Apex Dependency Admin Guide

Are you tired of manually tracking down dependencies for your Apex classes, triggers, pages, components, and LWCs? Say goodbye to the hassle with **Apex Dependency**!

Our lightweight yet powerful application automatically identifies where your Apex classes and methods are used across the entire Salesforce org—including Lightning Web Components (LWC), Flows, Apex Triggers, Apex Pages, Apex Components, and Aura Components.

With Apex Dependency, development, troubleshooting, impact analysis, and maintenance become faster, clearer, and far more efficient.

Key Features:

- **Comprehensive Dependency Analysis:** Apex Dependency delivers a complete analysis of your Apex classes and methods, revealing how they interact with and are referenced by various components throughout your Salesforce org.
- **Flow Occurrence Detection:** Effortlessly identify where Apex methods are used within Salesforce Flows—via Apex Actions or invoked logic—ensuring reliable automation and easier impact analysis.
- **LWC, Aura, Trigger, Page & Component Detection:** Quickly locate occurrences of your Apex methods across:
    - Lightning Web Components (LWC)
    - Aura Components (controllers, helpers, markup)
    - Apex Triggers
    - Apex Pages (Visualforce)
    - Apex Components
    - Other Apex Classes

  This makes debugging, refactoring, and code analysis significantly more efficient.

- **User-Friendly Interface:** The app provides an intuitive, organized interface that helps admins and developers browse dependencies, view detailed usage, and navigate to relevant Salesforce pages with ease—perfect for users of all skill levels.

How It Works:

Search Apex Class: Enter the name of the Apex class you want to analyze in the search bar or browse the full class list directly from the UI.

View Dependencies: Apex Dependency automatically generates a detailed dependency report showing where the selected Apex class or its methods are referenced across the org, including:

- Lightning Web Components (LWC)
- Aura Components
- Apex Triggers
- Apex Pages (Visualforce)
- Apex Components
- Salesforce Flows
- Other Apex Classes

Retrieve Code & Context: For each detected occurrence, you can quickly access the relevant snippet or metadata context—such as LWC JS lines, Trigger code blocks, Flow Apex Action configuration, or Aura helper/controller calls—allowing you to review, analyze, and debug efficiently.

Why Choose Apex Dependency?

Efficiency: Save valuable time and effort by automating the process of dependency analysis for your Apex classes.

Accuracy: Ensure accuracy and consistency in your codebase by identifying and addressing dependencies effectively.

Enhanced Collaboration: Facilitate collaboration among development teams by providing clear visibility into code dependencies and usage.

Streamline your development workflow and optimize your Apex codebase with Apex Dependency. Try it today and experience the difference!

Prerequisites:

Set up Domain: If you haven't already set up a domain for your Salesforce org, you need to create one. Deploy the domain as a Lightning preview, as Lightning preview requires a domain URL. You can create a domain from Setup by navigating to Domain Management.

Enable Enhanced Profile User Interface: From Setup, go to User Management Settings and enable the Enhanced Profile User Interface.

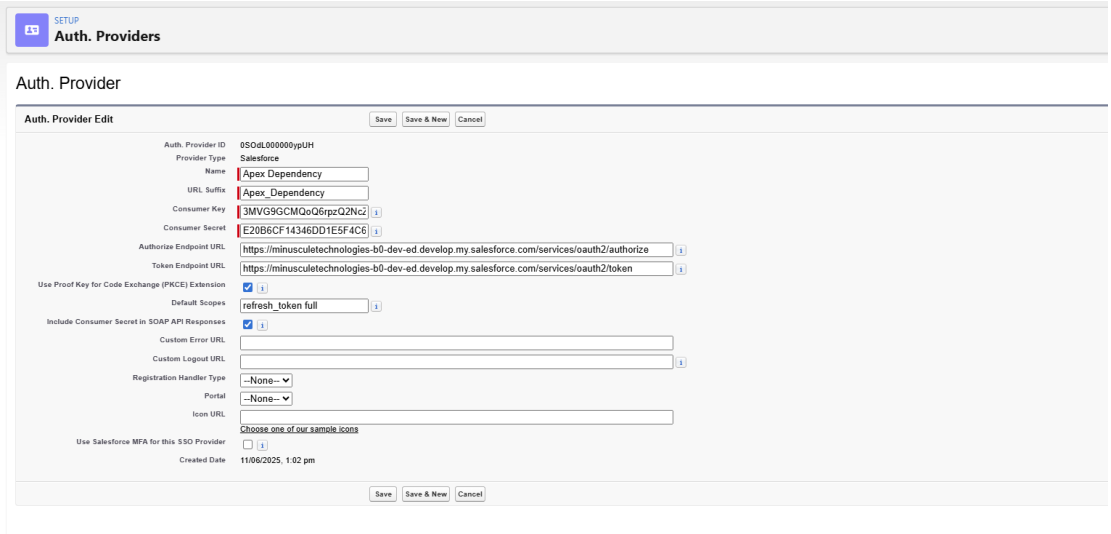
Create a Connected App:

