Backstage.io Integration

How to add Stack Overflow for Teams search results to your Backstage application.

Document generated 05/09/2025

PDF VERSION

Tags | **Integrations** | **Backstage** |

Applies to: Free Basic Business Enterprise

*This documentation is for **Stack Overflow for Teams Enterprise**. Free, Basic, and Business users can access their documentation here. Find your plan.*

Overview

Backstage.io is a developer portal that streamlines software development by providing developers a common repository of code, libraries, documentation, and other resources. You can connect Backstage to your Stack Overflow for Teams Enterprise site to display questions, tags, and users, as well as ask questions from your Backstage application.

With this Backstage integration, you can unlock your team's collective intelligence directly within backstage. Seamlessly integrate Stack Overflow Enterprise (SOE) to provide developers with instant access to your organization's knowledge hub. Empower them to find answers faster with:

- **Centralized knowledge** A dedicated Stack Overflow HUB showcases the latest questions, trending tags, and top contributors.
- **Contextual insights** Embeddable UI components bring vital Stack Overflow information directly into your Backstage environment.
- **Frictionless collaboration** Secure, streamlined OAuth authentication with PKCE ensures easy access.
- **In-Context questioning** Integration allows developers to ask questions and get help within their Backstage workflow.
- **Unified search** Integrated unified surfaces valuable Stack Overflow knowledge alongside internal documentation, streamlining information discovery.

Quick Start with a Demo Docker Image

If you’d like to quickly experience how this integration works without having to fully set up your own Backstage instance, you can try it out using a pre-built Docker image:

estoesmoises/stackoverflow-backstage-demo:latest

This image runs a Backstage instance already pre-configured with the up-to-date Stack Overflow for Teams plugins and demo setup. All you need to do is pass a few environment variables when starting the container to connect it to your Stack Overflow for Teams Enterprise instance.

Required Environment Variables

Variable	Description
`STACK_OVERFLOW_INSTANCE_URL`	The base URL of your Stack Overflow for Teams (Enterprise) instance.
`STACK_OVERFLOW_API_ACCESS_TOKEN`	A read-only, no-expiry API access token for your instance (used by the search collator to index questions into Backstage search).
`STACK_OVERFLOW_CLIENT_ID`	The OAuth Client ID from your Stack Overflow application (required to enable secure question creation from within Backstage) This ClientID should have write access.
`STACK_OVERFLOW_REDIRECT_URI`	The redirect URI where Stack Overflow should send users after OAuth authentication. Typically `{app.baseUrl}/stack-overflow-teams`. For local use, you can use a redirect service like http://redirectmeto.com/http://localhost:7007/stack-overflow-teams.

Example Command

Once you have those values, you can launch the container by running:

```
docker run -p 7007:7007 \
  -e STACK_OVERFLOW_INSTANCE_URL=https://your-instance-url.stackenterprise.co \
  -e STACK_OVERFLOW_API_ACCESS_TOKEN='your-access-token' \
  -e STACK_OVERFLOW_CLIENT_ID=your-client-id \
  -e STACK_OVERFLOW_REDIRECT_URI=http://redirectmeto.com/http://localhost:7007/stack-overflow-teams \
  estoesmoises/stackoverflow-backstage-demo:latest
```

Integration components

The Backstage SOE integration uses three plugins:

- **Frontend Teams Plugin** Provides the user interface elements that display Stack Overflow content and functionality within your Backstage application.
- **Backend Teams Plugin** Handles API communication between Backstage and your SOE site.
- **Search Collator Plugin (Optional)** Indexes all questions from your SOE site, displaying search results directly within Backstage's search functionality.

This guide details the steps for setting up this integration.

***NOTE:** You'll find it helpful to have a text file or other working document open to copy/paste information as you go through this process.*

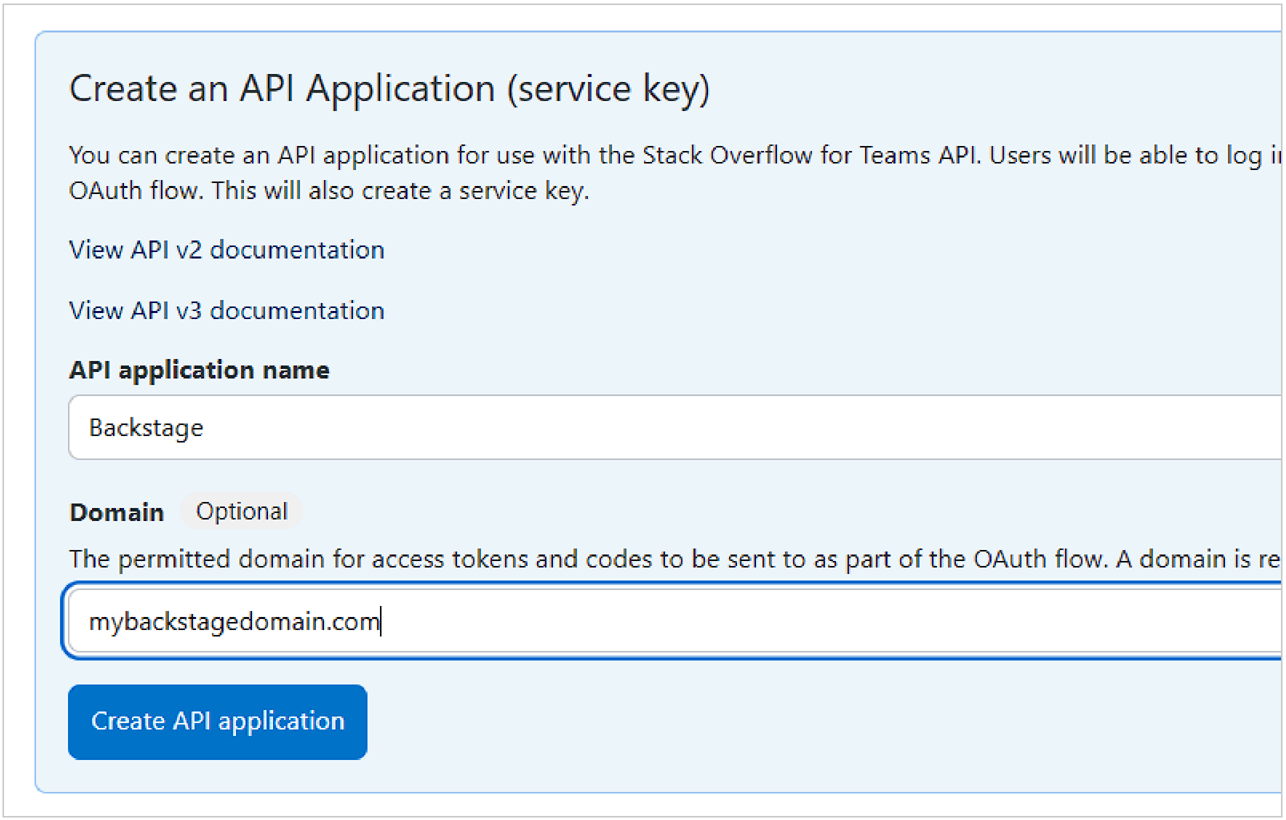
Code example

Our Github repository holds a Backstage SOE integration already configured with the three plugins. If in doubt, use this example to help configure your own site.

Create a service key

To connect your site to the Backstage integration, the best approach is for an SOE administrator to create a service key. Learn more about API applications and SOE service keys in the Stack Overflow for Teams API v3 article.

Other users will see your Backstage service key, so be sure to give it a descriptive name (such as "Backstage"). For the service key domain, specify the domain of your Backstage site.



Create an API Application (service key)

You can create an API application for use with the Stack Overflow for Teams API. Users will be able to log i
OAuth flow. This will also create a service key.

