Backstage.io Integration

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

Document generated 12/10/2024

PDF VERSION

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

Applies to: Free, Basic, Business | Enterprise

Overview

Backstage.io is a developer portal that streamlines software development by providing a common repository of code, libraries, documentation, and other resources. You can connect Backstage to your Stack Overflow for Teams Basic, Business, or Enterprise site to index and display search results in your Backstage application.

This guide details the steps for setting up this integration, including configuring the backend (underlying application code) to use Stack Overflow's API and optionally customizing the frontend (display code) to show search results.

NOTE: *This integration works only with Stack Overflow for Teams Basic, Business, or Enterprise plans (not Stack Overflow for Teams Free).*

Obtain API Credentials

In order to connect Stack Overflow for Teams with Backstage you'll first need to obtain the necessary API credentials so that your Backstage application can authenticate when making requests to your Team.

For Teams Basic and Business, you'll be using PAT Tokens. For more information, read the Stack Overflow for Teams API.

For Teams Enterprise, you'll obtain an API key from your Stack Overflow for Teams site. For detailed instructions, go to https://[your_site]/API/docs/authentication and read the "Instructions for Creating a Key" section of your site's API documentation.

Backend system

Backstage has made some changes to how the framework's backend system works. The instructions in this article are intended for users of the New Backend System.

You can still use this integration if you don't intend to migrate to the new backend system. Start by completing the Getting Started with Search guide from Backstage, then follow these instructions in the Stack Overflow search collator GitHub repository.

NOTE: *The following instructions assume the use of the new backend system.*

Installation

1. Add the Stack Overflow search collator plugin as a dependency to your Backstage application.

From your Backstage root directory run:

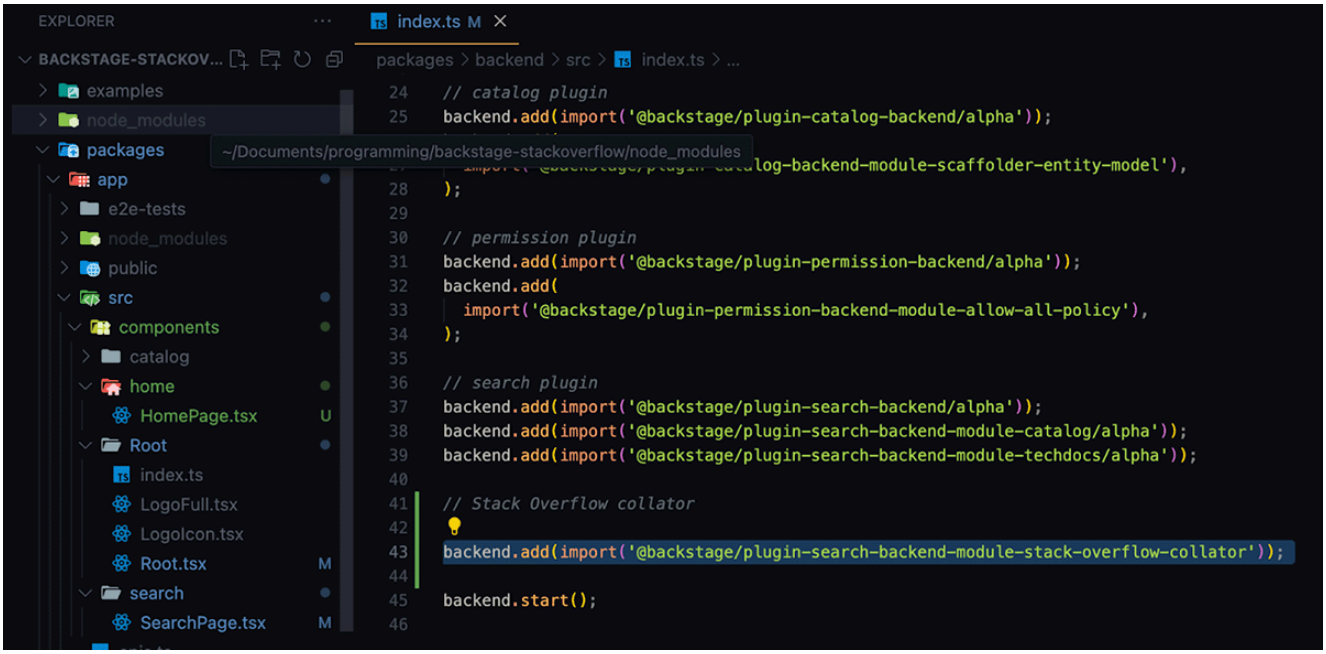
```
yarn --cwd packages/backend add @backstage/plugin-search-backend-module-stack-overflow-collator
```

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

2. Import the collator.

Edit packages/backend/src/index.ts and add the collator.

