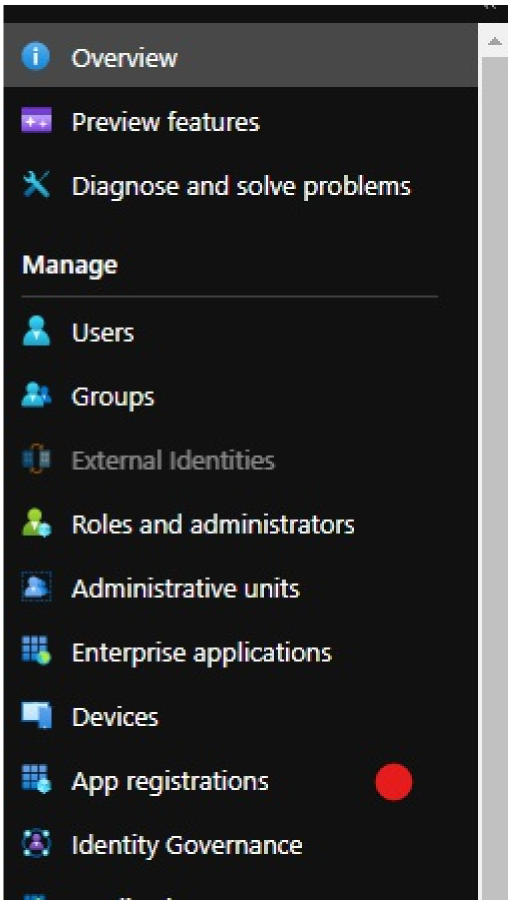
Microsoft Authentication Library (MSAL) for Javascript enables client-side web applications to authenticate users using Azure Active Directory (Azure AD), using security tokens acquired from the Microsoft Identity platform. It is particularly helpful if you want to specify which users can sign in to your application, e.g., your organization or school. You can learn more about it here (https://docs.microsoft.com/en-us/azure/active-directory/develop/msal-overview)