View API v2 documentation

View API v3 documentation

API application name

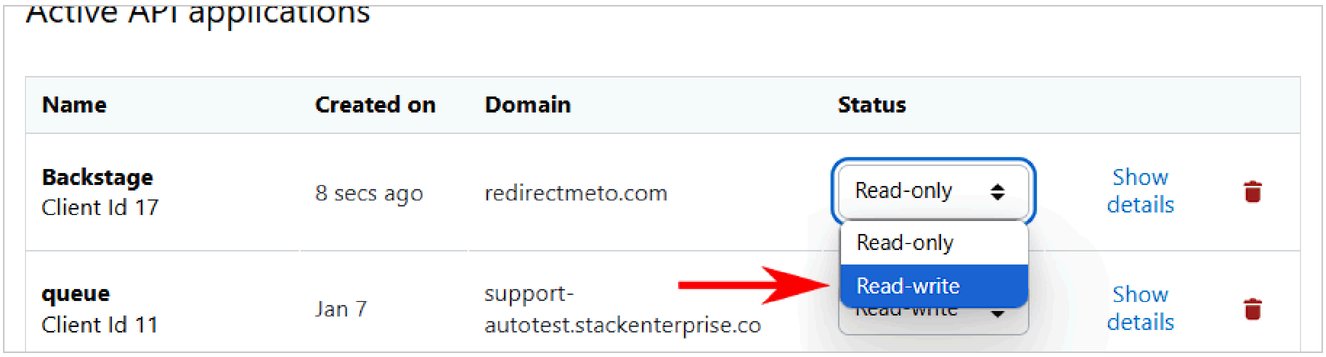
Backstage

Domain Optional

The permitted domain for access tokens and codes to be sent to as part of the OAuth flow. A domain is re

mybackstagedomain.com

Once you've created the service key, refresh the page, change its **Status** to **read-write**, and click **Confirm**. Backstage will need the **client_id**, so be sure to save it to your working document.



Active API applications

Name	Created on	Domain	Status		
Backstage Client Id 17	8 secs ago	redirectmeto.com	Read-only	Show details	
queue Client Id 11	Jan 7	support-autotest.stackenterprise.co	Read-write	Show details	

If you plan to use the optional search collator plugin, you'll need an access token. Generate an access token by following the instructions in the Secure API Token Generation with OAuth and PKCE article. The secure token doesn't require write access, but we recommend enabling the **no_expiry** scope. Copy this token and save it to your working document.

The Backstage backend system

Backstage made changes to how the framework's backend system works with their v1.31.0 release. The instructions in this article assume you're using this New Backend System. You can migrate to the new backend system by following the Migrating your Backend to the New Backend System article.

You can use this integration with the older backend system, but these instructions will not apply. Instead, start by completing the Getting Started with Search guide from Backstage. Then follow the instructions in this Stack Overflow search collator GitHub repository.

NOTE: *The following instructions require the new Backstage backend system.*

Backend installation

1. Add the Stack Overflow for Teams Backend plugin and collator plugin as dependencies in your Backstage application.

   From your Backstage root directory run:

   ```
   yarn --cwd packages/backend add backstage-plugin-stack-overflow-teams-backend
   yarn –cwd packages/backend add backstage-stack-overflow-teams-collator
   ```

   This will add the plugins to the backend of the application.

2. Import the plugins.

   Edit `packages/backend/src/index.ts` and add the Teams Backend plugin and collator:

   ```
   backend.add(import('backstage-plugin-stack-overflow-teams-backend'));
   backend.add(import('backstage-stack-overflow-teams-collator'));
   ```

   NOTE: *You can find an example* `index.ts` *file in our Github repository.*

3. Configure the Backstage application with your Stack Overflow for Teams API credentials.

   For security, you shouldn't hard-code sensitive information (such as passwords or access tokens) into your application. Instead, you should store these in the environment and pass them to your application at runtime. How you'll accomplish this depends on your application and environment configuration. What follows is an example using the dotenv library to safely store and pass credentials.

   Install dotenv
   Run the following command to add the dotenv library to your backend:

   ```
   yarn --cwd packages/backend add dotenv
   ```

   Configure dotenv
   Edit the `packages/backend/src/index.ts` file to load the `.env` variables into the application. Add the following lines of code:

   ```
   import dotenv from 'dotenv'; // Import the dotenv module
   dotenv.config(); // Load environment variables from the .env file into process.env
   ```

   Create and configure the `.env` file
   Create a new `.env` file in the `packages/backend` folder. Define your environment variables in that file:

   ```
   STACK_OVERFLOW_INSTANCE_URL=your_instance_url
   STACK_OVERFLOW_API_ACCESS_TOKEN=your_access_token
   STACK_OVERFLOW_CLIENT_ID=your_clientid
   ```

Ensure git doesn't upload `.env`

Add `.env` to your `.gitignore` file to prevent git from pushing it to the server:

```
# Environment Variables
.env
```

Load environment variables

Your `app-config.yaml` should contain the following lines:

```
  stackoverflow:
    baseUrl: ${STACK_OVERFLOW_INSTANCE_URL}
    teamName: ${STACK_OVERFLOW_TEAM_NAME} // optional
    apiAccessToken: ${STACK_OVERFLOW_API_ACCESS_TOKEN}
    client_id: ${STACK_OVERFLOW_CLIENT_ID}
    redirect_uri: ${STACK_OVERFLOW_REDIRECT_URI} // this should navigate to /stack-overflow-teams i
```

***NOTE:** You can find an example `app-config.yaml` file in our Github repository.*

Frontend installation

Set up search

Install the frontend plugin to use the integration functionality and format the indexed questions. As a prerequisite, you'll need to have your `SearchPage.tsx` file ready for modifications. For more information, read the Backstage Getting Started with Search guide.

In this example, we'll obtain the indexed search results and render them conditionally.

***NOTE:** If you created your application by using `npx @backstage/create-app`, you'll already have a search page defined at `packages/app/src/components/search`. You can simply edit that file.*

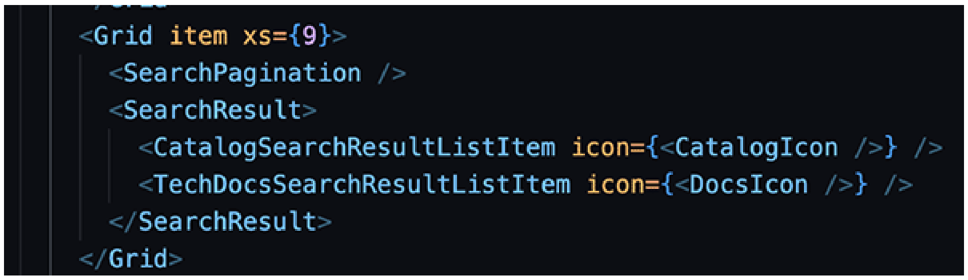
First, install the frontend Stack Overflow plugin. Run:

```
yarn --cwd packages/app add backstage-plugin-stack-overflow-teams
```

Edit `packages/app/src/components/search/SearchPage.tsx`.