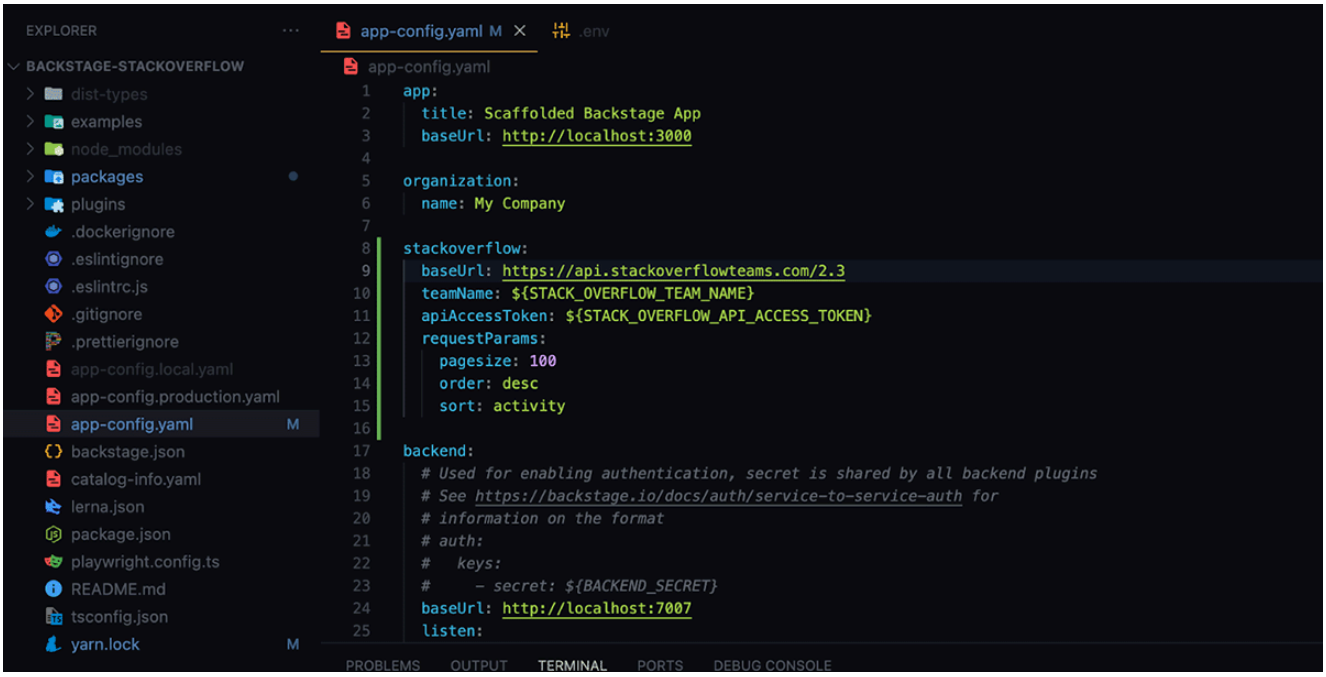
```
backend.add(import('@backstage/plugin-search-backend-module-stack-overflow-collator'));
```



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

Edit the .yaml config file in the root directory. Configure the app with the credentials and request parameters as below.

```
stackoverflow:
baseUrl: https://api.stackoverflowteams.com/2.3
teamName: ${STACK_OVERFLOW_TEAM_NAME}
apiAccessToken: ${STACK_OVERFLOW_API_ACCESS_TOKEN}
requestParams:
    pagesize: 100
    order: desc
    sort: activity
```



For **Stack Overflow for Teams Enterprise,** you don't need to specify the teamName and should remove that line from configuration file. You'll also use an API key instead of an access token. For more information on how to generate and find this key, read your site's API 2.3 documentation at https://[your_site]/API/docs/authentication.