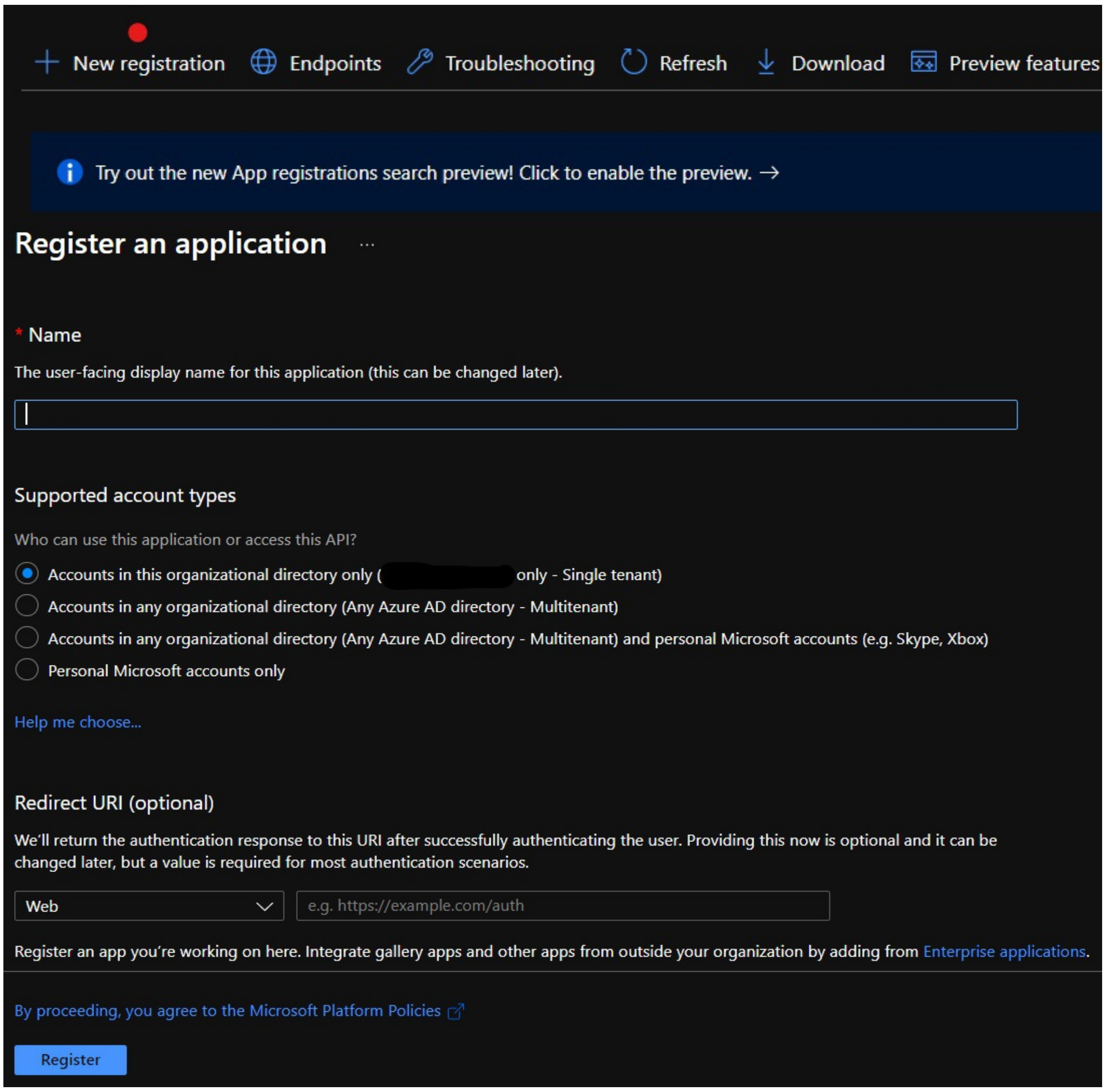
App Registration

To get started, we need to register our application in the Microsoft Identity Platform (Azure AD). We will register a single-page application (SPA) and use the recommended authentication flow, MSAL.js 2.0, which supports the authorization code flow with PKCE. More on this can be found here (https://docs.microsoft.com/en-us/azure/active-directory/develop/scenario-spa-app-registration)

- Sign in to the Azure Portal (https://portal.azure.com/)
- Search for and select Azure Active Directory.
- Under Manage in the side menu, find and select App Registration

- 

- Create a New Registration and fill in the form by entering the name of the app (can be changed later) and selecting the support account types (I used the single-tenant option, the first option in the screenshot below). DO NOT enter a Redirect URI for now.

- 

Try out the new App registrations search preview! Click to enable the preview. →

Register an application ...

* Name

The user-facing display name for this application (this can be changed later).

Supported account types

Who can use this application or access this API?

- [✓ Accounts in this organizational directory only ([redacted] only - Single tenant)]
- Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only

Help me choose...

Redirect URI (optional)

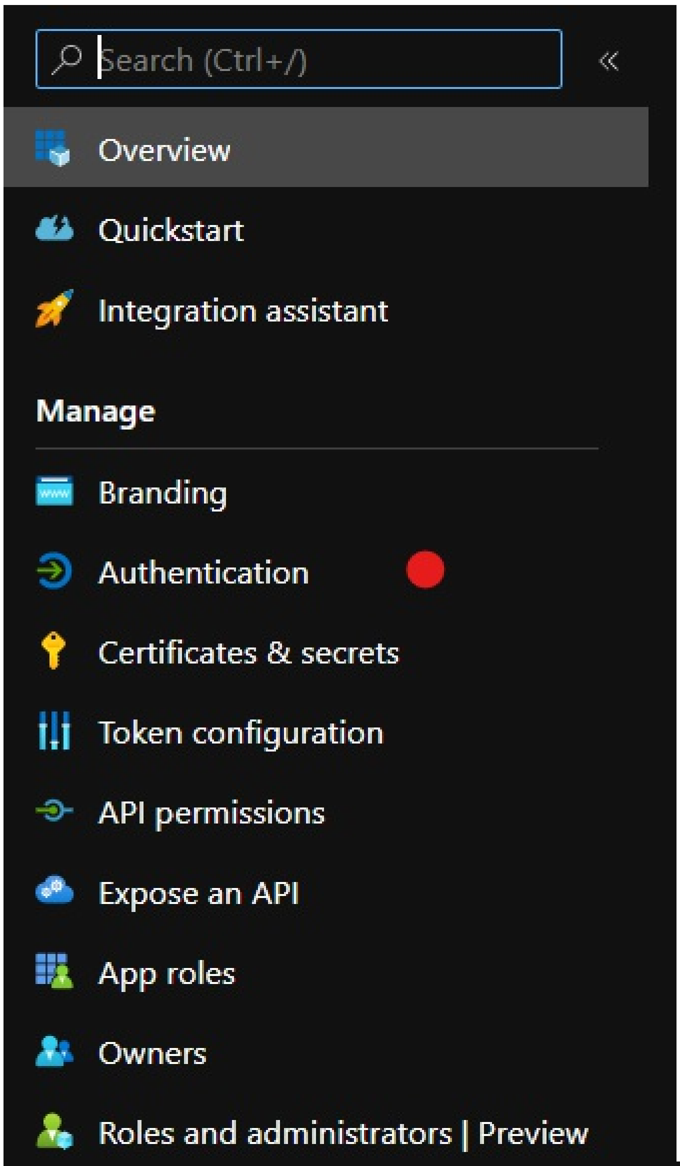
We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

