Ruchi Shah

NEUDESIC TECHNOLOGIES

AAD Auth Starter Kit for REACT

# **Introduction:**

This implementation demonstrates authentication by using Azure AD and the Microsoft identity platform. It is based on Authorization Code Grant Flow. It contains steps to accelerate setting up Azure Active Directory to authenticate users logging in from the client application built using React.

* Sign in users to web applications and provide authorized access to protected web APIs.
* Protect a web API by requiring an access token to perform API operations.

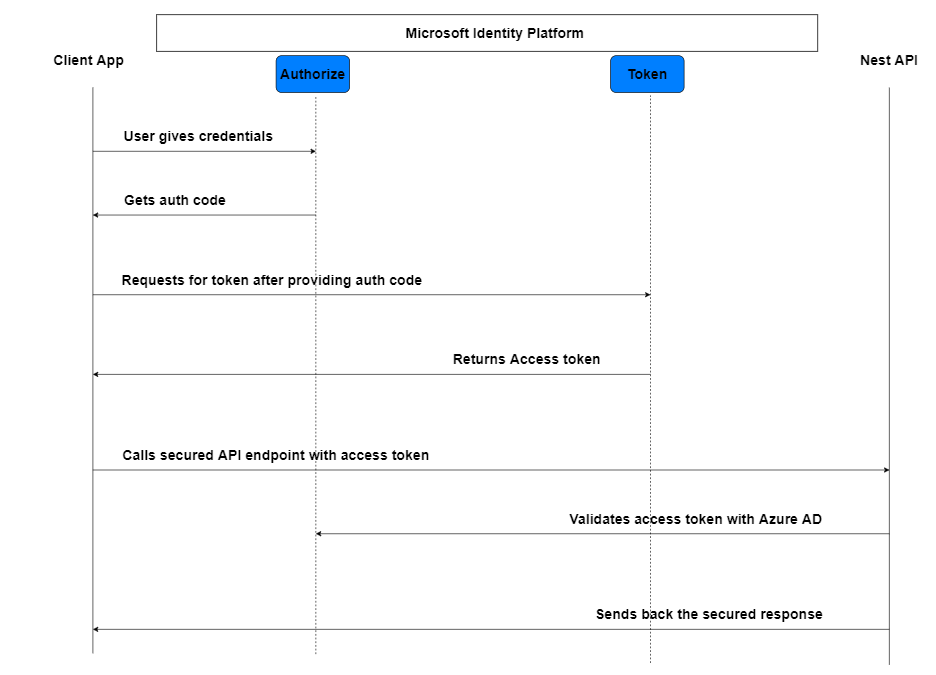
**This Implementation has below elements:**

1. Azure Active Directory
2. Nest JS Application – Web APIs
3. Client - React SPA

Goal: Authenticating Users Logging into React Application through AAD and using the access token received on successful Authentication from AAD to access APIs built using NestJS protected via AAD.

1. Create a React project with npm
   1. Register the application in the Azure portal
   2. Add code to support user sign-in and sign-out
   3. Add code to call Nest JS Web API
2. Create a NestJS Web API application
   1. Register Web API Application and Expose Web APIs
   2. Add Code to configure Nest JS App to authorize with AAD

**Flow Diagram**



**Prerequisites:**

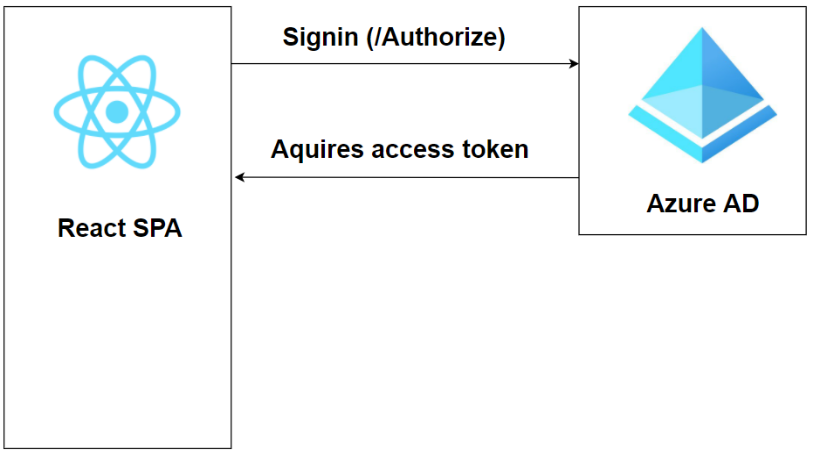
NodeJS – version 16.x

React – version 17.x

NestJS – version 14.x

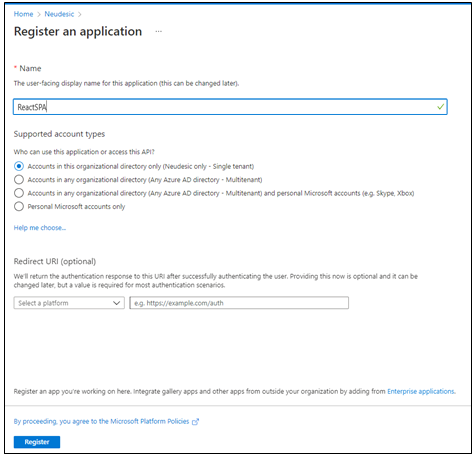
**CONFIGURE REACT SPA IN AAD AND CODE SET UP**

FLOW BETWEEN REACT SPA AND AAD



[STEPS TO REGISTER REACT APP IN AAD](#REGISTER_REACT_SPA_CLIENT_APPLICATION)