Below is an example configuration for Stack Overflow for Teams Enterprise.

```
stackoverflow:
baseUrl: https://[your-enterprise-site]/api/2.3
apiKey: ${STACK_OVERFLOW_API_KEY}
requestParams:
    pagesize: 100
    order: desc
    sort: activity
```

NOTE: *The Stack Overflow Backstage plugin doesn't support API v3.*

4. Define environment variables with a dotenv file.

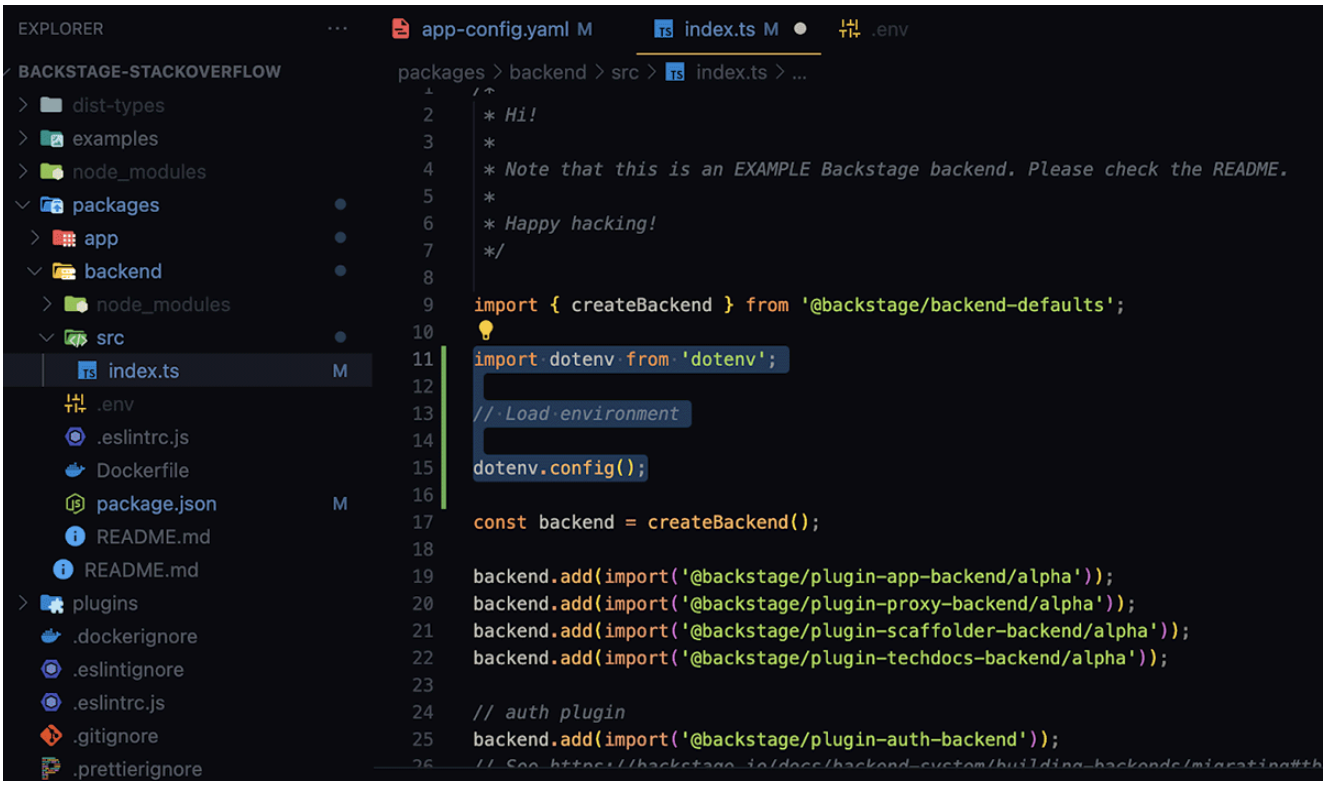
While you can directly replace the placeholders in your .yaml file with the specific values you need, it's best practice to define them in an environment file instead.