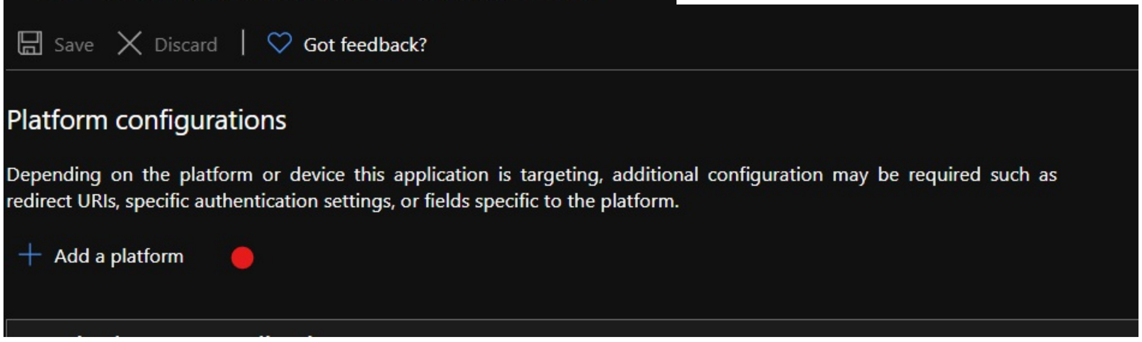
Web | e.g. https://example.com/auth

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from Enterprise applications.

By proceeding, you agree to the Microsoft Platform Policies

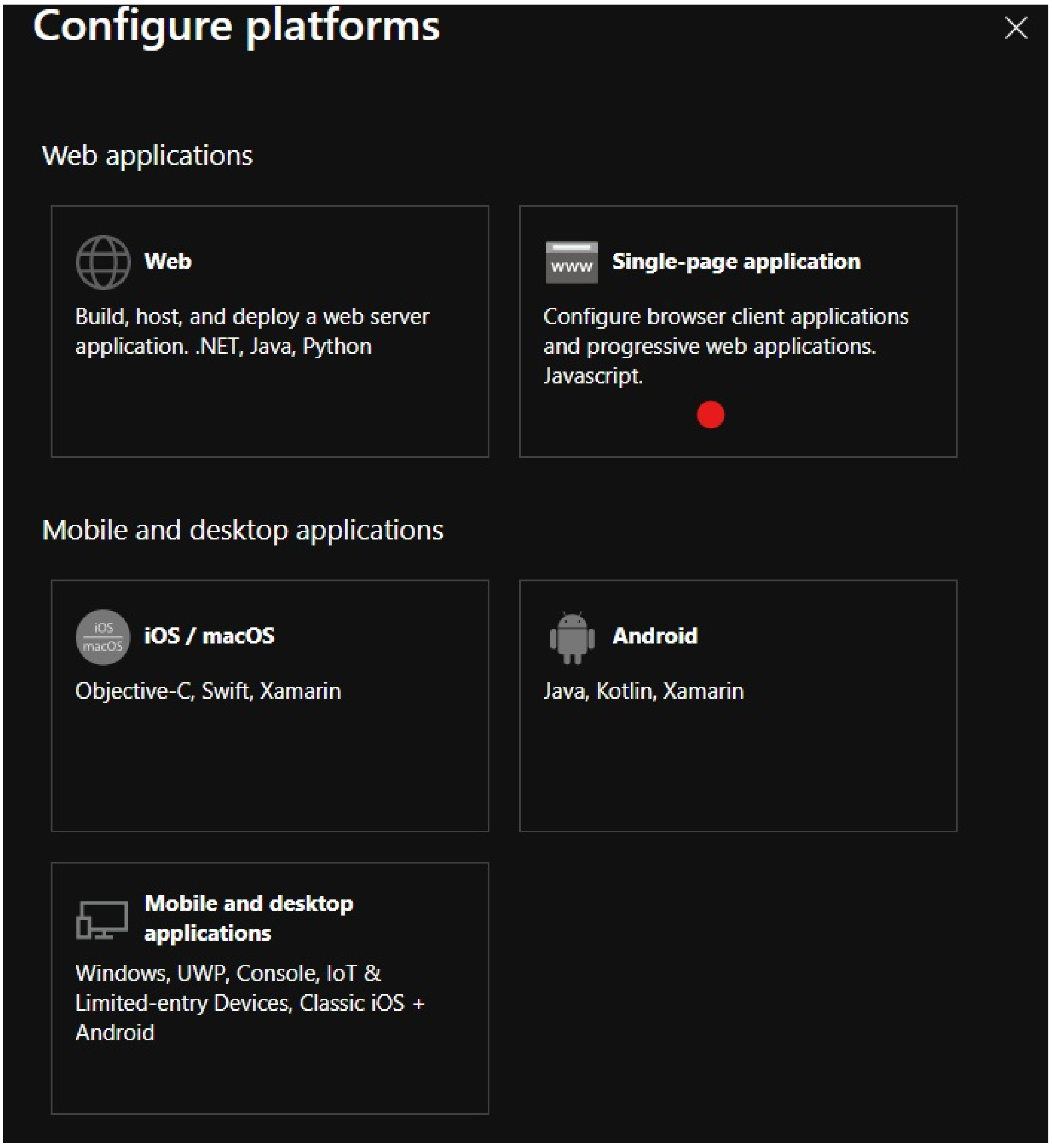
- Select the newly created application and then select Authentication > Add Platform under the Manage menu

