Salesforce Project Phase-6

Phase 6: User Interface Development

Lightning App Builder

Drag-and-drop tool to create App, Home, or Record pages.

Types of pages:

App Page: Standalone page in App Launcher.

Home Page: Landing page with dashboards, reports, and components.

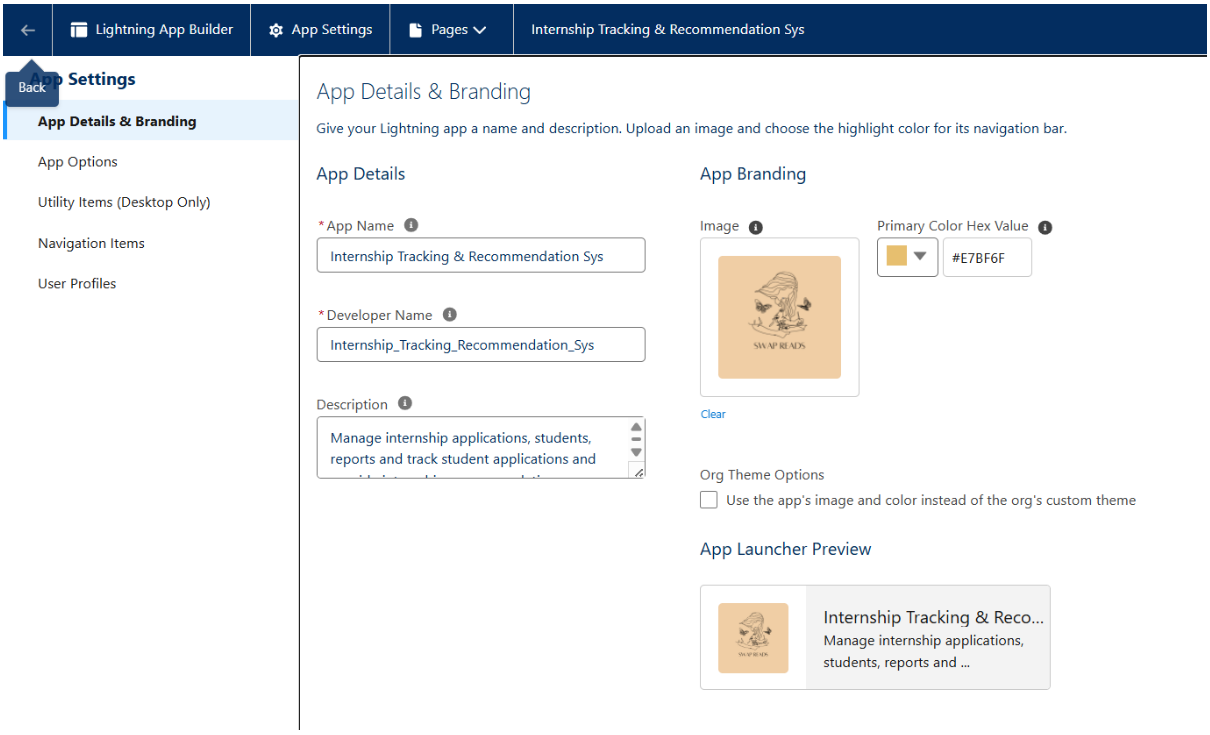
Record Page: Custom layout for specific objects (e.g., Video).

Features:

Dynamic visibility

Tabs for organization

Custom LWC integration



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 Name: Internship Tracking & Recommendation Sys

* Developer Name: Internship_Tracking_Recommendation_Sys

Description: Manage internship applications, students, reports and track student applications and

App Branding

Image

Primary Color Hex Value: #E7BF6F

Org Theme Options

☐ Use the app's image and color instead of the org's custom theme

App Launcher Preview

Internship Tracking & Reco...
Manage internship applications, students, reports and ...

Record Pages :

Customizes layout for **object records**.

Include:

Highlights Panel: Key fields at top.

Tabs: Organize Details, Related Lists, and Custom LWCs.

Assign pages to **profiles or apps**.

3. Tabs

Organize information efficiently.