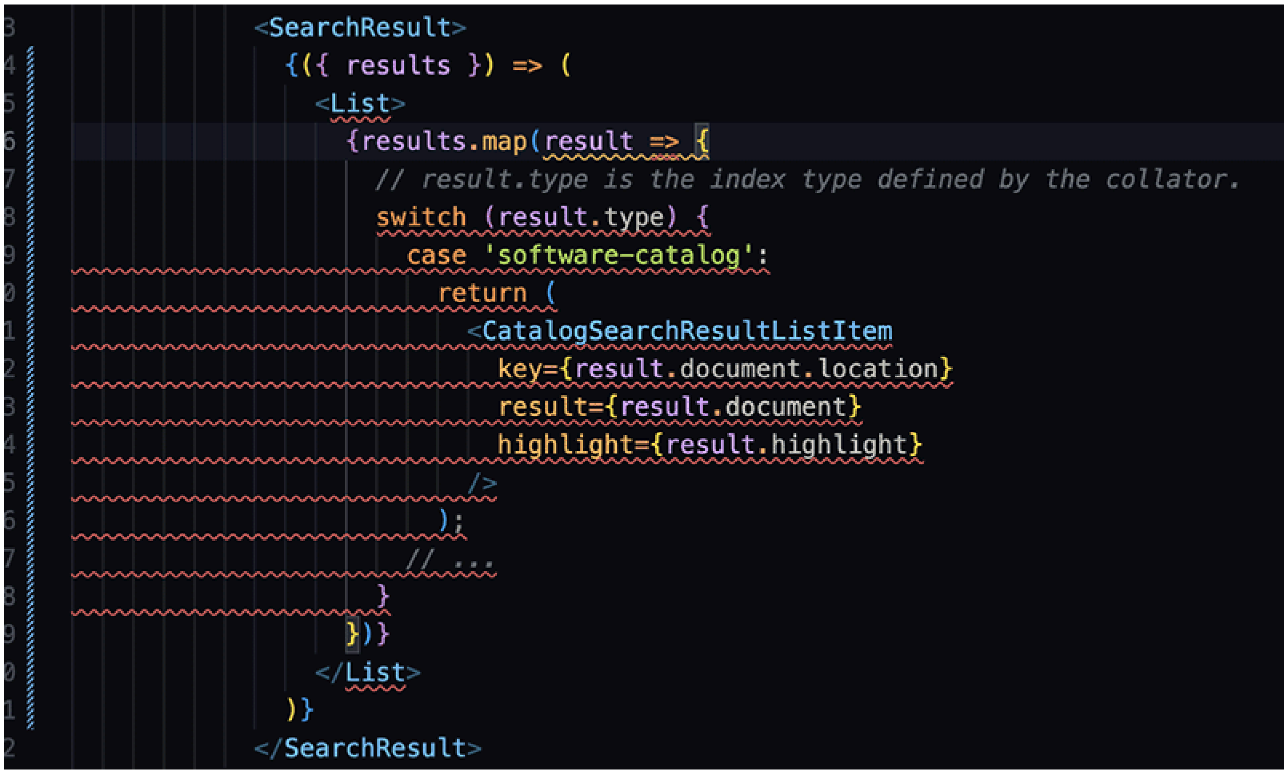
By default, `SearchPage.tsx` displays the search results without conditional formatting. You'll modify this to let the type of returned search results control how the search results appear.

Locate the following section in the existing `<SearchResult><SearchResult/>` code:



```
<Grid item xs={9}>
  <SearchPagination />
  <SearchResult>
    <CatalogSearchResultListItem icon={<CatalogIcon />} />
    <TechDocsSearchResultListItem icon={<DocsIcon />} />
  </SearchResult>
</Grid>
```

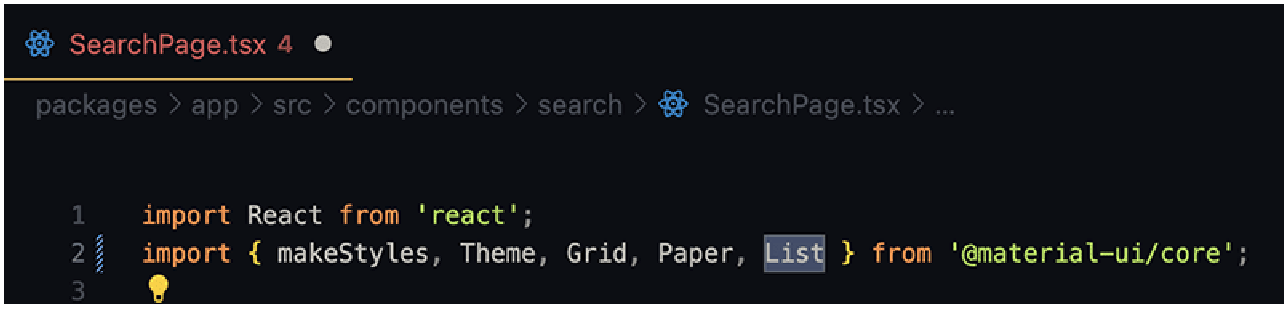
Replace this code with the snippet found in the Backstage Customizing Search documentation as below:



```
<SearchResult>
  {({ results }) => (
    <List>
      {results.map(result => {
        // result.type is the index type defined by the collator.
        switch (result.type) {
          case 'software-catalog':
            return (
              <CatalogSearchResultListItem
                key={result.document.location}
                result={result.document}
                highlight={result.highlight}
              />
            );
          // ...
        }
      })}
    </List>
  )}
</SearchResult>
```

Disregard the error indicators—you'll fix them in the following steps.

1. Import the `<List/>` component from the Material UI library by adding it to the `@material-ui/core` import array:



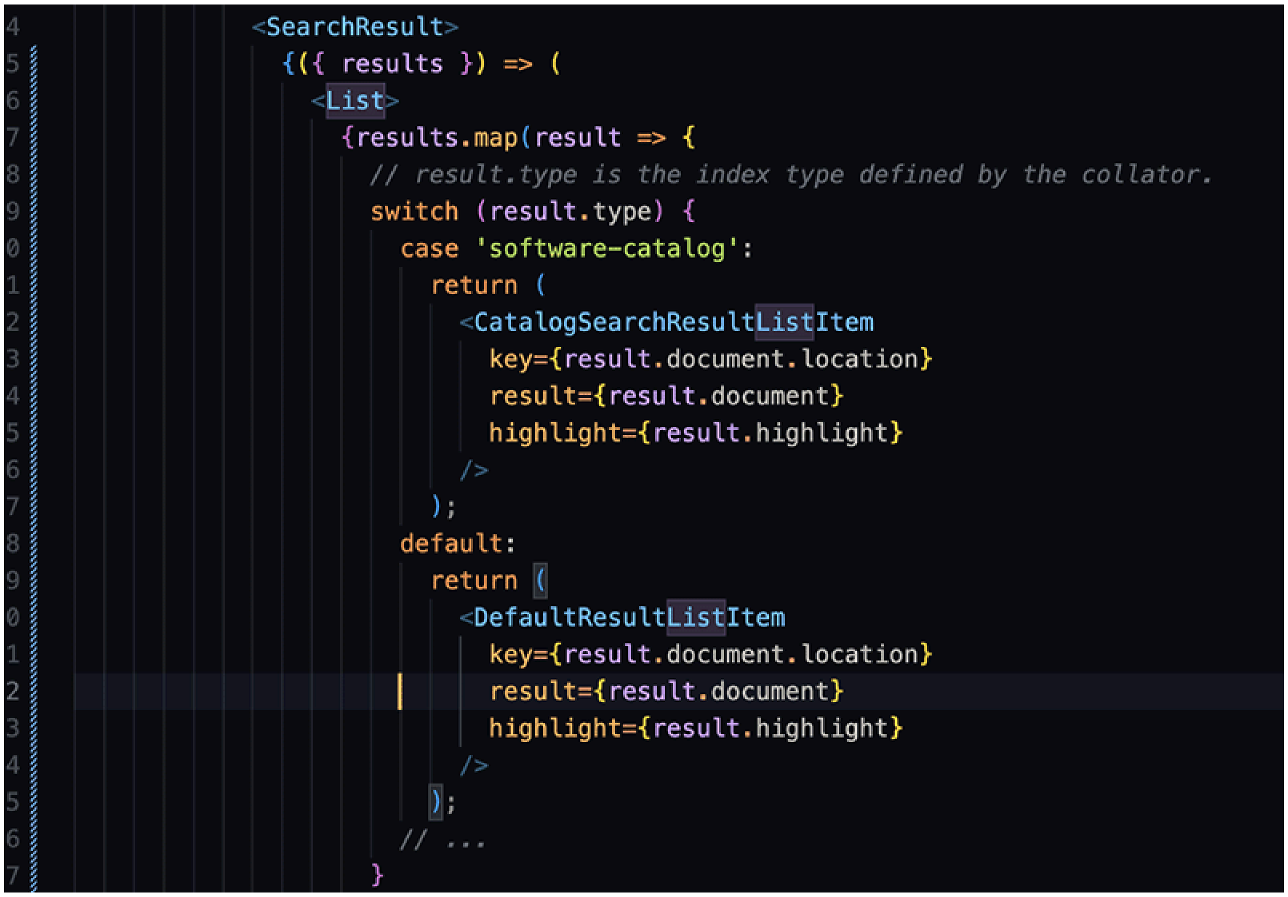
SearchPage.tsx 4

packages > app > src > components > search > SearchPage.tsx > ...

```
import React from 'react';
import { makeStyles, Theme, Grid, Paper, List } from '@material-ui/core';
```