- 



Platform configurations

Depending on the platform or device this application is targeting, additional configuration may be required such as redirect URIs, specific authentication settings, or fields specific to the platform.
- Select the Single-Page Application tile under the Web applications heading

- 

Configure platforms

Web applications

Web
Build, host, and deploy a web server application. .NET, Java, Python

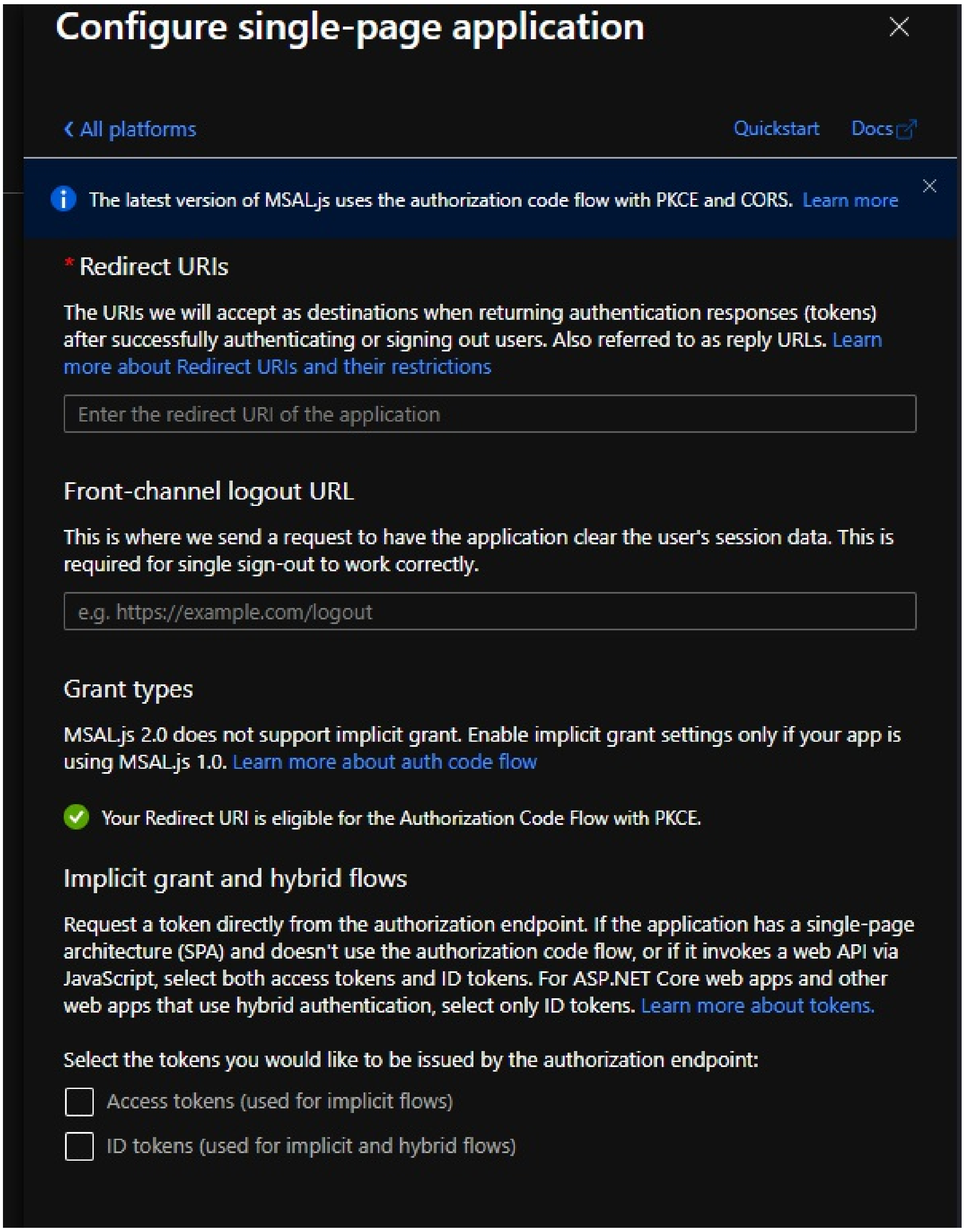
Single-page application
Configure browser client applications and progressive web applications. Javascript.