Examples :

Student

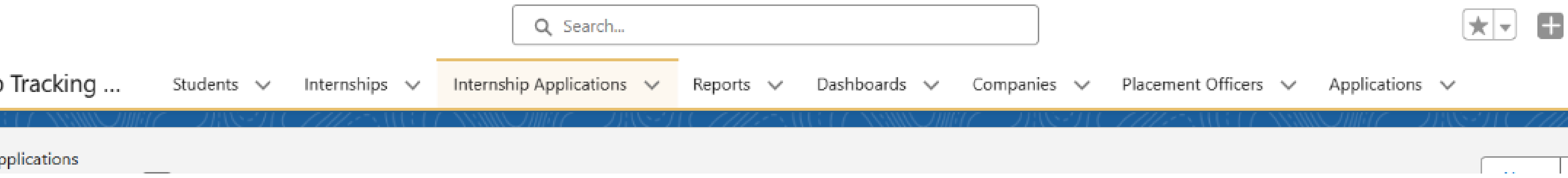
Internship

Internship Application

Reports

Companies

Placement Officer etc.



Home page Layout

The **Home Page** is the first page a user sees when opening your app. Customizing it ensures users like Placement Officers or Students get **relevant information at a glance** — recent applications, charts, instructions, etc.

Components: Reports, Dashboards

Announcements

Custom LWC (e.g., Recent Videos)

Can assign different layouts for different **profiles**.