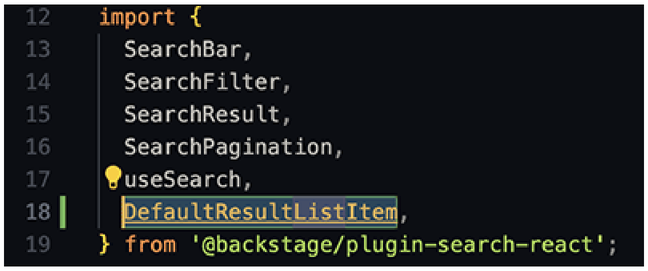
2. Add a default case which uses Backstage's `<DefaultResultListItem/>` component:

```
default:
  return (
    <DefaultResultListItem
      key={result.document.location}
      result={result.document}
      highlight={result.highlight}
    />
```



```
<SearchResult>
  {({ results }) => (
    <List>
      {results.map(result => {
        // result.type is the index type defined by the collator.
        switch (result.type) {
          case 'software-catalog':
            return (
              <CatalogSearchResultListItem
                key={result.document.location}
                result={result.document}
                highlight={result.highlight}
              />
            );
          default:
            return (
              <DefaultResultListItem
                key={result.document.location}
                result={result.document}
                highlight={result.highlight}
              />
            );
          // ...
        }
```

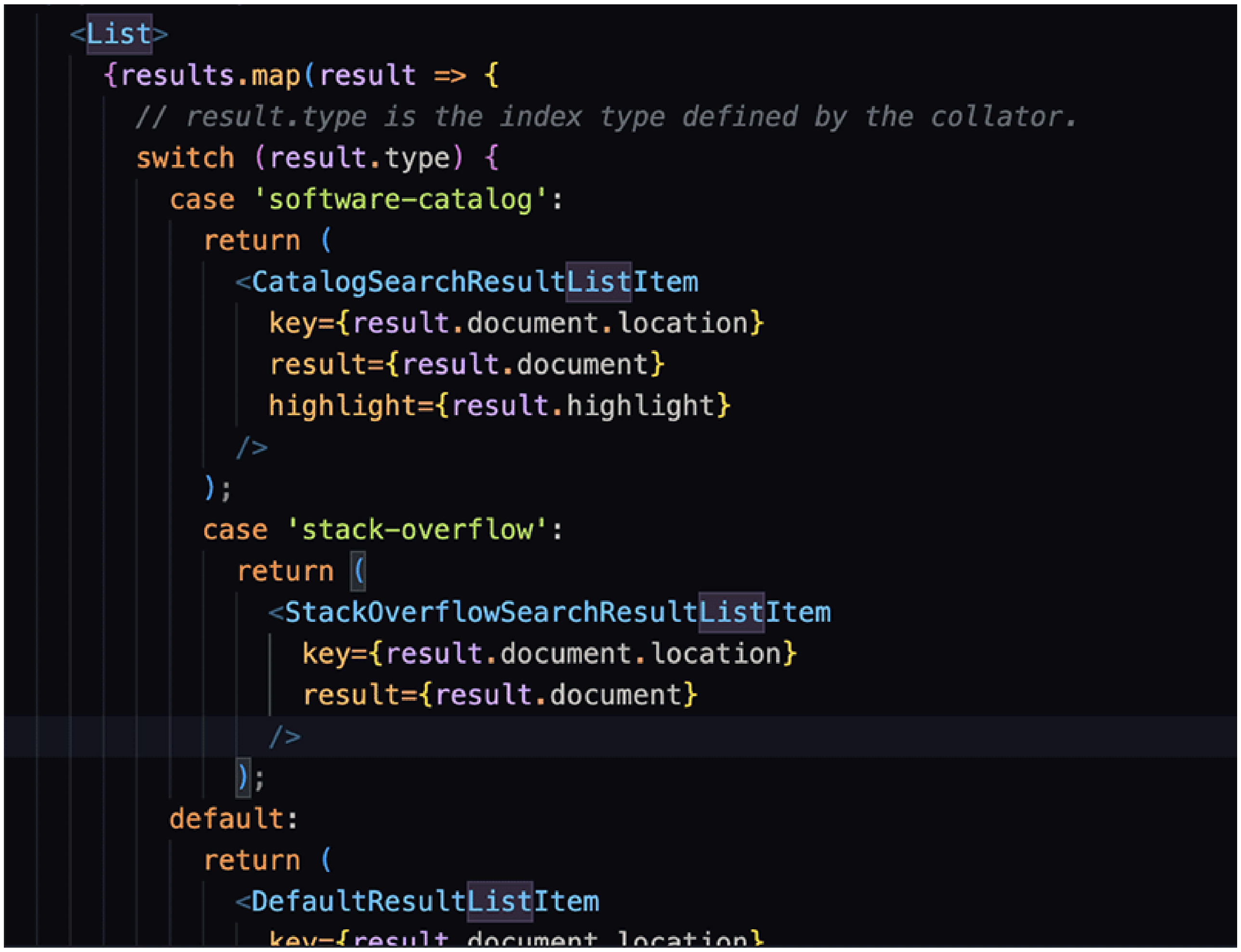
3. Import the `<DefaultResultListItem/>` component:



```
import {
  SearchBar,
  SearchFilter,
  SearchResult,
  SearchPagination,
  useSearch,
  DefaultResultListItem,
} from '@backstage/plugin-search-react';
```

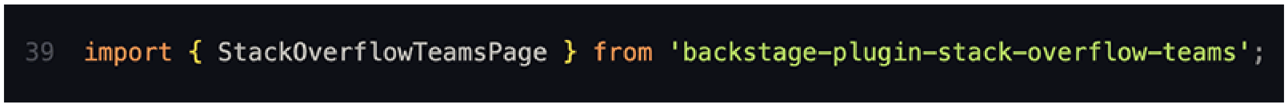
4. Add the StackOverflow case to conditionally render the `<StackOverflowSearchResultListItem />` component from the frontend plugin we installed:

```
case 'stack-overflow':
  return (
    <StackOverflowSearchResultListItem
      key={result.document.location}
      result={result.document}
      highlight={result.highlight}
    />
  );
```



```
<List>
  {results.map(result => {
    // result.type is the index type defined by the collator.
    switch (result.type) {
      case 'software-catalog':
        return (
          <CatalogSearchResultListItem
            key={result.document.location}
            result={result.document}
            highlight={result.highlight}
          />
        );
        case 'stack-overflow':
          return (
            <StackOverflowSearchResultListItem
              key={result.document.location}
              result={result.document}
            />
          );
      default:
        return (
          <DefaultResultListItem
            key={result.document.location}
```