Mobile and desktop applications

iOS / macOS
Objective-C, Swift, Xamarin

Android
Java, Kotlin, Xamarin

Mobile and desktop applications
Windows, UWP, Console, IoT & Limited-entry Devices, Classic iOS + Android

- 

Configure single-page application

‹ All platforms

The latest version of MSAL.js uses the authorization code flow with PKCE and CORS. Learn more

* Redirect URIs

The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating or signing out users. Also referred to as reply URLs. Learn more about Redirect URIs and their restrictions

Enter the redirect URI of the application

Front-channel logout URL

This is where we send a request to have the application clear the user's session data. This is required for single sign-out to work correctly.

e.g. https://example.com/logout

Grant types

MSAL.js 2.0 does not support implicit grant. Enable implicit grant settings only if your app is using MSAL.js 1.0. Learn more about auth code flow

✓ Your Redirect URI is eligible for the Authorization Code Flow with PKCE.

Implicit grant and hybrid flows

Request a token directly from the authorization endpoint. If the application has a single-page architecture (SPA) and doesn't use the authorization code flow, or if it invokes a web API via JavaScript, select both access tokens and ID tokens. For ASP.NET Core web apps and other web apps that use hybrid authentication, select only ID tokens. Learn more about tokens.

Select the tokens you would like to be issued by the authorization endpoint:

☐ Access tokens (used for implicit flows)

☐ ID tokens (used for implicit and hybrid flows)

- Enter a redirect URI. I'll be using http://localhost:3000/. DO NOT check the boxes under Implicit grant and hybrid flows heading
- Click on the Configure button to finish.

Note You can add more than one redirect URI, for instance, for different environments, development, staging, production, etc.

Installation

Open your terminal and create a Next.js app

```
npx create-next-app msal-next-auth --use-npm



cd msal-next-auth
```

The only dependencies that you will need are the MSAL react and browser libraries.

```
npm install @azure/msal-react @azure/msal-browser
```