1. Click on Setup → Click on App Manager → Click on New External Client App
2. Give External Client App Name
3. Give Contact Email with the user's email address
4. Enable OAuth Settings under the API section
5. For time being, give the Callback URL as your org domain name from Setup → Domains → Domain Name (My Domain). ex: https://yourdomainname
6. **Select the following OAuth scopes:** Access unique user identifier (openid), Full access (full), Perform requests on your behalf at any time (refresh_token, offline_access), Manage user data via APIs (api)
7. Enable Require Secret for Web Server Flow
8. Click on Create After saving, open the app and copy the **Consumer Key** and **Consumer Secret**.

Create an Authentication Provider:

1. Click on Setup → search for Auth. Providers → click Auth. Providers.
2. Click New.
3. Select **Salesforce** as the Provider Type.
4. Give a Provider Name.
5. Enter Consumer Key.
6. Enter Consumer Secret.
7. Copy your org domain name from Setup → Domains → Domain Name (My Domain). ex: https://yourdomainname
8. **Give Authorize Endpoint URL as:** https://<your-org-domain>.my.salesforce.com/services/oauth2/authorize
9. **Give Token Endpoint URL as:** https://<your-org-domain>.my.salesforce.com/services/oauth2/token
10. **Give Default Scopes as:** refresh_token full
11. Enable Include Consumer Secret in API Responses.
12. Click Save.

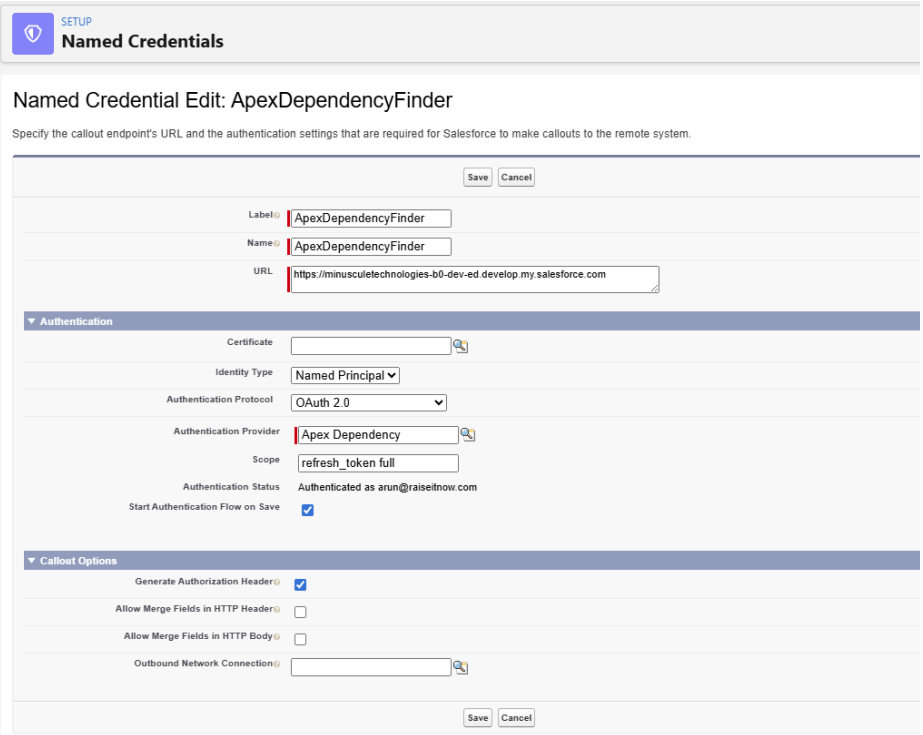
13. **Copy the Callback URL from the saved Auth Provider and update it in the External Client App's Callback URL section (**i.e. https://yourdomainname**). This step is required.**



