardingTasks from '@salesforce/apex/OnboardingTasksController.getOnboardingTasks';
import completeTask from '@salesforce/apex/OnboardingTasksController.completeTask';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import { refreshApex } from '@salesforce/apex';

export default class OnboardingTasks extends LightningElement {
    @api recordId;
    tasks;
    wiredTasksResult;

    columns = [
        { label: 'Task Name', fieldName: 'Name' },
        { label: 'Status', fieldName: 'Status__c' },
        { label: 'Due Date', fieldName: 'Due_Date__c', type: 'date' }
    ];

    @wire(getOnboardingTasks, { accountId: '$recordId' })
    wiredTasks(result) {
        this.wiredTasksResult = result;
        if(result.data) this.tasks = result.data;
        if(result.error) this.showToast('Error', result.error.body.message, 'error');
    }

    handleComplete(event) {
        const taskId = event.target.dataset.id;
        completeTask({ taskId })
            .then(() => {
                this.showToast('Success', 'Task marked as completed', 'success');
                return refreshApex(this.wiredTasksResult);
            })
            .catch(error => this.showToast('Error', error.body.message, 'error'));
    }

    showToast(title, message, variant) {
        this.dispatchEvent(new ShowToastEvent({ title, message, variant }));
    }
}
```



```
Lightning Web Component > supportCasesQueue > supportCasesQueue.html
<template>
    <lightning-card title="Open Support Cases">
        <div class="slds-p-around_medium">
            <template if:true={cases}>
                <lightning-datatable
                    key-field="Id"
                    data={cases}
                    columns={columns}>
                </lightning-datatable>
            </template>
            <template if:false={cases}>
                <p>No open cases found.</p>
            </template>
        </div>
    </lightning-card>
</template>
```

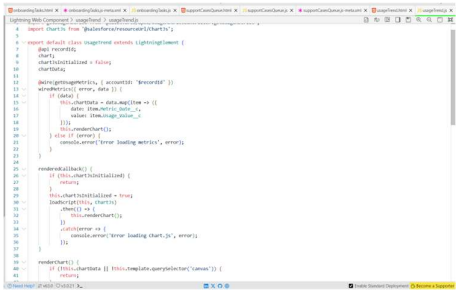
```
Lightning Web Component > supportCasesQueue > supportCasesQueue.js
import { LightningElement, api, wire } from 'lwc';
import getSupportCases from '@salesforce/apex/SupportCasesController.getSupportCases';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import { refreshApex } from '@salesforce/apex';

export default class SupportCasesQueue extends LightningElement {
    @api recordId;
    cases;
    wiredCasesResult;

    columns = [
        { label: 'Case Number', fieldName: 'CaseNumber' },
        { label: 'Subject', fieldName: 'Subject' },
        { label: 'Status', fieldName: 'Status' },
        { label: 'Owner', fieldName: 'OwnerName' }
    ];

    @wire(getSupportCases, { accountId: '$recordId' })
    wiredCases(result) {
        this.wiredCasesResult = result;
        if(result.data) {
            // Map owner name for datatable
            this.cases = result.data.map(c => ({
                ...c, OwnerName: c.Owner.Name
            }));
        } else if(result.error) {
            this.cases = undefined;
            this.showToast('Error', result.error.body.message, 'error');
        }
    }

    showToast(title, message, variant) {
        this.dispatchEvent(new ShowToastEvent({ title, message, variant }));
    }
}
```

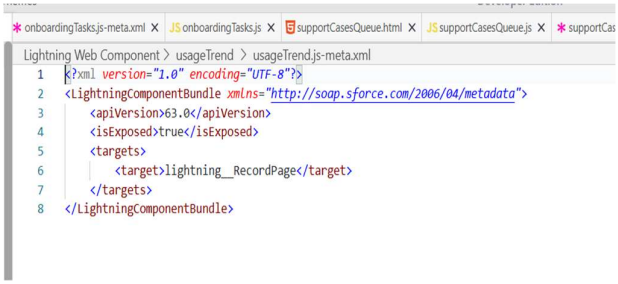
```
Lightning Web Component > supportCasesQueue > supportCasesQueue.js-meta.xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
    </targets>
</LightningComponentBundle>
```

```
public with sharing class SupportCasesController {
    @AuraEnabled(cacheable=true)
    public static List<Case> getSupportCases(Id accountId) {
        return [SELECT Id, CaseNumber, Subject, Status, Owner.Name
                FROM Case
                WHERE AccountId = :accountId AND Status != 'Closed'
                ORDER BY CreatedDate DESC];
    }
}
```

Lightning Web Component 〉 usageTrend 〉 usageTrend.html

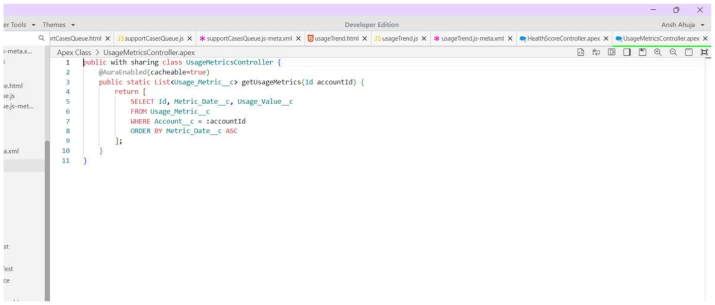
```
<template>
    <lightning-card title="Usage Trend">
        <template if:true={chartData}>
            <canvas class="chart"></canvas>
        </template>
        <template if:false={chartData}>
            <p class="slds-p-around_medium">No usage data available.</p>
        </template>
    </lightning-card>
</template>
```





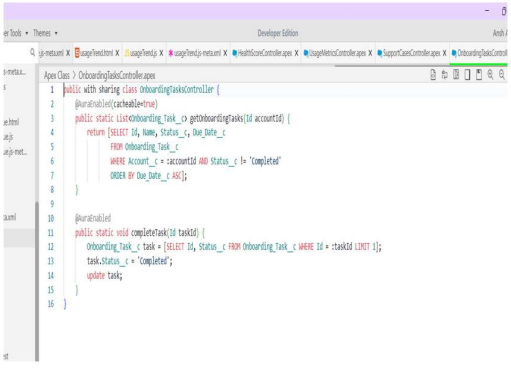
Lightning Web Component 〉 usageTrend 〉 usageTrend.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>63.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
    </targets>
</LightningComponentBundle>
```



Apex Class 〉 UsageMetricsController.apex

```
public with sharing class UsageMetricsController {
    @AuraEnabled(cacheable=true)
    public static List<Usage_Metric__c> getUsageMetrics(Id accountId) {
        return [
            SELECT Id, Metric_Date__c, Usage_Value__c
            FROM Usage_Metric__c
            WHERE Account__c = :accountId
            ORDER BY Metric_Date__c ASC
        ];
    }
}
```



Apex Class 〉 OnboardingTasksController.apex

```
public with sharing class OnboardingTasksController {
    @AuraEnabled(cacheable=true)
    public static List<Onboarding_Task__c> getOnboardingTasks(Id accountId) {
        return [SELECT Id, Name, Status__c, Due_Date__c
                FROM Onboarding_Task__c
                WHERE Account__c = :accountId AND Status__c != 'Completed'
                ORDER BY Due_Date__c ASC];
    }

    @AuraEnabled
    public static void completeTask(Id taskId) {
        Onboarding_Task__c task = [SELECT Id, Status__c FROM Onboarding_Task__c WHERE Id = :taskId LIMIT 1];
        task.Status__c = 'Completed';
        update task;
    }
}
```