In this example, we'll use dotenv for this. To add dotenv to the backend of your app, run the following command on the console:

`yarn—cwd packages/backend add dotenv`

After installing dotenv, navigate to packages/backend/src/index.ts to configure your backend to use the dotenv module. Add the following lines of code:

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



Create a .env file inside the packages/backend folder and add the values of each variable. For example:

5. Make sure all dependencies are installed.

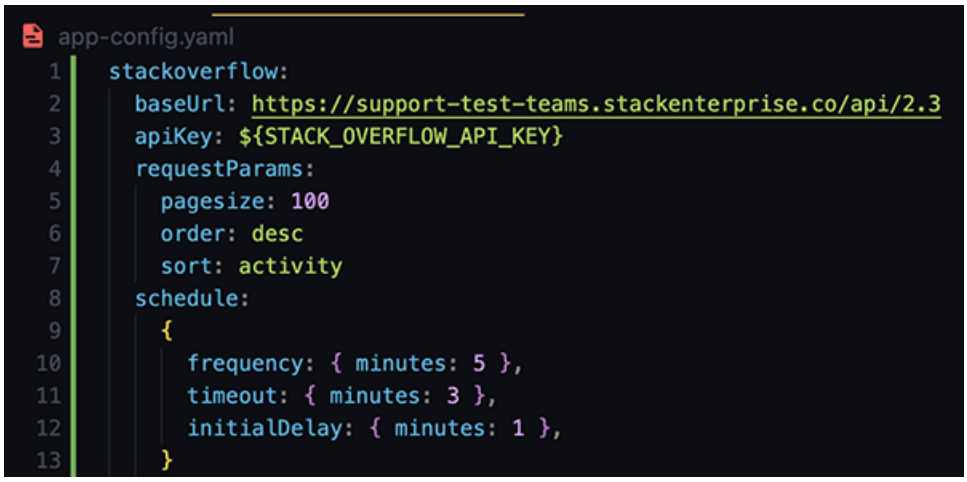
Before running the app, ensure all dependencies are correctly installed by running 'yarn install' on the Backstage root directory one last time. Once that's done, run the app and look for any errors.

6. Customize collator schedule (optional).

Within the same Stack Overflow object in the app-config.yaml file, you can add a schedule property to control when the collator retrieves information from your Stack Overflow for Teams site.

This schedule property should contain an object that specifies the following:

- **frequency** You can define frequency in hours, minutes, or seconds. The default is 10 minutes.
- **timeout** This specifies how long the task will run before being considered timed out (the default is 5 minutes).
- **initialDelay** This is the delay before the task runs for the first time (the default is 3 seconds).



Understanding errors

With the new backend system, few things can go wrong when installing plugins. One possible error with the Stack Overflow collator plugin is that, despite the collator succeeding, your indexer doesn't receive any data from Stack Overflow. This will cause the application to fail in creating the index.

If this happens, you'll see a warning message in the backend console stating that the index for Stack Overflow was not created. Double-check the configuration file you edited before to ensure your credentials and other optional configurations are correct. For more information, refer to Teams API documentation and the config definition file for the backend plugin.

```
[1] 2024-07-19T11:30:42.728Z search info Collating documents for software-catalog succeeded documentType=software-catalog
[1] 2024-07-19T11:30:44.109Z search warn Index for stack-overflow was not created: indexer received 0 documents documentType=stack-overflow
[1] 2024-07-19T11:30:44.109Z 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.

Compose the SearchPage.tsx with Stack Overflow for Teams search results (optional)

You can optionally compose (format) your frontend to better display the search results. As a prerequisite, you'll need to have your SearchPage.tsx ready for modifications. For more information, read the Backstage Getting Started with Search guide.

In this example, you'll explicitly obtain the search results that have been indexed 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 edit that file.*

1. Install the frontend Stack Overflow plugin.

Run: `yarn --cwd packages/app add @backstage-community/plugin-stack-overflow`

By default, the SearchPage.tsx displays the search results without conditional rendering. You'll modify this to control how the search results are displayed based on the type of search result that is returned.

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

Add to the existing `<SearchResult><SearchResult/>` code block. This will contain the following piece of code by default:

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

Replace this with the code snippet from the Backstage Customizing Search documentation:

```
<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 these errors with the following steps.

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

4. Add a default case that 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}
              />
            );
          // ...
        }
```

Import the `<DefaultResultListItem/>` component.

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

5. Add the StackOverflow case to conditionally render the component from the frontend plugin we installed.

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

Import the list item component from `@backstage-community/plugin-stack-overflow`.

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 > ...