Create a Named Credential:

1. Click on Setup → Named Credentials → click the New Down Arrow → select New Legacy.
2. Give Name: **ApexDependencyFinder** (do not use any other name).
3. URL: Use your Domain URL:
4. Go to Setup → Domains → Domain Name (My Domain). Example: https://yourdomainname
5. Identity Type: **Named Principal**
6. Authentication Protocol: **OAuth 2.0**
7. Select the Authentication Provider (the one you created earlier) under the Authentication Provider section
8. Scope: **refresh_token full**
9. Enable Generate Authorization Header.
10. Save the Named Credential.

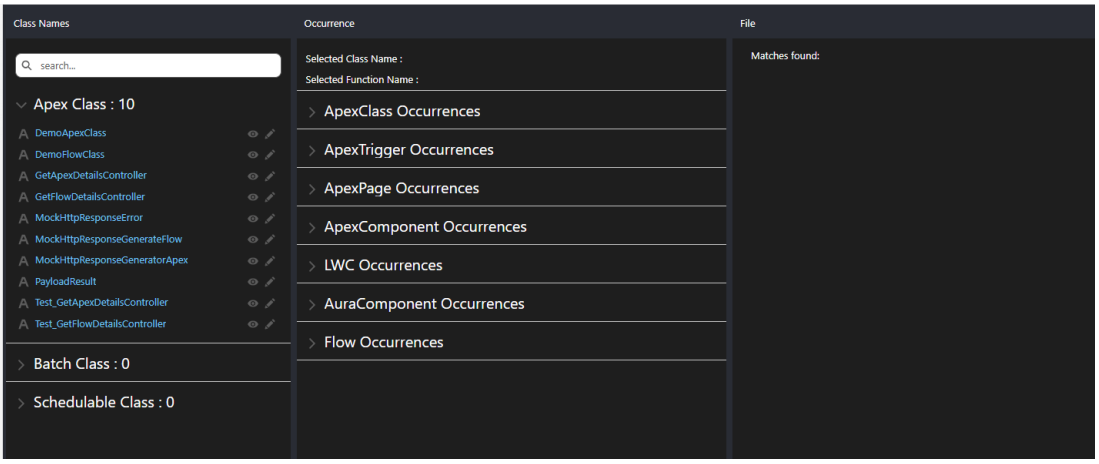
11. After saving, you will be redirected to a login page:
    - Provide your credentials.
    - Accept the permissions and click OK.



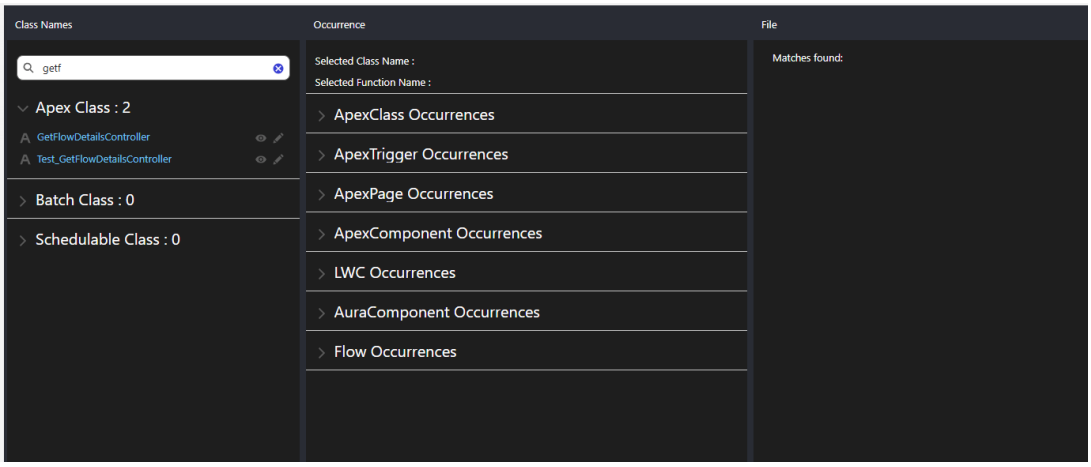
Steps to how to use this Apex Dependency:

Click on App Launcher → Search for Apex Dependency → Select Apex Dependency
It will launch as below

The first function of this app is to display all classes in the org, including Apex classes, Batch classes, and Scheduled classes, along with the count of each class type.