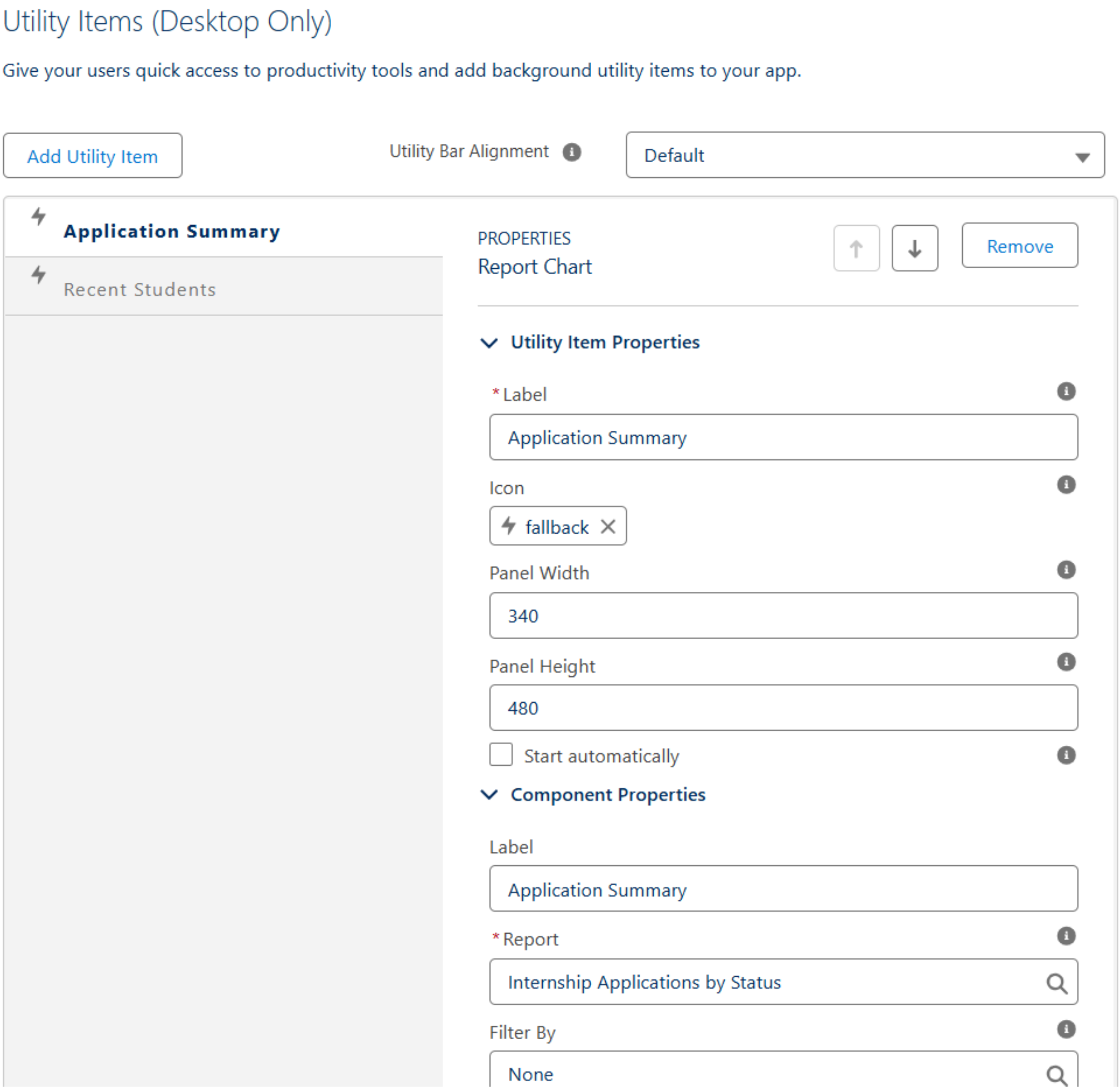
Utility Bar

- Persistent footer in Lightning apps.
- Quick access tools for productivity.

Examples : Application Summary

Recent Added Student

Add via **App Manager → Utility Bar → Add Utility**.



Utility Items (Desktop Only)

Give your users quick access to productivity tools and add background utility items to your app.

Utility Bar Alignment: Default

PROPERTIES
Report Chart

Utility Item Properties

*Label: Application Summary

Icon: fallback

Panel Width: 340

Panel Height: 480

□ Start automatically

Component Properties

Label: Application Summary

*Report: Internship Applications by Status

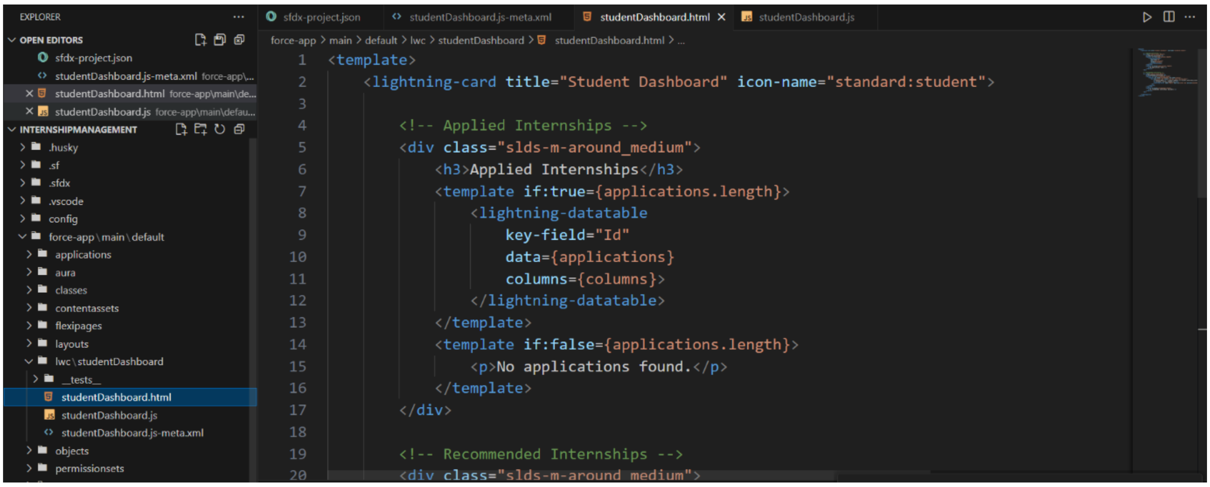
Filter By: None

. LWC (Lightning Web Components)