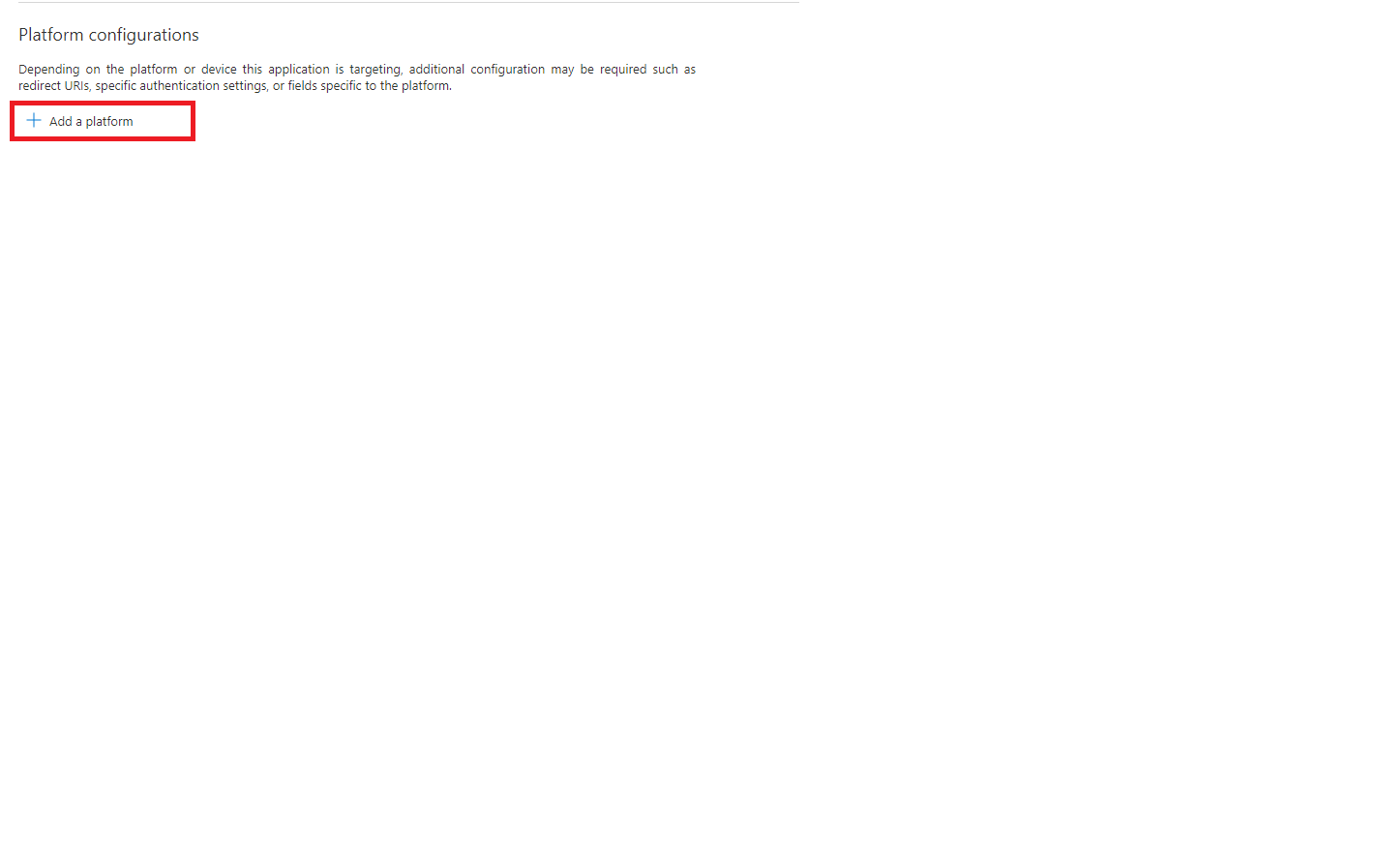
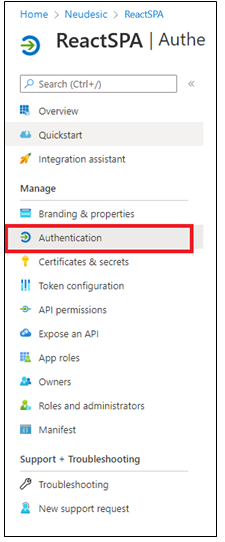
* 1. To Register the React SPA, we need to follow steps given at <https://docs.microsoft.com/en-us/azure/active-directory/develop/quickstart-register-app>
  2. Navigate to Azure Active Directory à App Registrations à New Registration



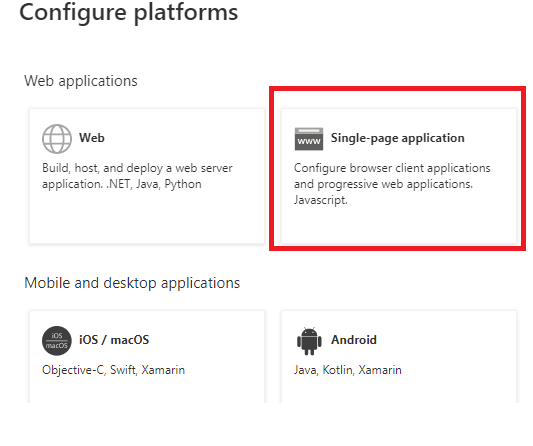
* 1. On Successful registration note down the **ClientID** and **TenantID** for future use