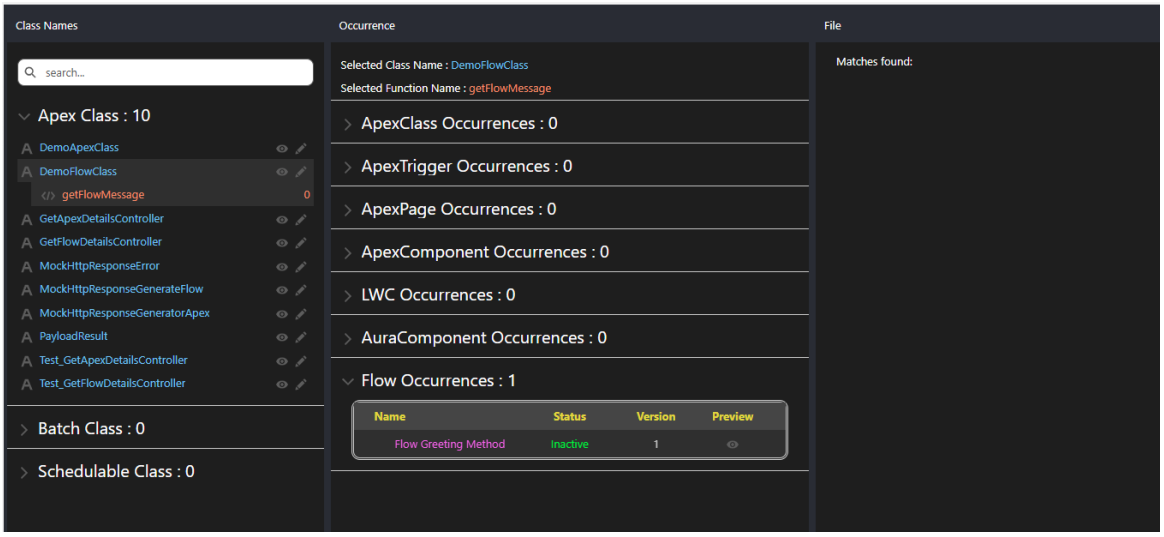
The app simplifies searching for Custom Labels in Salesforce. Users can type any three letters to filter and select the desired Custom Label. Once selected, they can enter the label's name into an input box and click to display relevant information. This functionality streamlines the process of finding specific Custom Labels within the org.



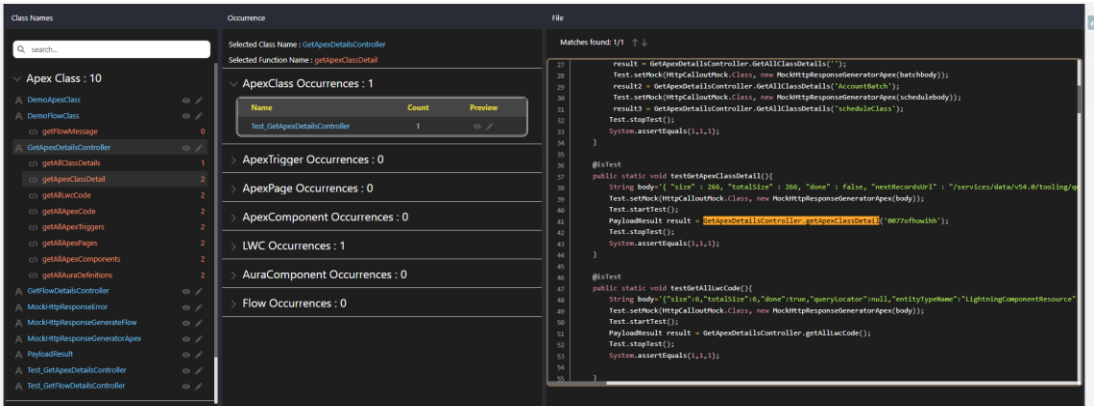
Upon clicking the class name, the app will retrieve the **flow occurrences** of that class and display them in a table. Additionally, the **methods within the class** will be displayed below it. Clicking on a method name will fetch the occurrences in **LWC, Apex, Apex Trigger, Apex Page, Apex Component, and Aura Component** and present them in a table format.