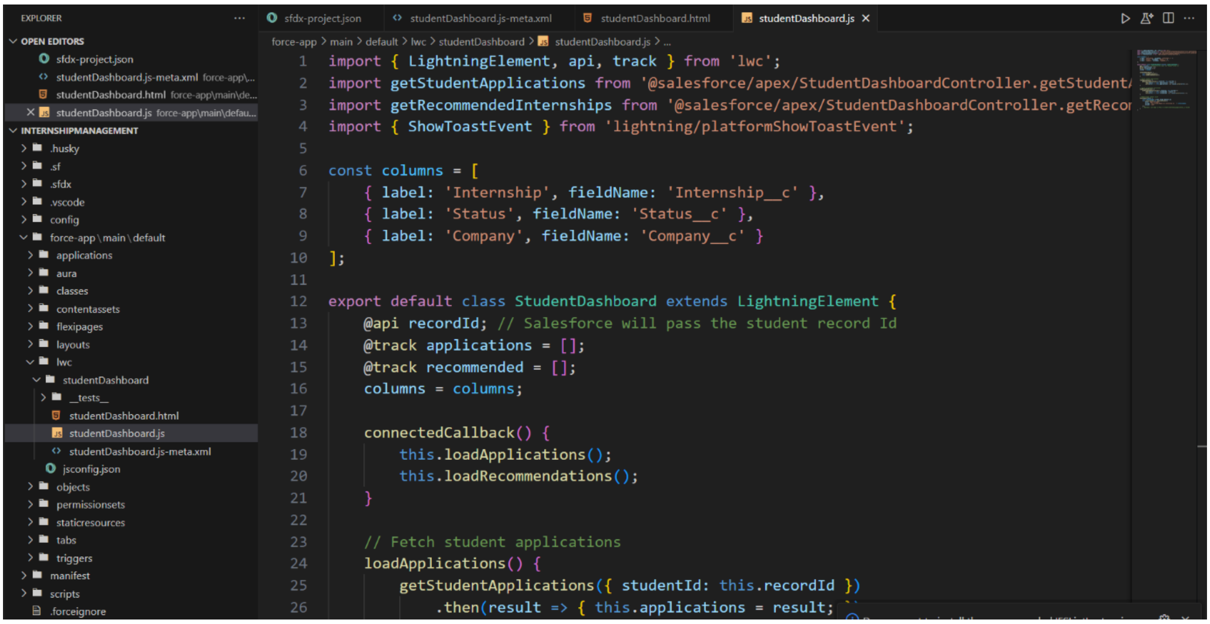
Use Case:
We will create LWCs to provide students and placement officers with an interactive interface. For example, a **"Student Dashboard" LWC** will display applied internships, application status, and recommended opportunities. Similarly, a **"Placement Officer Dashboard" LWC** will list all student applications and their current stages.

1. Student Dashboard

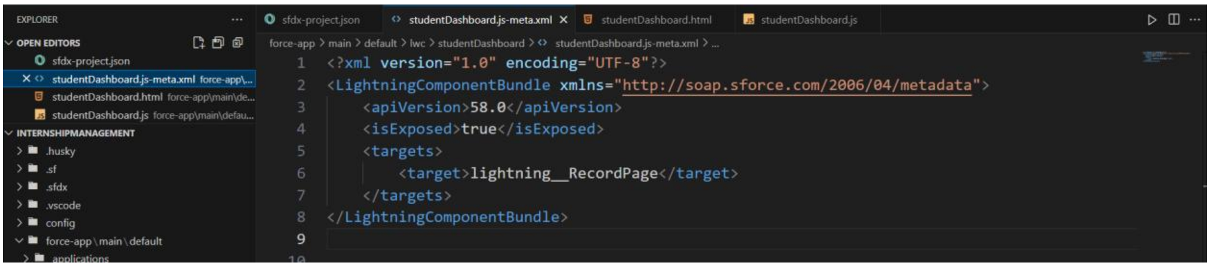
- Shows:
  - Applied internships (Application__c linked to Student)
  - Status of applications
  - Recommended internships (using skill match logic)
- LWC Name: studentDashboard



```html
<template>
    <lightning-card title="Student Dashboard" icon-name="standard:student">

        <!-- Applied Internships -->
        <div class="slds-m-around_medium">
            <h3>Applied Internships</h3>
            <template if:true={applications.length}>
                <lightning-datatable
                    key-field="Id"
                    data={applications}
                    columns={columns}>
                </lightning-datatable>
            </template>
            <template if:false={applications.length}>
                <p>No applications found.</p>
            </template>
        </div>

        <!-- Recommended Internships -->
        <div class="slds-m-around_medium">
```