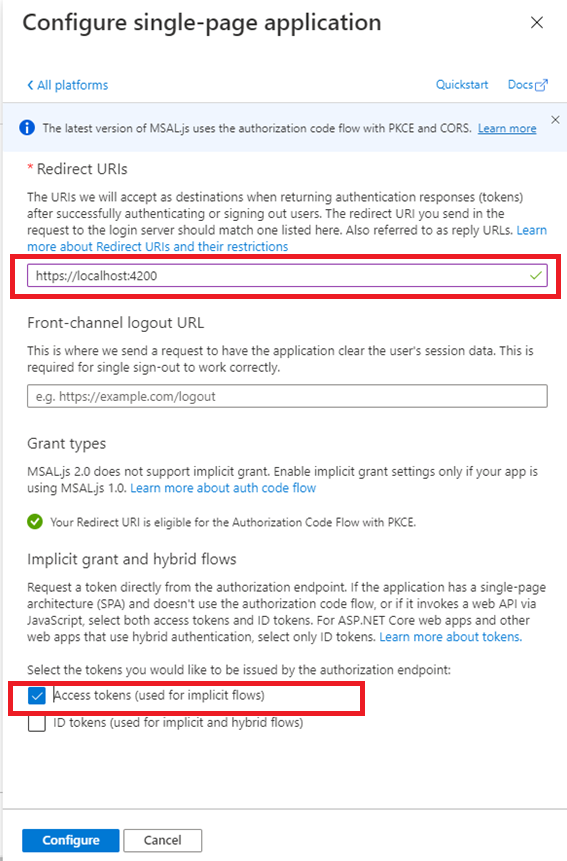


* 1. Navigate to the Authentication section and add a platform

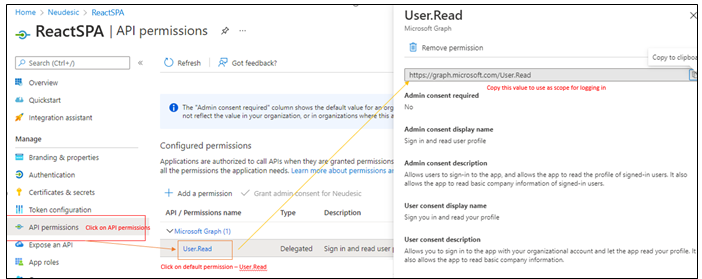


Select Single Page Applications



* 1. Provide redirect URI and token type that must be issued:
  2. Navigate to ReactSPA à API Permissions à Microsoft.Graph/User.Read

Note down the default permission available here and to be used later as value for “defaultScope” in env-config.js



REACT CLIENT APP CODE SET UP / CONFIGURATION AND EXECUTION

Follow steps below to set up React SPA Client App:

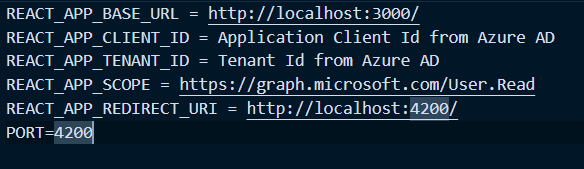
1. Repository Location: <https://github.com/Monaswini/user-login-azureAd>

Branch: **Master**

1. Clone the repository, open the code in IDE and run **npm install** in the root directory (**ReactSPA**)
2. Follow the instructions to register a single-page applicationin the Azure portal

[# REGISTER\_REACT\_SPA\_CLIENT\_APPLICATION](#REGISTER_REACT_SPA_CLIENT_APPLICATION)

1. Once the App Registration is complete update the following keys in the **.env.development or .env** file under the **root** folder.
   * **ClientID** (Registered React SPA ClientID)
   * **TenantID (**Registered React SPA TenantID)
   * **Redirect URL** (Url of the Client Application Server. It should match with the **Redirect URL** specified while registering the Client SPA in the Azure portal)
   * **defaultScope (**Permission Url noted down from API permissions)



1. Integrate MSAL Libraries to guard routes:

React App uses the Microsoft Authentication Library (MSAL). MSAL React enables React 16+ applications to authenticate enterprise users by using Azure Active Directory (Azure AD). These libraries will install as a part of step#2

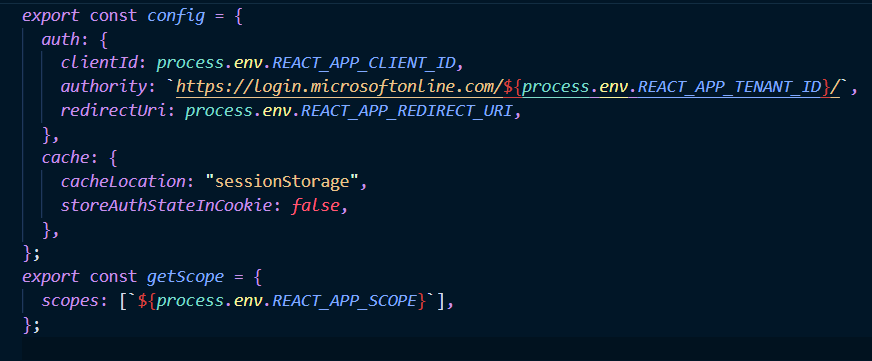
[MSAL React](https://github.com/AzureAD/microsoft-authentication-library-for-js/tree/dev/lib/msal-react)- Microsoft Authentication Library for JavaScript React Wrapper

[MSAL Browser](https://github.com/AzureAD/microsoft-authentication-library-for-js/tree/dev/lib/msal-browser) Microsoft Authentication Library for JavaScript v2 browser package

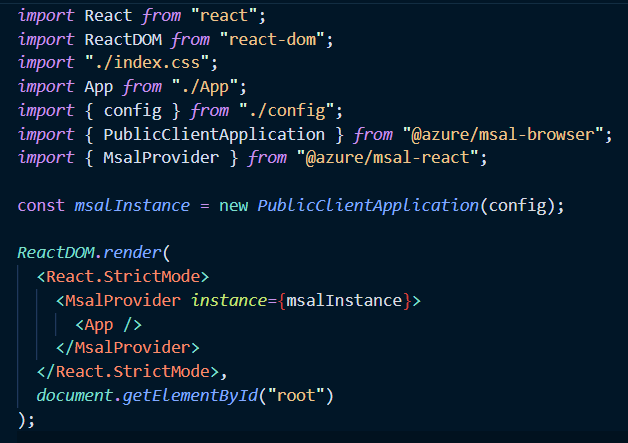
1. Configurations for MSAL are as under:

NOTE: These settings are already in place. You can read through or skip and go to next step

* 1. MSAL Config located at config.js



* 1. Wrap the <App /> component in the src.index.js with the MsalProvider.