```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}
```

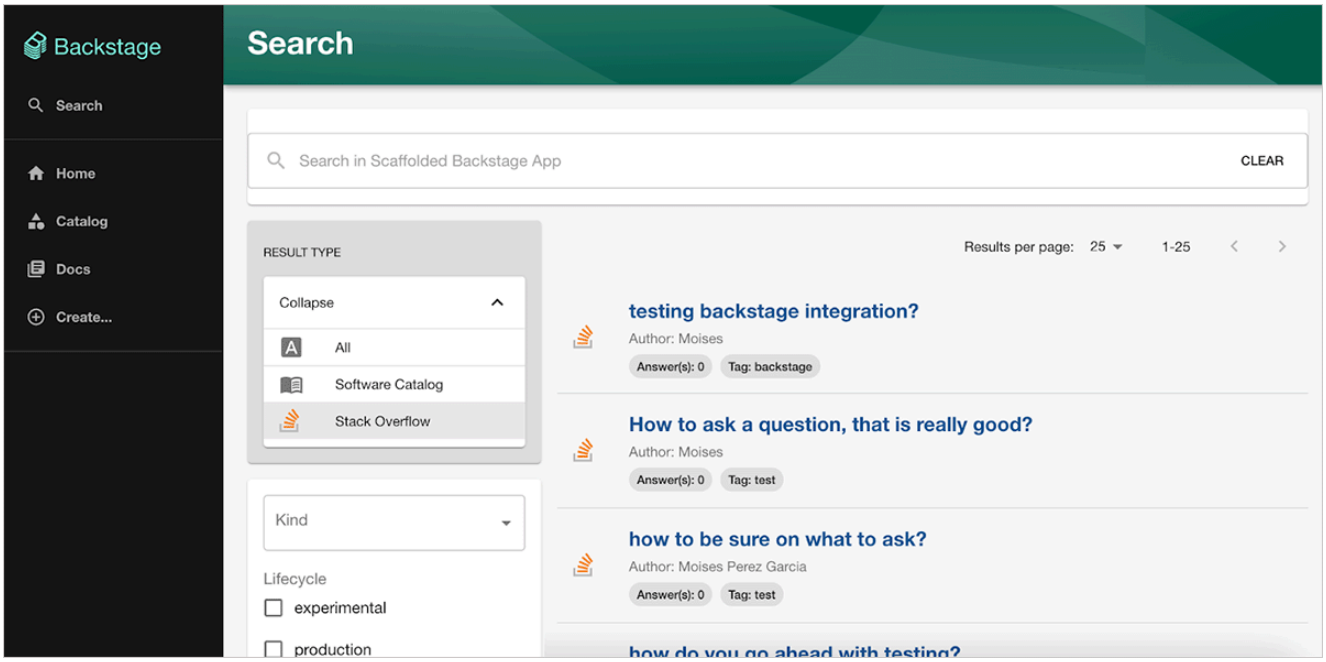
7. Import the `<StackOverflowIcon />` component from `@backstage-community/plugin-stack-overflow`.

```tsx
import {
  StackOverflowIcon,
  StackOverflowSearchResultListItem,
} from '@backstage-community/plugin-stack-overflow';
```

Configuration is complete.

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

This should be the final result of your search frontend:



Example SearchPage.tsx code

Below is an example code snippet for the entire SearchPage.tsx file used in this article.

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

import { CatalogSearchResultListItem } from '@backstage/plugin-catalog';
import {
 catalogApiRef,
 CATALOG_FILTER_EXISTS,
} from '@backstage/plugin-catalog-react';

import { SearchType } from '@backstage/plugin-search';
import {
 SearchBar,
 SearchFilter,
 SearchResult,
 SearchPagination,
 useSearch,
 DefaultResultListItem,
} from '@backstage/plugin-search-react';
import { CatalogIcon, Content, Header, Page } from '@backstage/core-components';
import { useApi } from '@backstage/core-plugin-api';
import {
 StackOverflowIcon,
 StackOverflowSearchResultListItem,
} from '@backstage-community/plugin-stack-overflow';

const useStyles = makeStyles((theme: Theme) => ({
 bar: {
   padding: theme.spacing(1, 0),
 },
 filters: {
   padding: theme.spacing(2),
```

```
    marginTop: theme.spacing(2),
```

NOTE: *The* `<HomePageStackOverflowQuestions/>` *component from the frontend plugin does not support Private Team instances. It will only display search results from the main site.*

Need help? Submit an issue or question through our support portal.