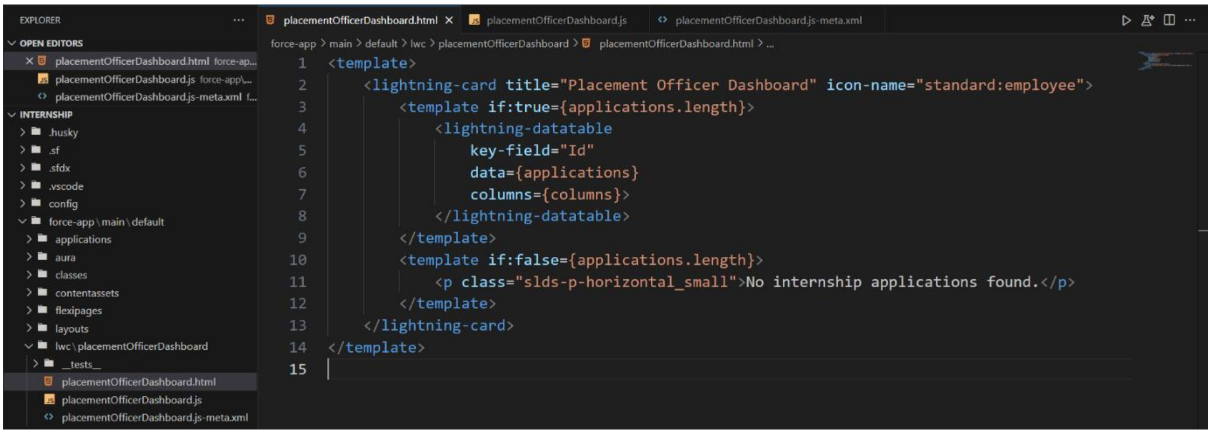
```javascript
import { LightningElement, api, track } from 'lwc';
import getStudentApplications from '@salesforce/apex/StudentDashboardController.getStudent
import getRecommendedInternships from '@salesforce/apex/StudentDashboardController.getRecor
import { ShowToastEvent } from 'lightning/platformShowToastEvent';

const columns = [
    { label: 'Internship', fieldName: 'Internship__c' },
    { label: 'Status', fieldName: 'Status__c' },
    { label: 'Company', fieldName: 'Company__c' }
];

export default class StudentDashboard extends LightningElement {
    @api recordId; // Salesforce will pass the student record Id
    @track applications = [];
    @track recommended = [];
    columns = columns;

    connectedCallback() {
        this.loadApplications();
        this.loadRecommendations();
    }

    // Fetch student applications
    loadApplications() {
        getStudentApplications({ studentId: this.recordId })
            .then(result => { this.applications = result;
```


EXPLORER
OPEN EDITORS
sfdx-project.json
studentDashboard.js-meta.xml force-app\...
studentDashboard.html force-app\main\de...
studentDashboard.js force-app\main\defau...
INTERNSHIPMANAGEMENT
.husky
.sf
.sfdx
.vscode
config
force-app \ main \ default
applications
sfdx-project.json
studentDashboard.js-meta.xml
studentDashboard.html
studentDashboard.js
force-app > main > default > lwc > studentDashboard > studentDashboard.js-meta.xml > ...

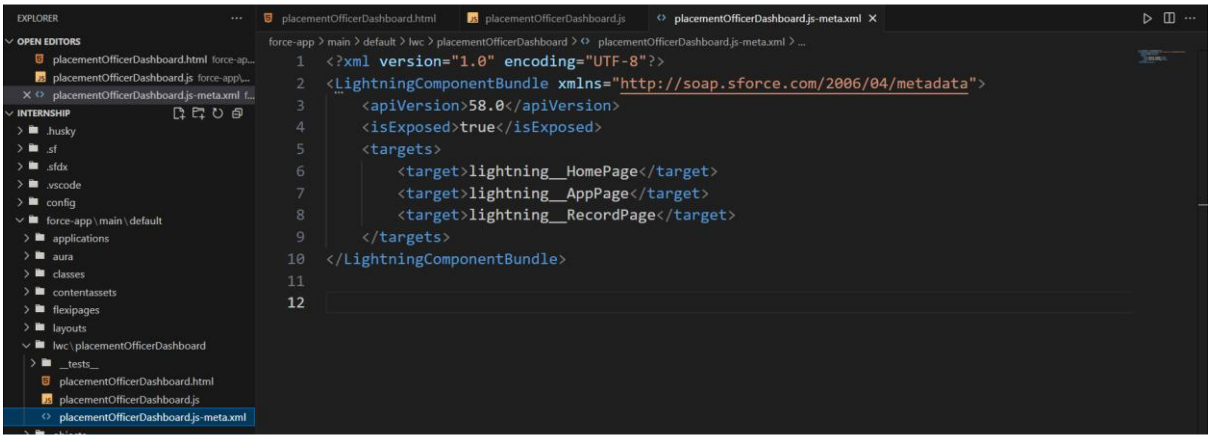
```xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>58.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
    </targets>
</LightningComponentBundle>
```