* 1. Use <AuthenticatedTemplate /> component from msal-react library in the <App /> component to render certain component only when you are authenticated.

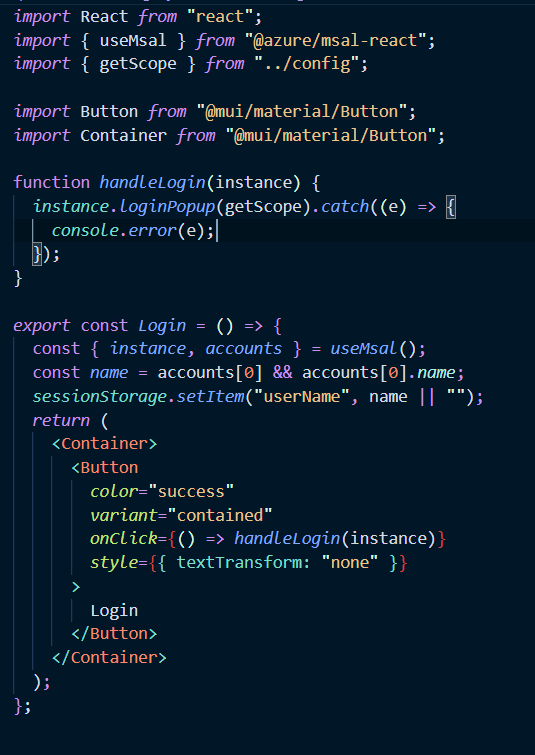


* 1. Create your custom interceptor in the root folder to add the access token in the request header.



* 1. Login (src/components/Login.jsx)

Login component used PopupRquest with scopes this application needs for successful login.



* 1. Logout (src/components/Logout.jsx)

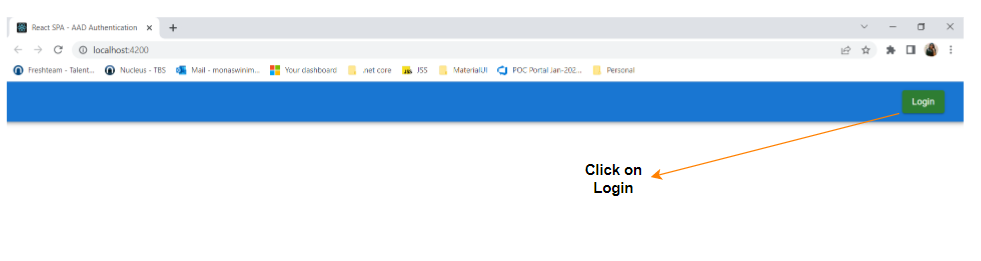


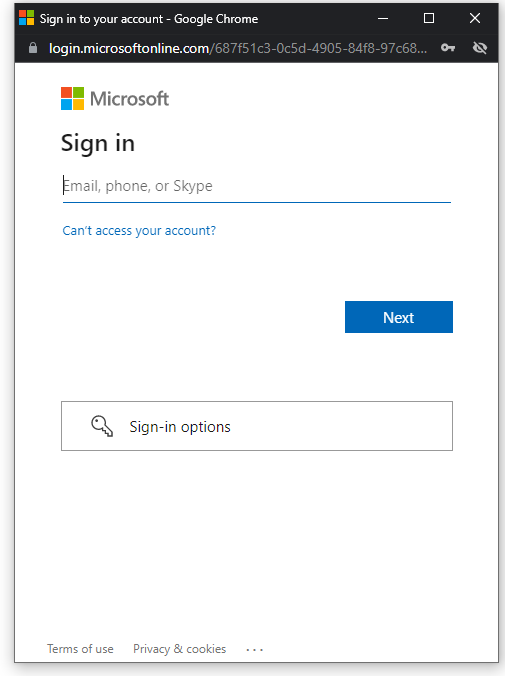
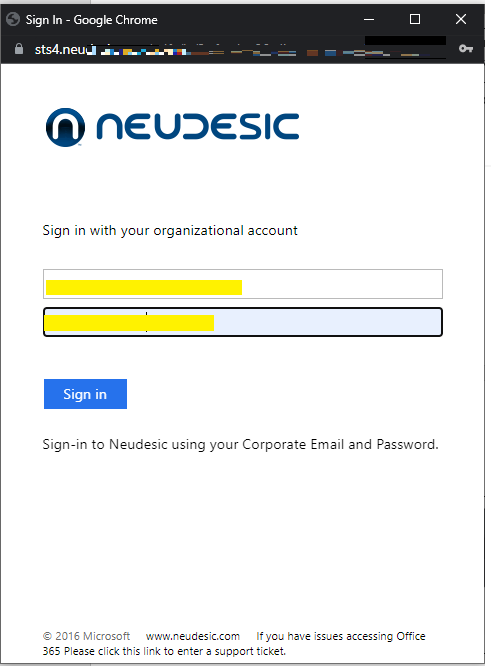
1. Now run **npm start** to see the running Application.