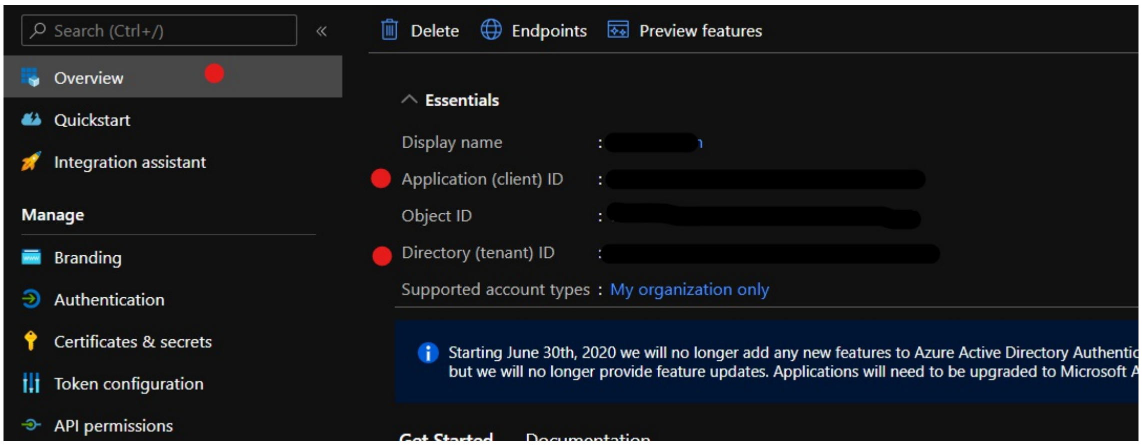
Initialization

Create a file `services/msal.js` in the root of the project and add the following code

```
import * as msal from "@azure/msal-browser";



const msalConfig = {

   auth: {

      clientId: process.env.NEXT_PUBLIC_AZURE_AD_CLIENT_ID,

      authority:
`https://login.microsoftonline.com/${process.env.NEXT_PUBLIC_AZURE_AD_TENAN
T_ID}`,
            redirectUri: '/'

    }

};



const msalInstance = new msal.PublicClientApplication(msalConfig);



export { msalInstance }
```

You can find your CLIENT ID AND TENANT ID in the Azure portal. Select the app you registered and copy and paste the actual values from the Overview page (under Essentials) into an environment variables file, `.env.local`, in the root of the project folder.

```
NEXT_PUBLIC_AZURE_AD_CLIENT_ID='your-client-id'

NEXT_PUBLIC_AZURE_AD_TENANT_ID='your-tenant-id'
```



Essentials

Display name	: [redacted]
Application (client) ID	: [redacted]
Object ID	: [redacted]
Directory (tenant) ID	: [redacted]
Supported account types	: My organization only

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentic
but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft A

We'll be using React's Context API to provide/share the MSAL service instance to all our components/pages. To do that, make the following changes to `pages/_app.js`

```
import { MsalProvider } from '@azure/msal-react';
import { msalInstance } from '../services/msal';

import '../styles/globals.css';



function MyApp({ Component, pageProps }) {

  return (

    <MsalProvider instance={msalInstance}>

      <Component {...pageProps} />

    </MsalProvider>

  );

}



export default MyApp;
```

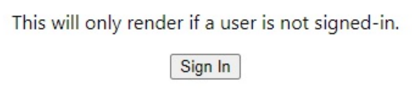
Sign-in Functionality

The MSAL React library allows us to protect our pages and components by wrapping them with the `MsalAuthenticationTemplate` component. Paired with `UnauthenticatedTemplate`, this can be useful when rendering specific content to authenticated or unauthenticated users respectively.