5. Import the list item component from `@backstage-plugin-stack-overflow-teams`:

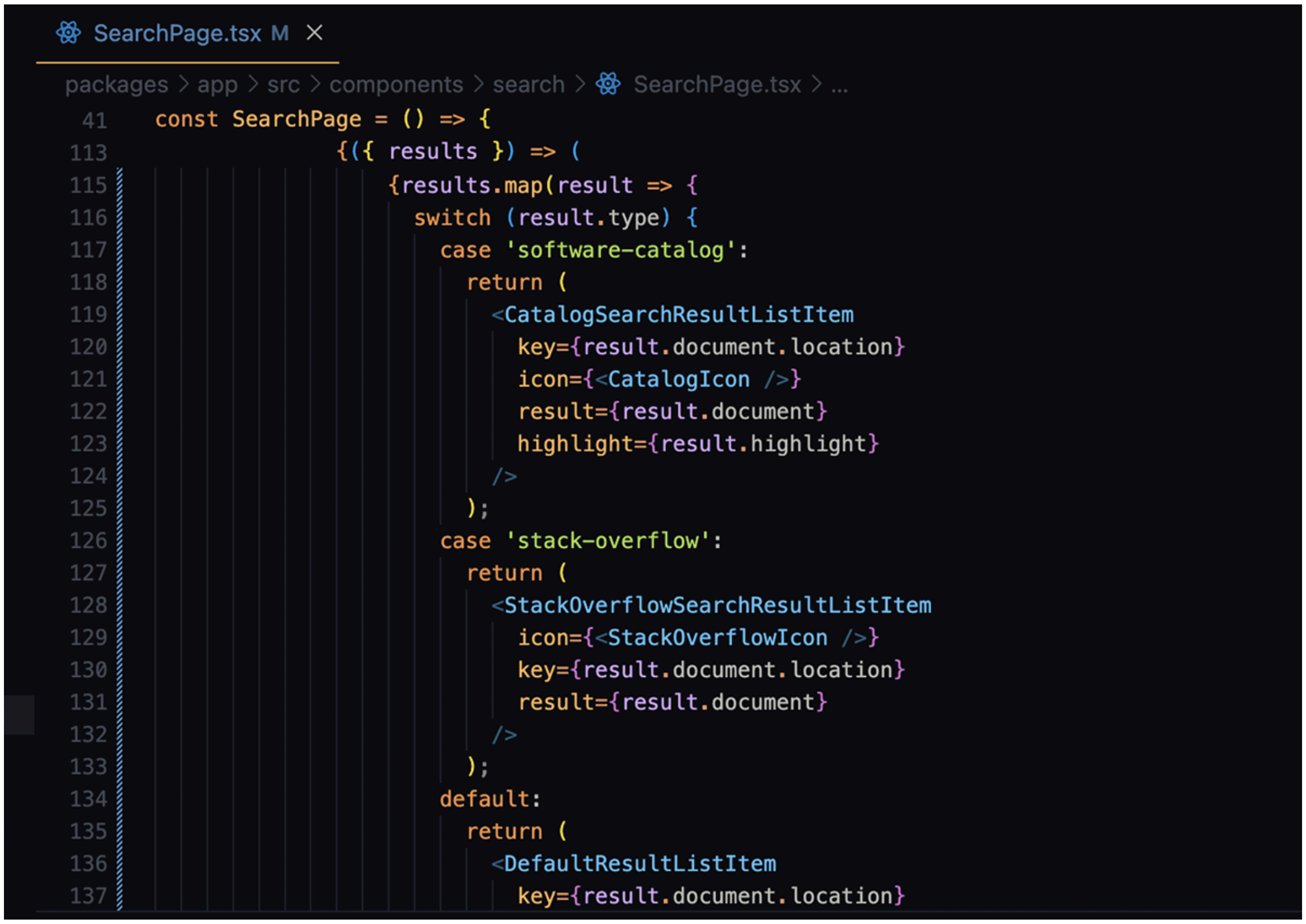


```
import { StackOverflowTeamsPage } from 'backstage-plugin-stack-overflow-teams';
```

6. Add the Stack Overflow results to the Search accordion.

Locate the `<SearchType.Accordion/>` component and add the following code snippet to the accordion:

```
{
  value: 'stack-overflow',
  name: 'Stack Overflow',
  icon: <StackOverflowIcon />,
},
```

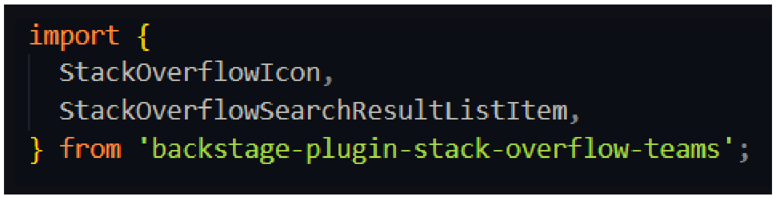


SearchPage.tsx M

packages > app > src > components > search > SearchPage.tsx > ...

```
const SearchPage = () => {
                {({ results }) => (
                  {results.map(result => {
                    switch (result.type) {
                      case 'software-catalog':
                        return (
                          <CatalogSearchResultListItem
                            key={result.document.location}
                            icon={<CatalogIcon />}
                            result={result.document}
                            highlight={result.highlight}
                          />
                        );
                      case 'stack-overflow':
                        return (
                          <StackOverflowSearchResultListItem
                            icon={<StackOverflowIcon />}
                            key={result.document.location}
                            result={result.document}
                          />
                        );
                      default:
                        return (
                          <DefaultResultListItem
                            key={result.document.location}
```

7. Import the `<StackOverflowIcon />` component from `@backstage-plugin-stack-overflow-teams`.



```
import {
  StackOverflowIcon,
  StackOverflowSearchResultListItem,
} from 'backstage-plugin-stack-overflow-teams';
```

Configuration is complete. Before running the application, ensure all dependencies are installed by running `yarn install` on the root directory of your Backstage application.

Example `SearchPage.tsx` code

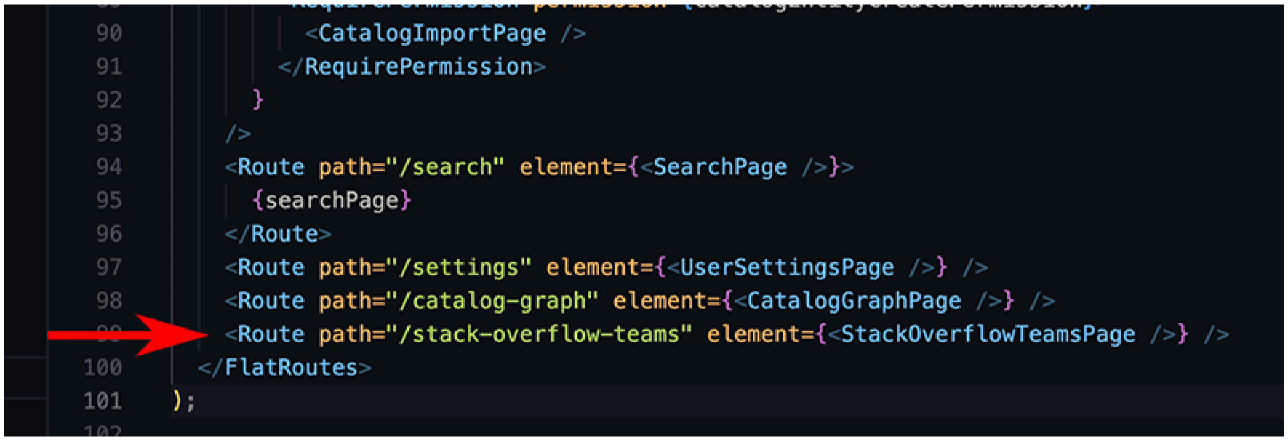
You can find an example of the SearchPage.tsx used in this article in our Github repository.

Set up the Stack Overflow for Teams page and questions modal

The frontend plugin includes the Stack Overflow hub (reusable frontend components) and a modal component you can integrate anywhere in your UI. The hub displays top questions, tags, and users. To incorporate this hub into your Backstage frontend, create a new frontend route and place the route's shortcut within the UI. Additionally, you can add the modal component in a convenient location.