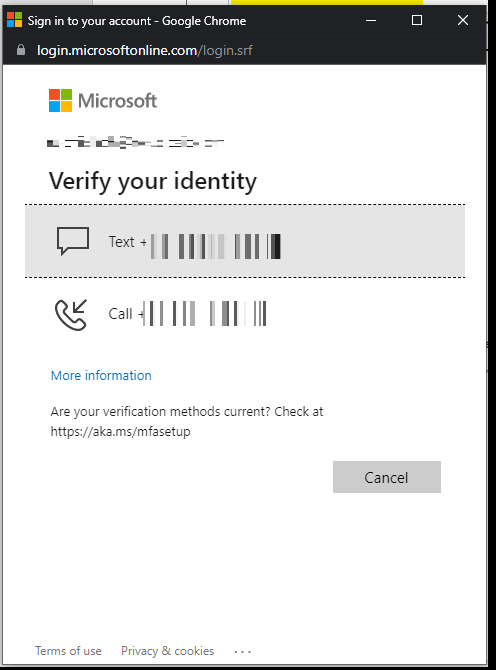
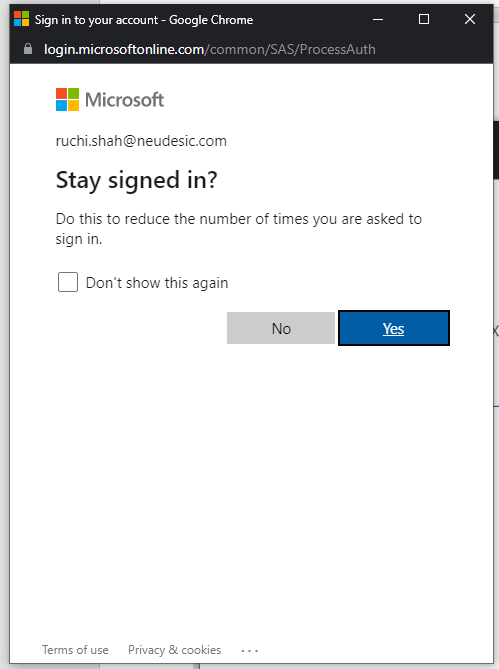
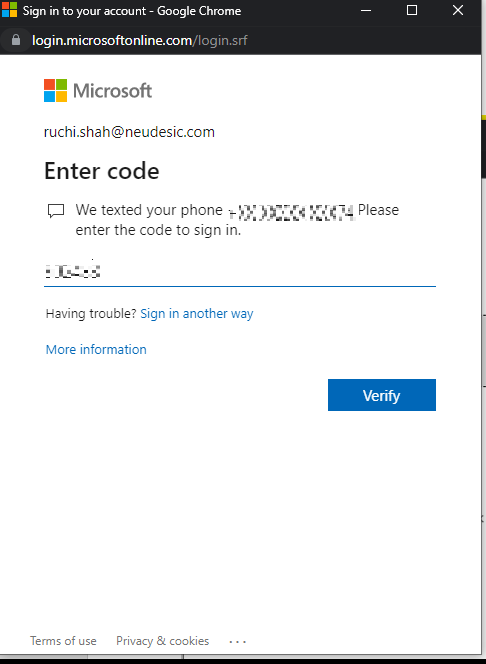
7.1 – open a browser and navigate to <http://localhost:4200>

7.2 – Login to application through Popup configured using MSAL Library

**Figure 1 - Login Page**

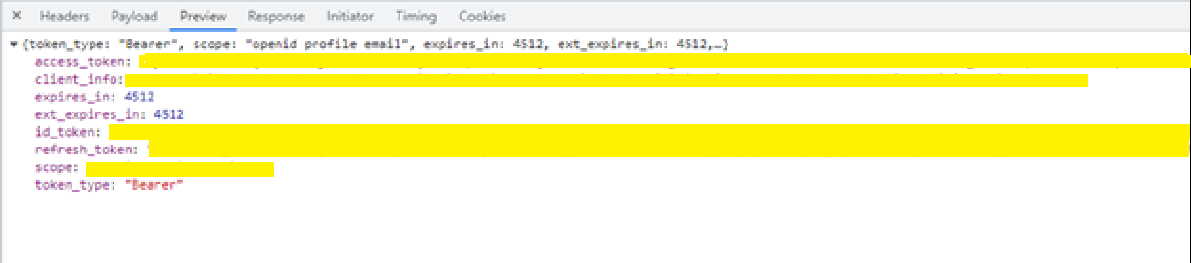


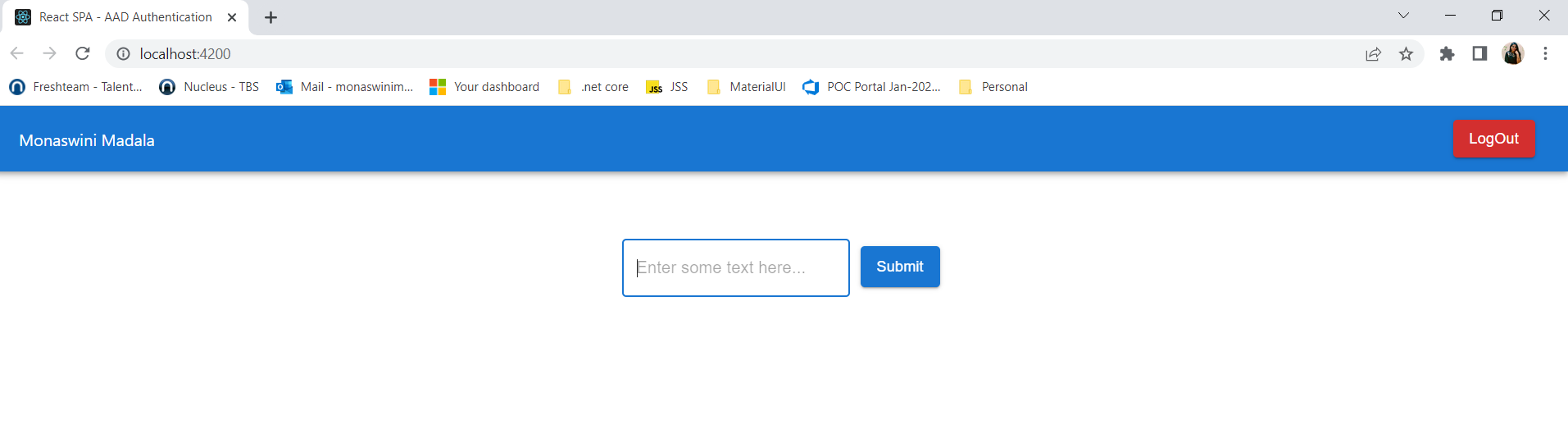
 **Figure 2 – Enter credentials and go through secondary authentication mechanism**