2. Placement Officer Dashboard

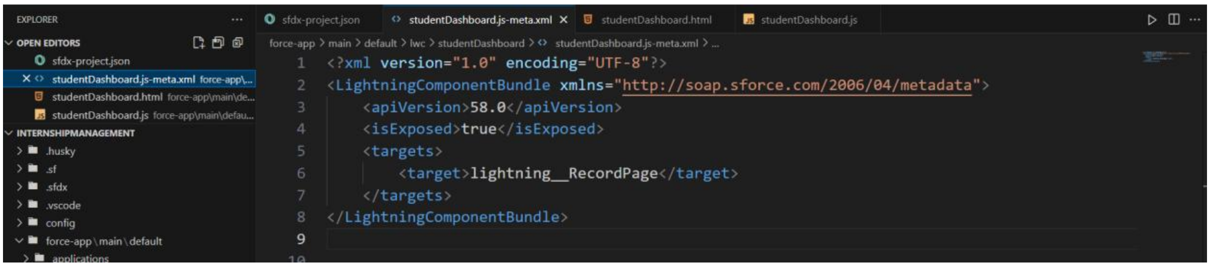
- Shows:
    - All student applications
    - Current stage/status
    - Option to update status (Optional: inline edit or quick action)
- LWC Name: placementOfficerDashboard


EXPLORER
OPEN EDITORS
placementOfficerDashboard.html force-ap...
placementOfficerDashboard.js force-app\...
placementOfficerDashboard.js-meta.xml f...
INTERNSHIP
.husky
.sf
.sfdx
.vscode
config
force-app \ main \ default
applications
aura
classes
contentassets
flexipages
layouts
lwc \ placementOfficerDashboard
__tests__
placementOfficerDashboard.html
placementOfficerDashboard.js
placementOfficerDashboard.js-meta.xml
force-app > main > default > lwc > placementOfficerDashboard > placementOfficerDashboard.html > ...

```html
<template>
    <lightning-card title="Placement Officer Dashboard" icon-name="standard:employee">
        <template if:true={applications.length}>
            <lightning-datatable
                key-field="Id"
                data={applications}
                columns={columns}>
            </lightning-datatable>
        </template>
        <template if:false={applications.length}>
            <p class="slds-p-horizontal_small">No internship applications found.</p>
        </template>
    </lightning-card>
</template>
```


EXPLORER
OPEN EDITORS
placementOfficerDashboard.html force-ap...
placementOfficerDashboard.js force-app\...
placementOfficerDashboard.js-meta.xml f...
INTERNSHIP
.husky
.sf
.sfdx
.vscode
config
force-app \ main \ default
applications
aura
classes
contentassets
flexipages
layouts
lwc \ placementOfficerDashboard
__tests__
placementOfficerDashboard.html
placementOfficerDashboard.js
placementOfficerDashboard.js-meta.xml
force-app > main > default > lwc > placementOfficerDashboard > placementOfficerDashboard.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__HomePage</target>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
    </targets>
</LightningComponentBundle>
```



EXPLORER
OPEN EDITORS
sfdx-project.json
studentDashboard.js-meta.xml force-app\...
studentDashboard.html force-app\main\de...
studentDashboard.js force-app\main\defau...
INTERNSHIPMANAGEMENT
.husky
.sf
.sfdx
.vscode
config
force-app \ main \ default
applications
sfdx-project.json
studentDashboard.js-meta.xml
studentDashboard.html
studentDashboard.js
force-app > main > default > lwc > studentDashboard > studentDashboard.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__RecordPage</target>
    </targets>
</LightningComponentBundle>
```

2. Apex With LWC

- Apex handles server-side logic
- Read-only data : use @AuraEnabled(cacheable=true) with wire adapter
- User action : use imperative apex calls for buttons/events