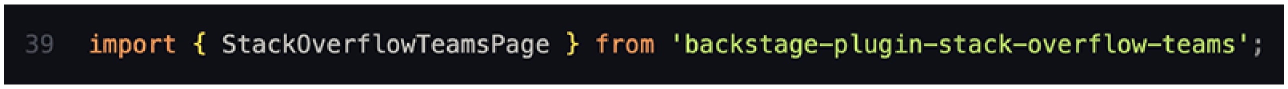
Create a frontend route for the plugin

1. Navigate to your frontend `App.tsx` file, usually found at `packages/frontend/src`. Scroll until you see the `<Route />` elements. Add a route for the StackOverflowTeamsPage component as follows:



```
          <CatalogImportPage />
        </RequirePermission>
      }
    />
    <Route path="/search" element={<SearchPage />}>
      {searchPage}
    </Route>
    <Route path="/settings" element={<UserSettingsPage />} />
    <Route path="/catalog-graph" element={<CatalogGraphPage />} />
    <Route path="/stack-overflow-teams" element={<StackOverflowTeamsPage />} />
  </FlatRoutes>
);
```

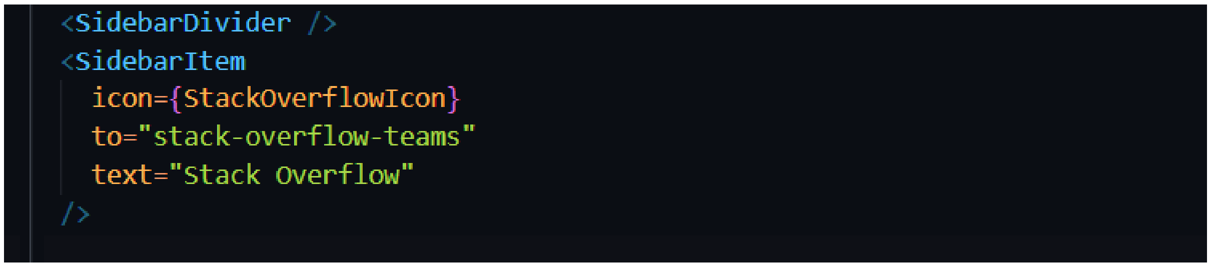
2. Make sure the component is imported from the plugin:



```
import { StackOverflowTeamsPage } from 'backstage-plugin-stack-overflow-teams';
```

3. Create a shortcut for the route (optional). Though not required, we recommend adding this shortcut to the sidebar for convenience.

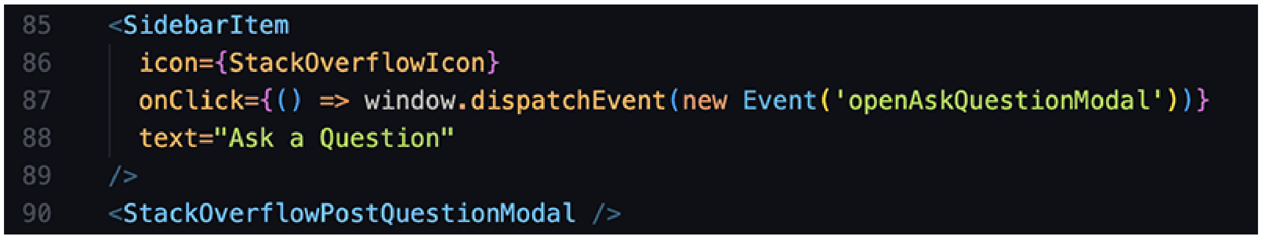
   To add the shortcut, navigate to `packages/app/src/components/Root.tsx`. Scroll down to locate the `SidebarItem` section. Here you'll add a new sidebar item with the Stack Overflow logo. In the UI, clicking the logo will direct users to the route we built earlier.



```
<SidebarDivider />
<SidebarItem
  icon={StackOverflowIcon}
  to="stack-overflow-teams"
  text="Stack Overflow"
/>
```

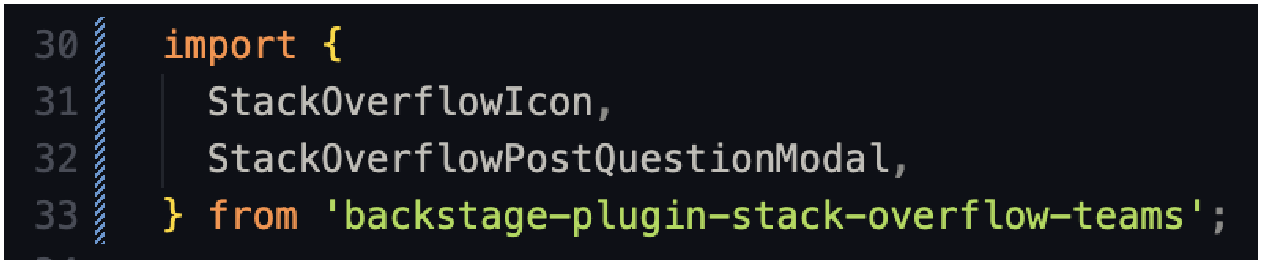
   While you're here, add the ‘Ask a Question’ modal to the sidebar. This makes the button available from anywhere in the Backstage application.

   The `<StackOverflowPostQuestionModal />` listens to the event 'openAskQuestionModal' regardless of where the event is created. Add a new `SidebarItem` to create that event, and also add the `<StackOverflowPostQuestionModal />` as seen below:



```
<SidebarItem
  icon={StackOverflowIcon}
  onClick={() => window.dispatchEvent(new Event('openAskQuestionModal'))}
  text="Ask a Question"
/>
<StackOverflowPostQuestionModal />
```

Get all the resources imported from the plugin:

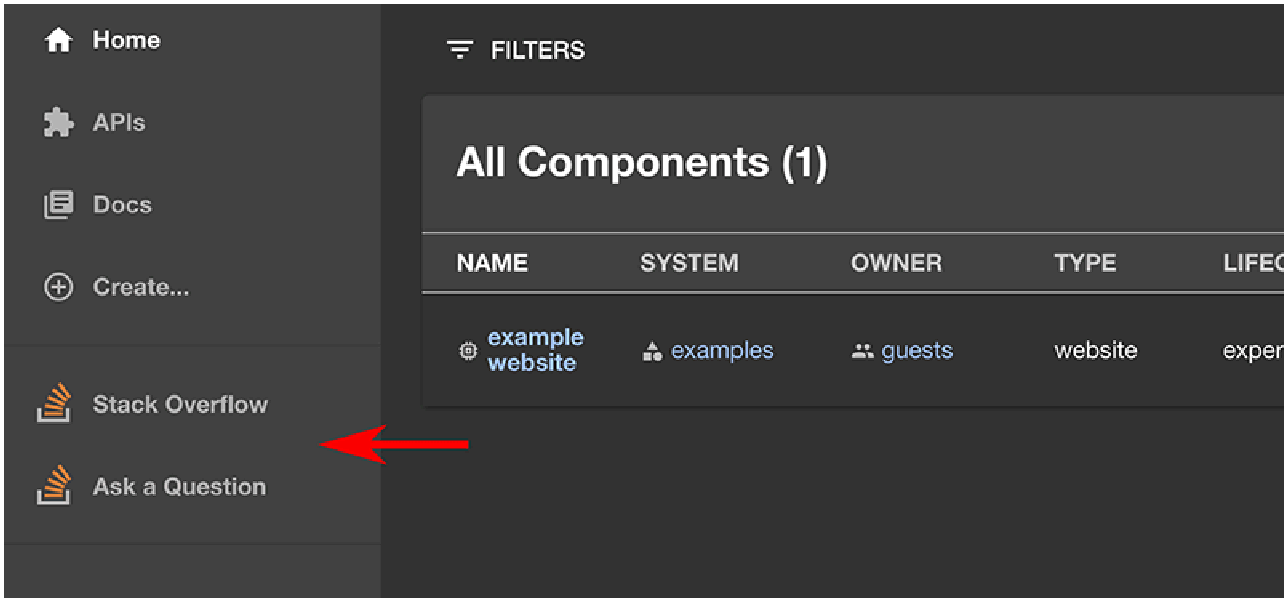


```
import {
  StackOverflowIcon,
  StackOverflowPostQuestionModal,
} from 'backstage-plugin-stack-overflow-teams';
```

UI examples

Below are examples of how the UI should look.