1. 8. On Successful login, MSAL makes a call to [https://login.microsoftonline.com/{{tenant-id}}/oauth2/v2.0/token](https://login.microsoftonline.com/%7b%7btenant-id%7d%7d/oauth2/v2.0/token) to fetch access token

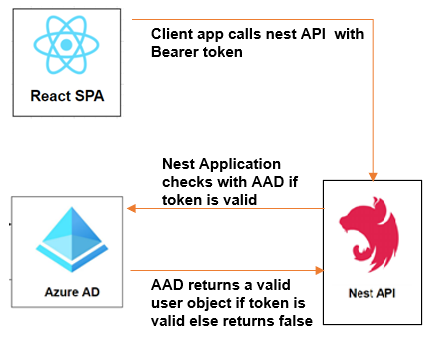
A successful response contains access token.



1. The page loads with user name displayed and log out button

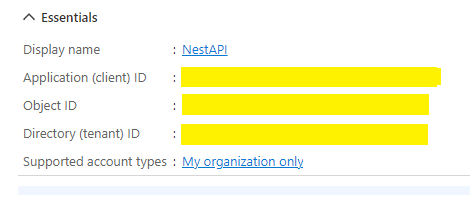
**Web API (NestJS) APPLICATION IN AAD**

FLOW DIAGRAM

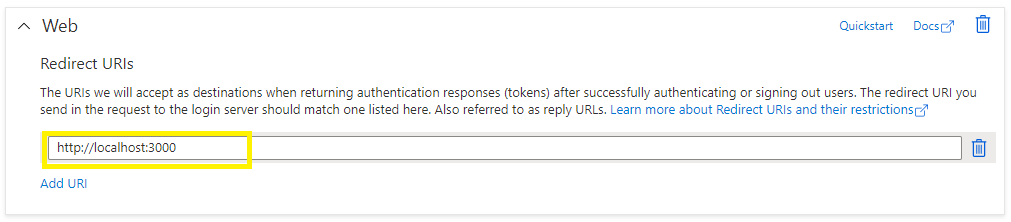


STEPS TO REGISTER Web API APP IN AAD

1. Register the NestJS Web API Application in AAD. Follow the process of App Registration
2. On Successful registration **Overview** section will appear as:

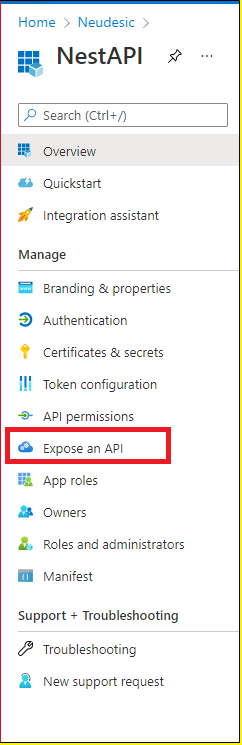
****

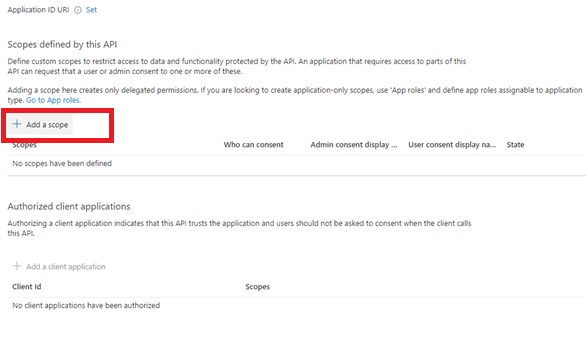
1. On Add a platform step (under Authentication link) – use redirect uri as http:/localhost:3000 (The port on which API server will be running)

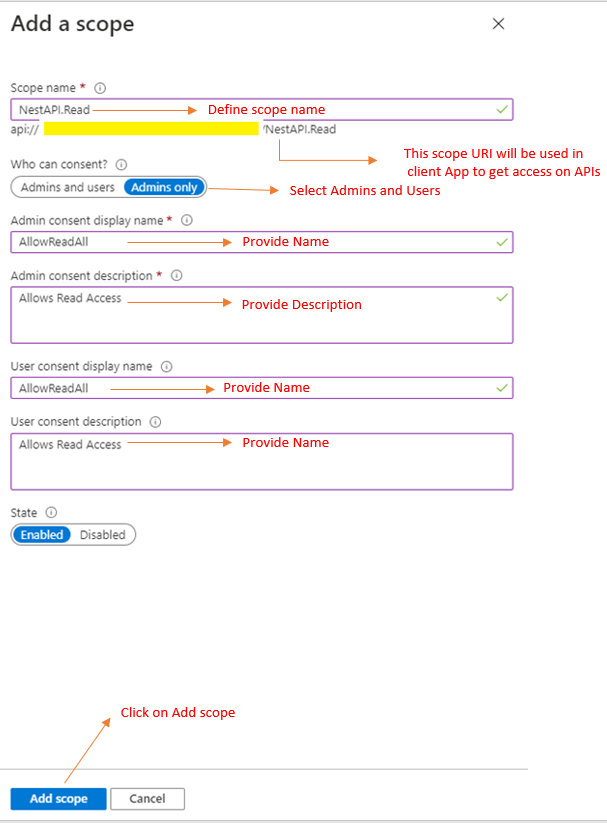


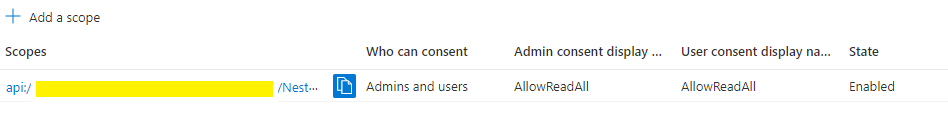
1. Under **Expose an API** section define a scope. By registering API and exposing it through scopes, you can provide permissions-based access to its resources to authorized users and client apps that access your API.