- Icons behind the **class name** allow direct navigation to the **Developer Console** or the **Apex Class page**.
- Clicking the icon in the **Flow Occurrence table** navigates to the **Flow Builder page**.
- Clicking the icon in the **LWC Occurrence table** navigates to the **LWC Bundle page**.

- Clicking the icon in the **Apex Trigger, Apex Page, Apex Component, or Aura Component occurrence tables** navigates to their respective **Salesforce metadata pages**.

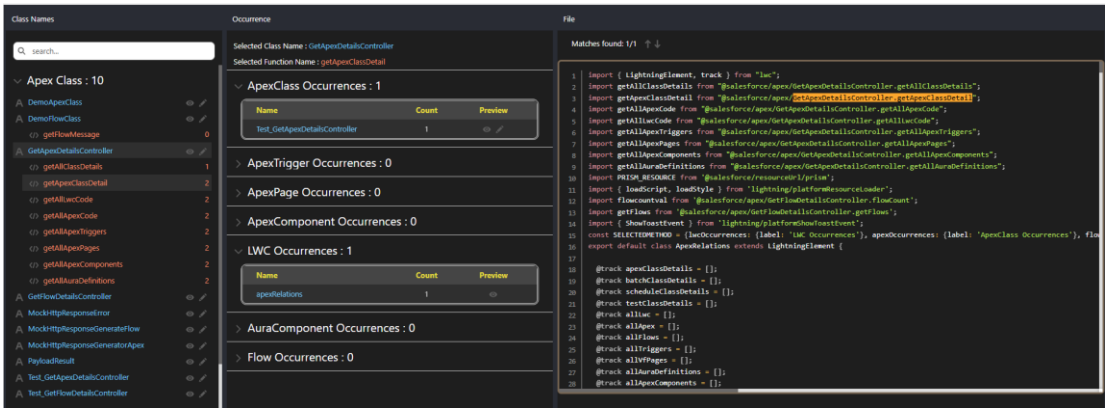


In the **ApexClass Occurrence table**, you can view the count of how many times the clicked method is called within the class. Clicking the class name will retrieve the entire code and display the Apex class within the app page. The app will highlight the occurrences of the method within the class to provide visual context of where the method is called.

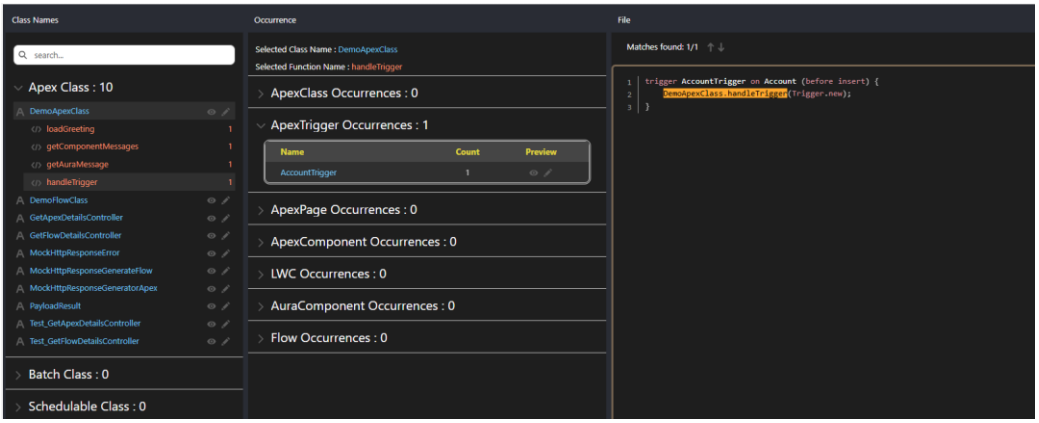


In the **LWC Occurrence table**, you can see the count of how many times the clicked method is called within the LWC code. Clicking the component name will fetch the entire code and display the JavaScript code within the app page. The app will

highlight the occurrences of the method within the JavaScript code to provide visual context of where the method is called.