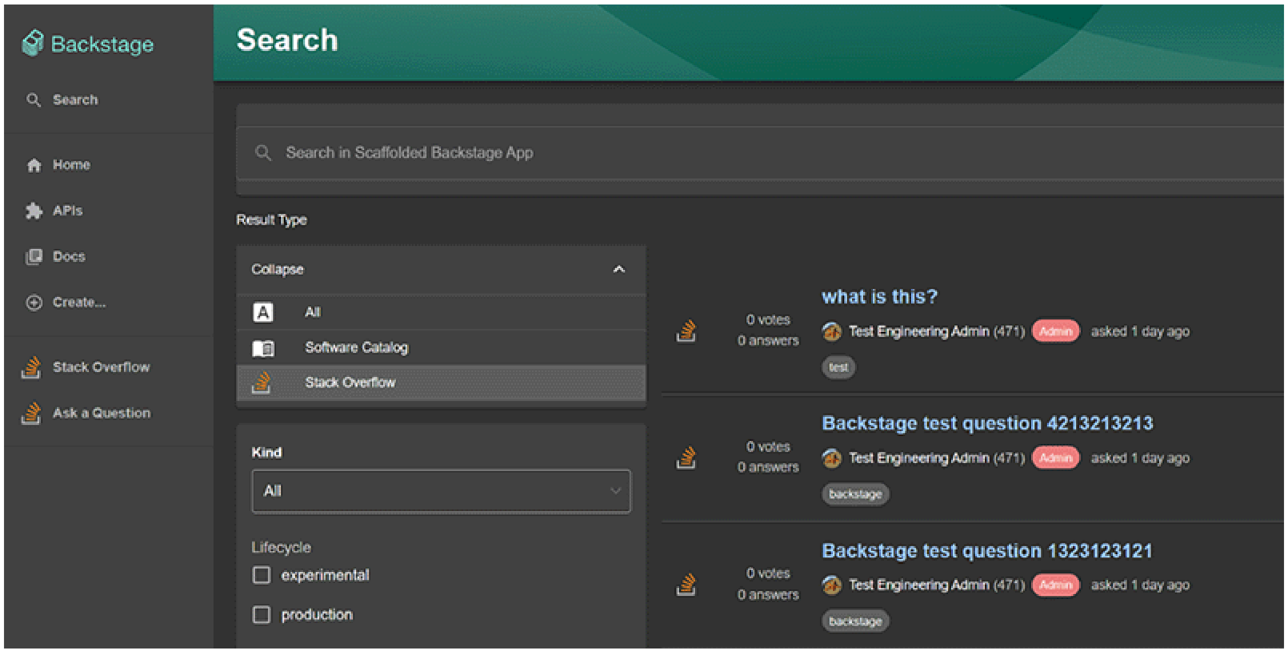
Sidebar



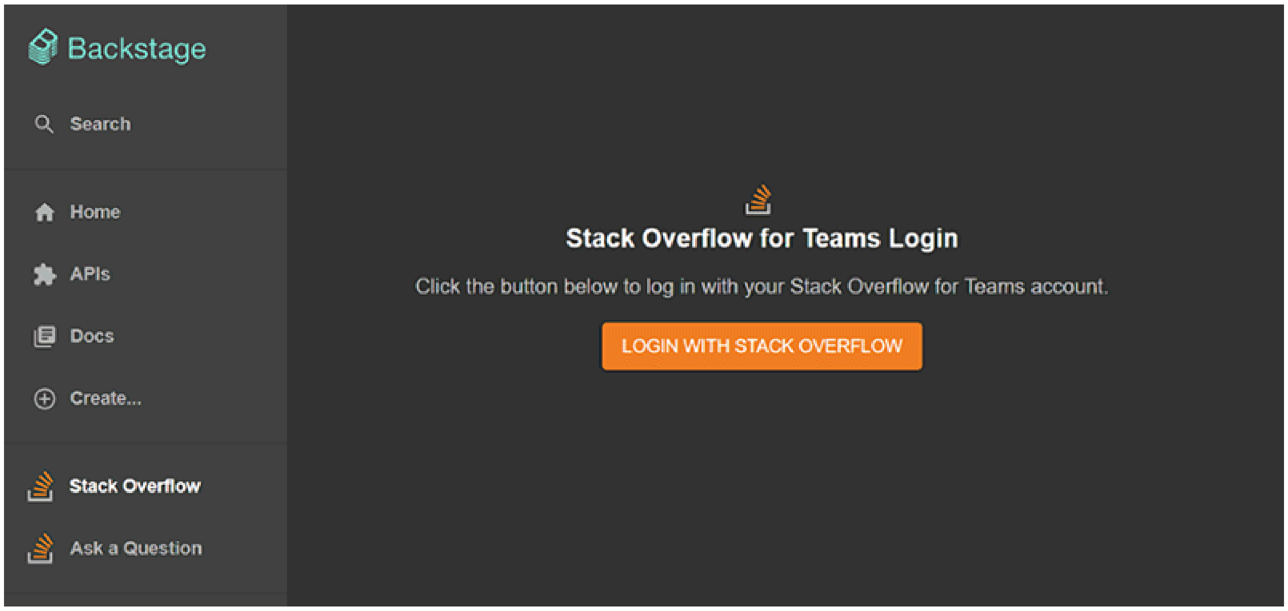
All Components (1)

NAME	SYSTEM	OWNER	TYPE	LIFEC
example website	examples	guests	website	exper

Search page



Stack Overflow login page



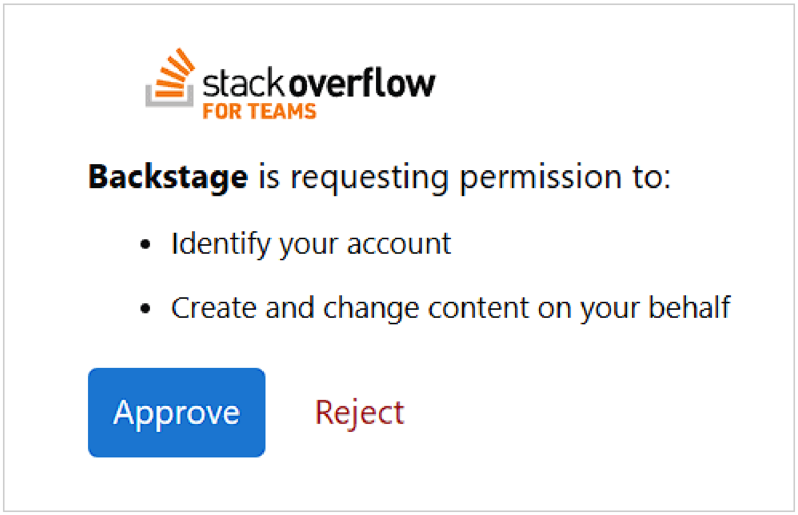
Stack Overflow for Teams Login

Click the button below to log in with your Stack Overflow for Teams account.