**For more detail** : <https://docs.microsoft.com/en-us/azure/active-directory/develop/quickstart-configure-app-expose-web-apis>



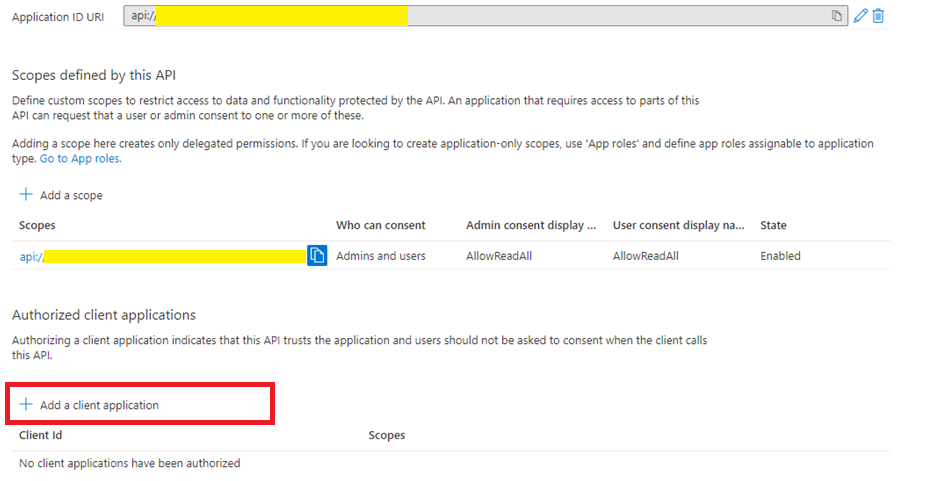


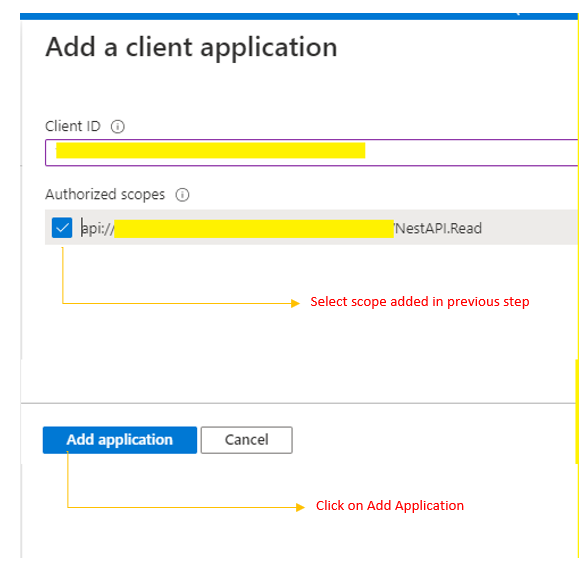


Scope Added successfully. Make a note of **scope** to use it in React application later.  


STEPS CONFIGURE REACT APP AS A CLIENT TO NEST Web API APP IN AAD

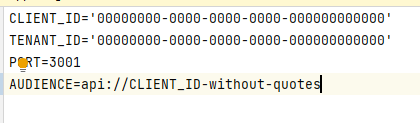
1. AAD à App Registrations à Nest à Expose An API à Add a Client Application





NESTJS APPLICATION CODE SET UP & EXECUTION

1. Repository Details:
   1. **Repo URL** : https://github.com/Monaswini/user-auth.git
   2. **Branch:** Master
2. Clone the repository and in the root folder directory run “npm install”
3. Update below values in **.env** file located in the root directory