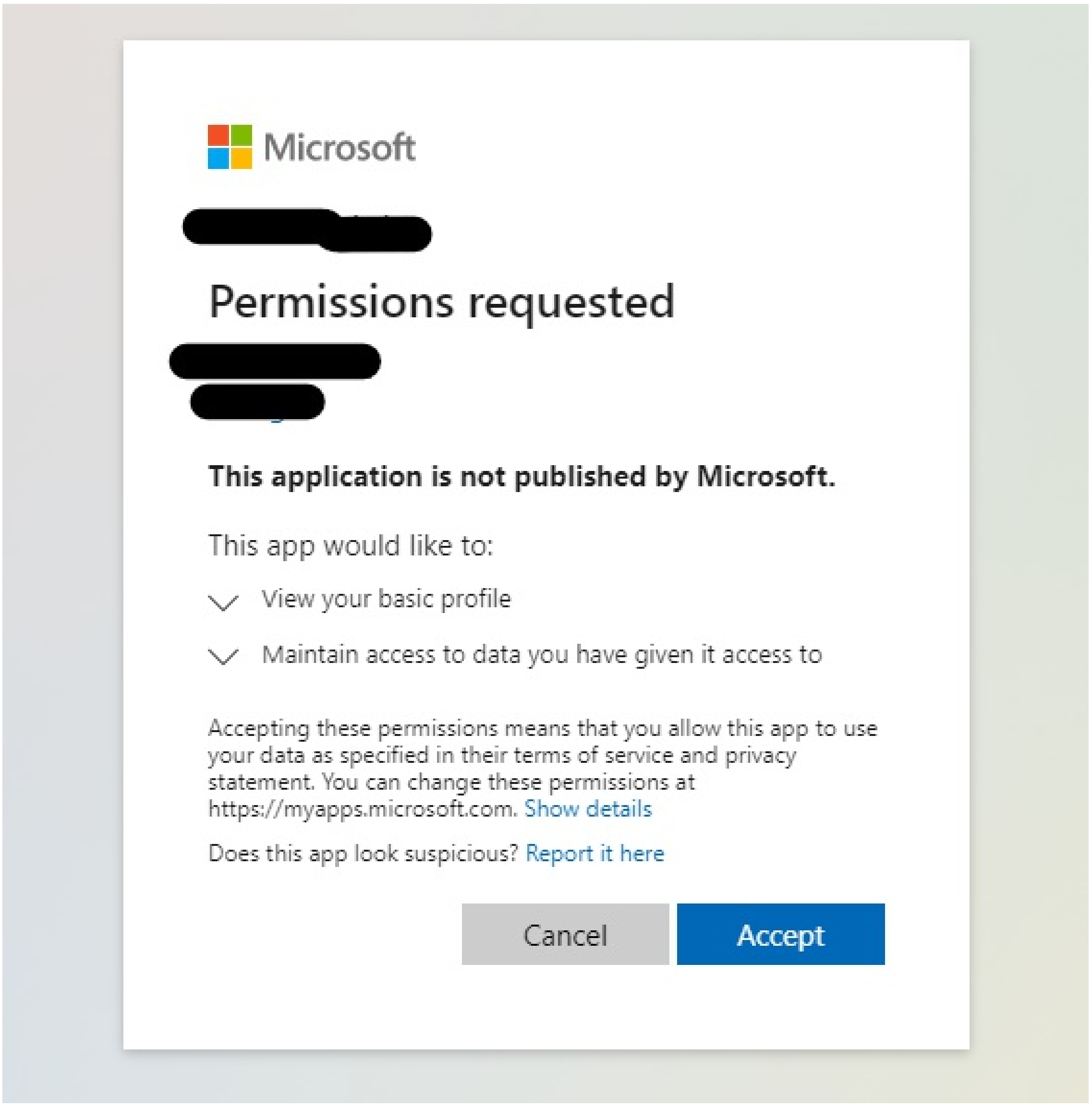
Update our home page `pages/index.js`, with the following code

```
import {

  AuthenticatedTemplate,

  UnauthenticatedTemplate,
  useMsal,

} from '@azure/msal-react';

import Head from 'next/head';

import styles from '../styles/Home.module.css';



function SignInButton() {

  // useMsal hook will return the PublicClientApplication instance you provided to
MsalProvider

  const { instance } = useMsal();



  return <button onClick={() => instance.loginRedirect()}>Sign In</button>;

}



function WelcomeUser() {

  const { accounts } = useMsal();

  const username = accounts[0].username;



  return <p>Welcome, {username}</p>;

}



export default function Home() {

  return (
    <div className={styles.container}>
      <Head>
        <title>Azure AD Authentication using MSAL and Next.js</title>
      </Head>

      <AuthenticatedTemplate>
        <p>This will only render if a user is signed-in.</p>
        <WelcomeUser />
      </AuthenticatedTemplate>
      <UnauthenticatedTemplate>
        <p>This will only render if a user is not signed-in.</p>
        <SignInButton />
      </UnauthenticatedTemplate>
    </div>
  );
}
```

If you run the application (`npm run dev`), you should see the following in the browser.


This will only render if a user is not signed-in.

When you click the sign in button, you will be prompted to sign in and accept requested permissions…



Microsoft

[redacted]

Permissions requested

[redacted]

This application is not published by Microsoft.

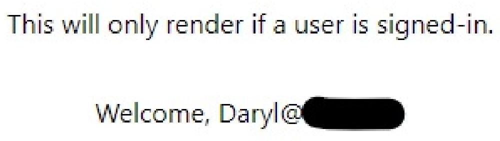
This app would like to:

- View your basic profile
- Maintain access to data you have given it access to

Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. You can change these permissions at https://myapps.microsoft.com. Show details

Does this app look suspicious? Report it here

… and then you will be redirected back to the application.



This will only render if a user is signed-in.

Welcome, Daryl@[redacted]

Notice the change in content rendered, when unauthenticated to authenticated. This is determined by the `UnauthenticatedTemplate` and `AuthenticatedTemplate` wrappers.