Test the integration

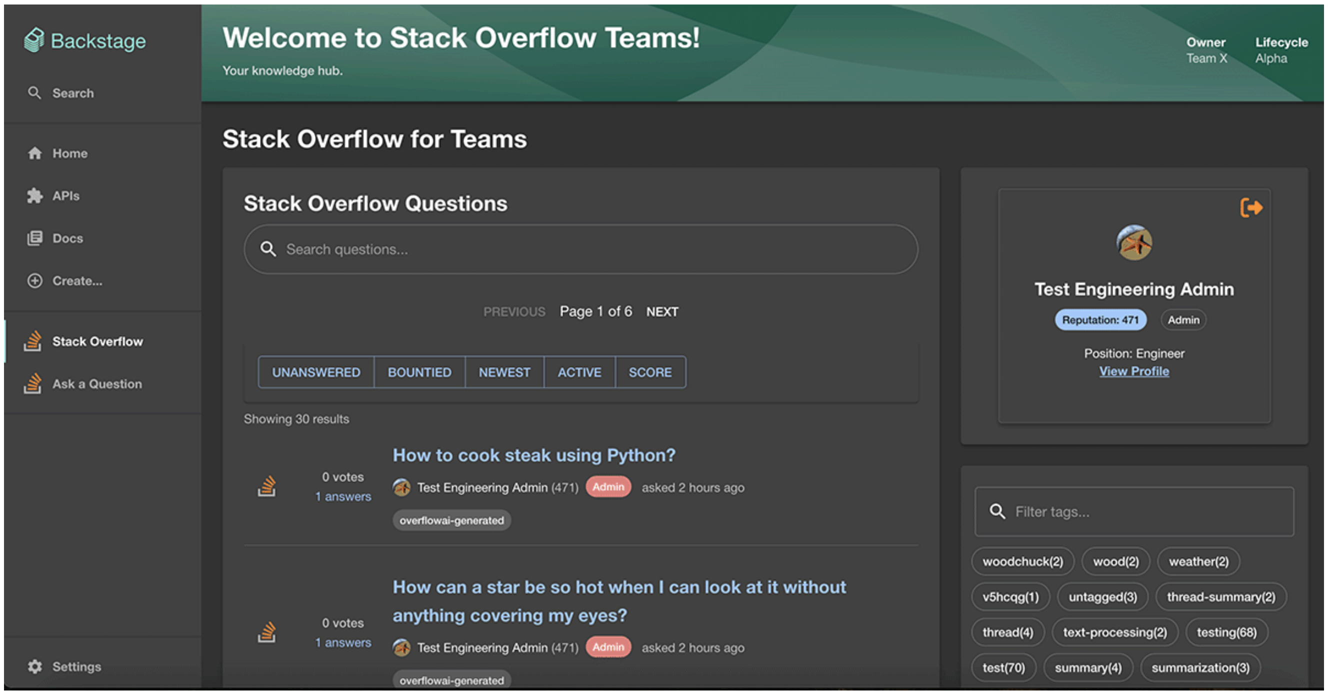
Before running the app, ensure all dependencies are correctly installed by running ‘yarn install’ on the Backstage root directory one last time. After that, run the app and look for any errors.

If the Backstage integration is set up correctly, logging in should redirect you to your Enterprise instance with the following message. Click **Approve**.


Backstage is requesting permission to:

- Identify your account
- Create and change content on your behalf

Once authorized, you should see the Stack Overflow hub.


Welcome to Stack Overflow Teams!

Your knowledge hub.

Stack Overflow for Teams

Stack Overflow Questions

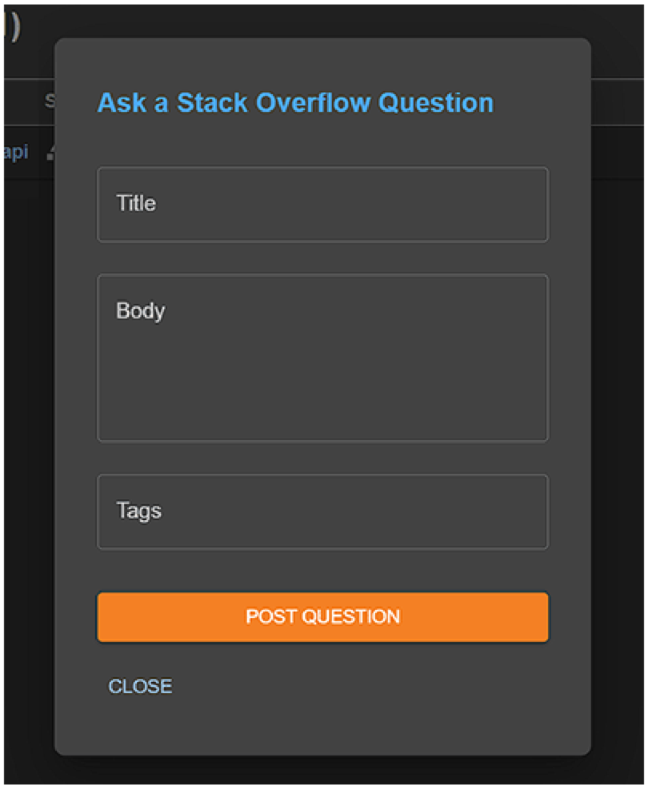
Showing 30 results

How to cook steak using Python?
0 votes · 1 answers · Test Engineering Admin (471) Admin asked 2 hours ago · overflowai-generated

How can a star be so hot when I can look at it without anything covering my eyes?
0 votes · 1 answers · Test Engineering Admin (471) Admin asked 2 hours ago · overflowai-generated

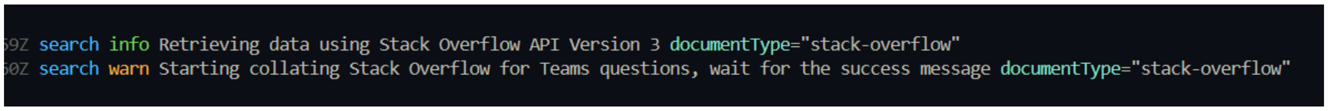
Test Engineering Admin
Reputation: 471 · Admin
Position: Engineer
View Profile

You should see **Ask a Question** in the sidebar everywhere in Backstage. Click this to bring up the "Ask a Stack Overflow Question" modal.



Troubleshooting and understanding errors

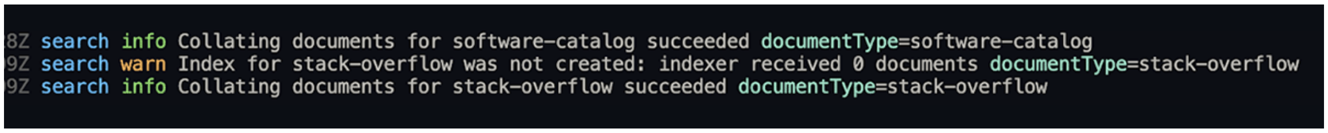
If your Stack Overflow site has many questions, it can take some time for the collator to build the index. If everything is configured correctly, you should see messages like these in the console:



```
59Z search info Retrieving data using Stack Overflow API Version 3 documentType="stack-overflow"
50Z search warn Starting collating Stack Overflow for Teams questions, wait for the success message documentType="stack-overflow"
```

If the collator plugin runs without error but doesn't receive any data from Stack Overflow, the application will fail to create the index. If this happens, check for a warning message in the backend console stating that the index for Stack Overflow was not created.

Double-check the configuration file to ensure your Stack Overflow credentials and other configurations are correct. For more information, refer to the Stack Overflow for Teams API v3 Overview article as well as the collator plugin's config definition file.



```
8Z search info Collating documents for software-catalog succeeded documentType=software-catalog
9Z search warn Index for stack-overflow was not created: indexer received 0 documents documentType=stack-overflow
9Z search info Collating documents for stack-overflow succeeded documentType=stack-overflow
```

If you don't see any warnings or errors, the indexer has successfully received the search data.

If you need further support or have questions, contact your site administrator.