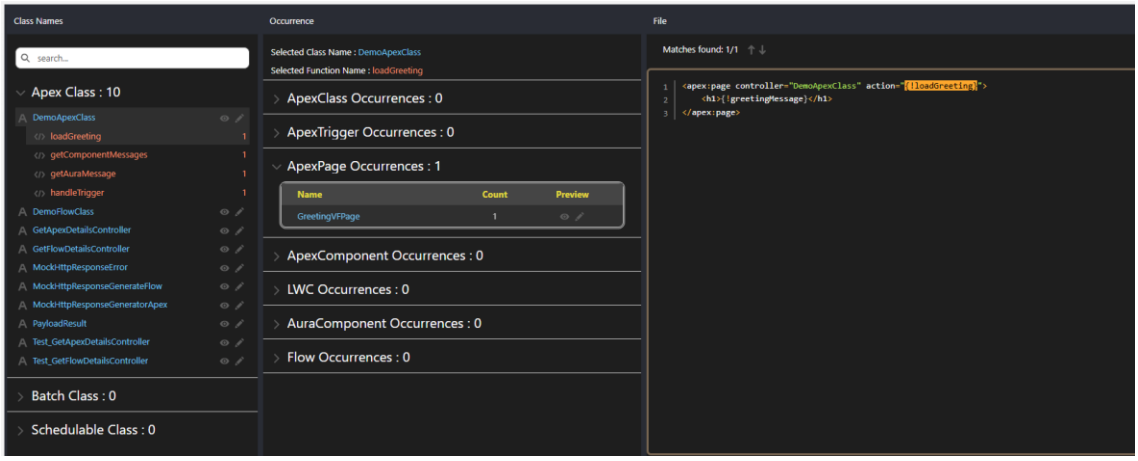


In the **ApexTrigger Occurrence table**, you can view the count of how many times the clicked method is referenced inside Apex triggers. Clicking the **trigger name** will retrieve the full trigger code and display it within the app page. The app will **highlight all occurrences** of the method inside the trigger body to give visual clarity on where the method is invoked.

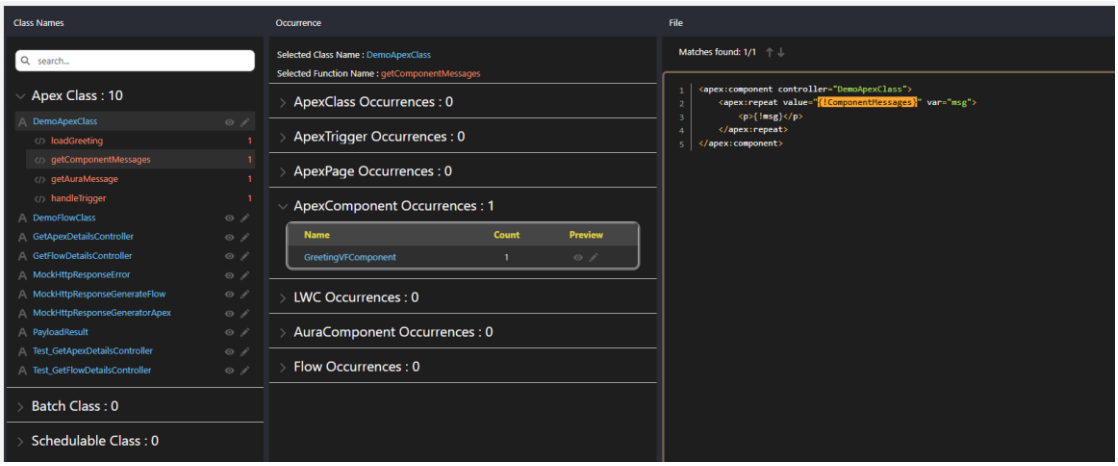


In the **ApexPage Occurrence table**, you can view the number of times the clicked method is referenced inside Visualforce pages. Clicking the **Apex page name** will load the complete Visualforce page markup and display it on the app page. The app will **highlight each method reference** in the page (such as

{!ClassName.MethodName}) so the user can easily identify where the method is used.



In the **ApexComponent Occurrence table**, you can view the total count of how many times the selected method is referenced in Visualforce components. Clicking the **Apex component name** will fetch the component code and display it within the app.The app will **highlight all controller method expressions** (like {!ClassName.MethodName}) inside the component markup to visually indicate where the method is used.



In the **AuraComponent Occurrence table**, you can view the count of how many times the clicked method is referenced inside Aura components. Clicking the **Aura component name** loads its full bundle (component, controller.js, helper.js) into the application's code viewer. The app will **highlight every call** to the Apex method

(e.g., c.methodName in JS controllers) across all bundle files to show precisely where the method is invoked.