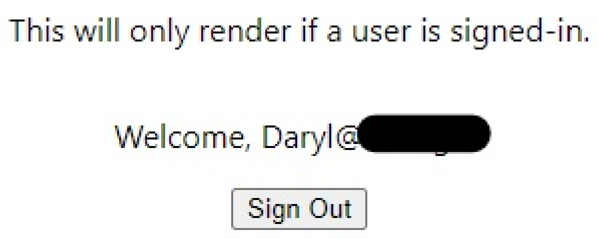
MSAL React also provides an alternative way to determine a user's authentication status using hooks, specifically the `useIsAuthenticated` hook. Learn more here

Sign-out functionality

Make the following changes to `pages/index.js`

```
...

// Define sign out button

function SignOutButton() {

  const { instance } = useMsal();



  return <button onClick={() => instance.logoutRedirect()}>Sign Out</button>;

}



...



export default function Home() {

  return (

    <div className={styles.container}>

      <Head>
    <title>Azure AD Authentication using MSAL and Next.js</title>

  </Head>



  <AuthenticatedTemplate>

    <p>This will only render if a user is signed-in.</p>

    <WelcomeUser />

    <SignOutButton /> // <-- Add to authenticated content

  </AuthenticatedTemplate>

...
```

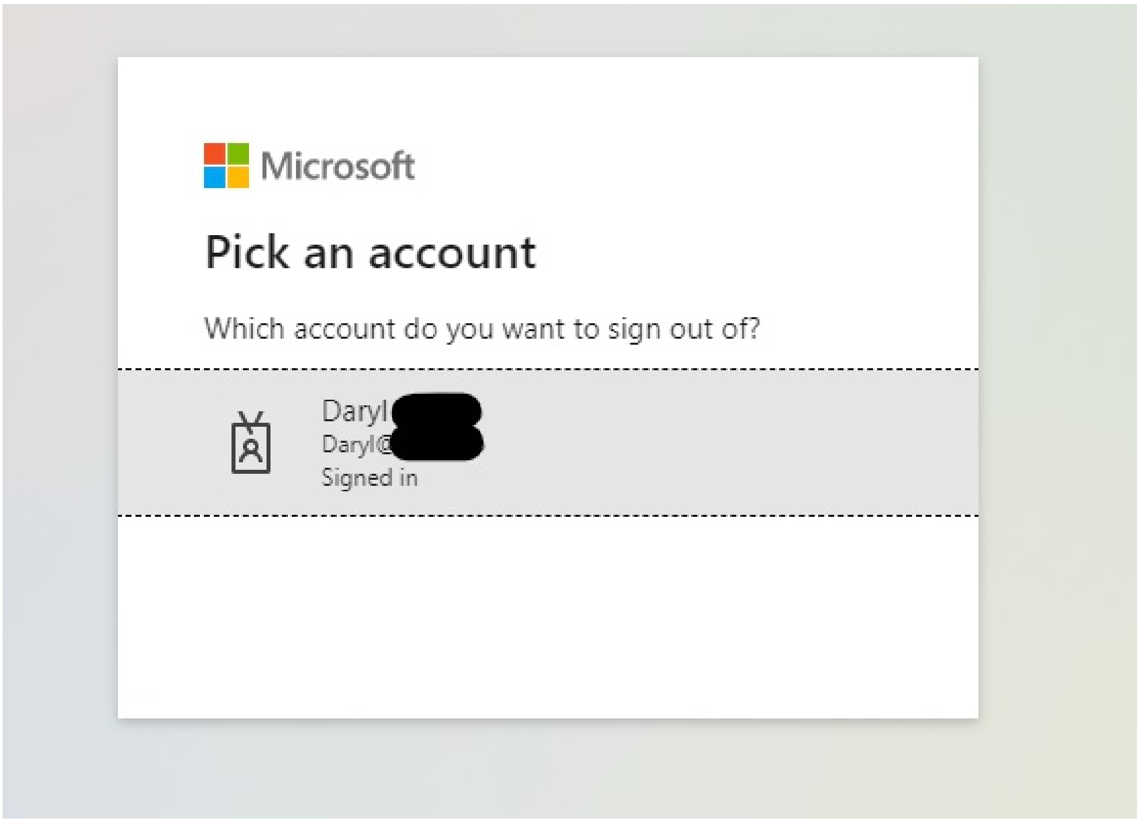
Run the app again, if not already running, `npm run dev`. The authenticated render should now look like this…



This will only render if a user is signed-in.

Welcome, Daryl@[redacted]

Clicking the sign-out button will redirect and prompt you to select the account to sign out from…


Microsoft

Pick an account

Which account do you want to sign out of?

Daryl [redacted]
Daryl@ [redacted]
Signed in

You can read more about Sign in and sign out functionality here (https://docs.microsoft.com/en-us/azure/active-directory/develop/scenario-spa-sign-in?tabs=javascript2)