1. Run “npm start” to run the application

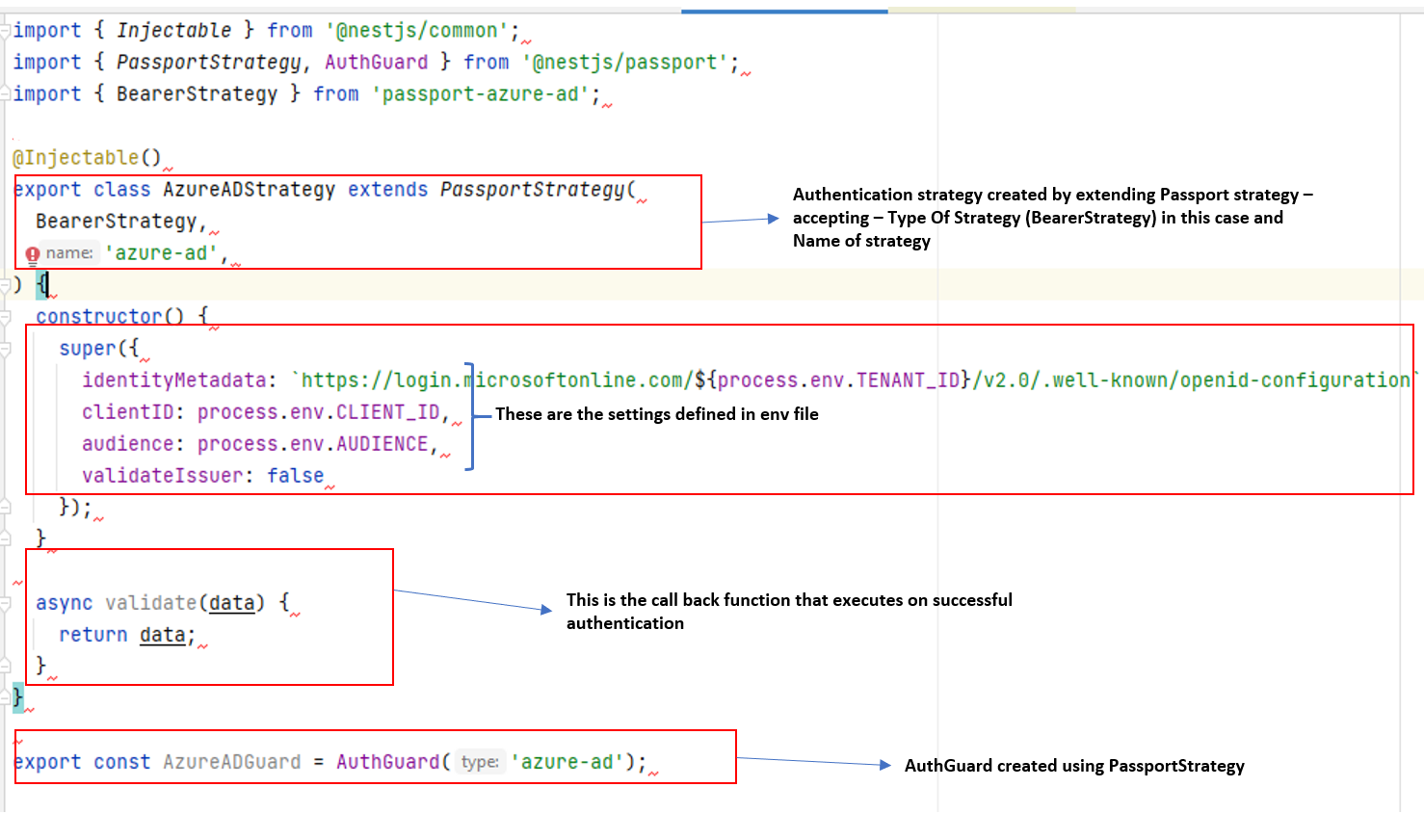
AUTHENTICATION IN NEST

In this implementation,

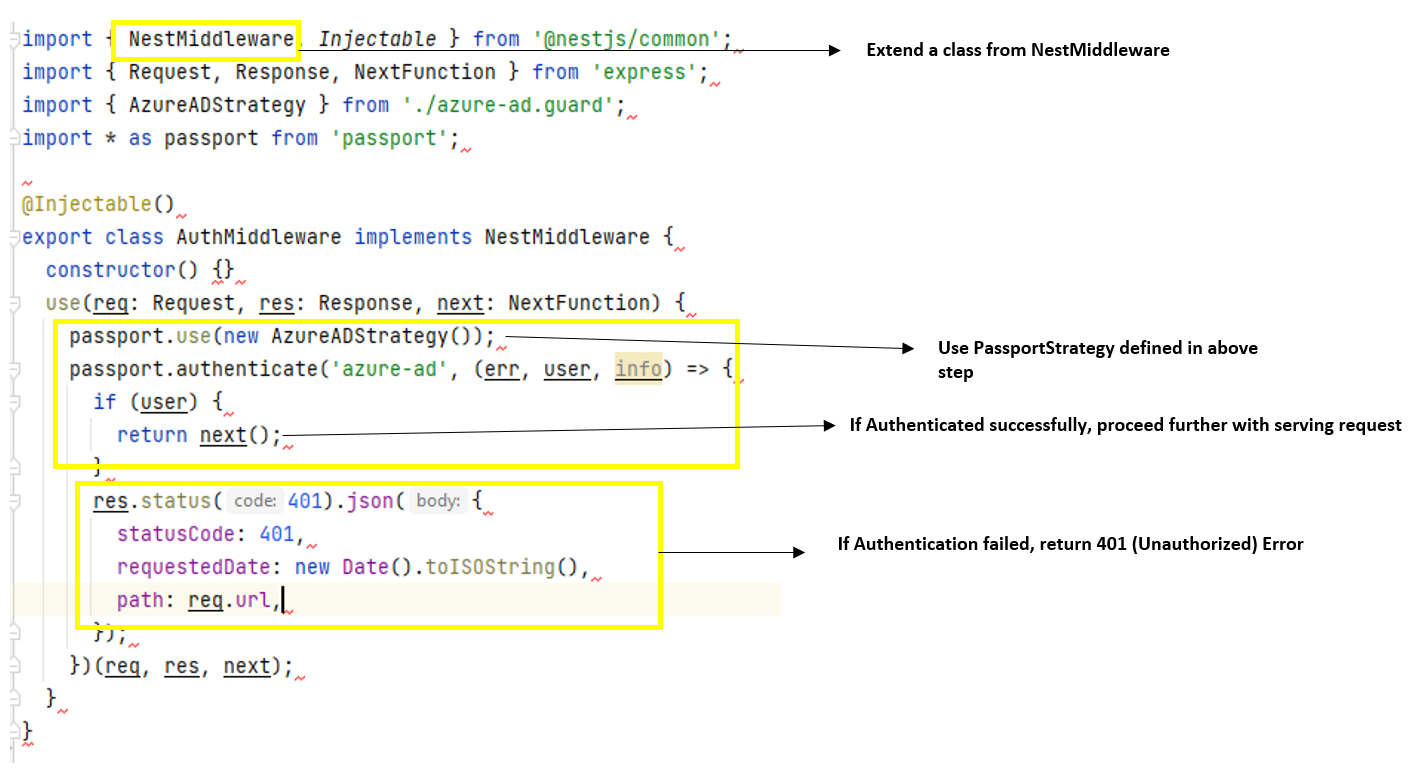
1. [**Passport**](https://github.com/jaredhanson/passport) – a very popular node.js authentication library is used to Authenticate a user.
2. [**passport-azure-ad**](https://www.npmjs.com/package/passport-azure-ad) - a collection of [**Passport**](https://github.com/jaredhanson/passport) Strategies to help you integrate with Azure Active Directory.

There are 3 steps needed to integrate passport:

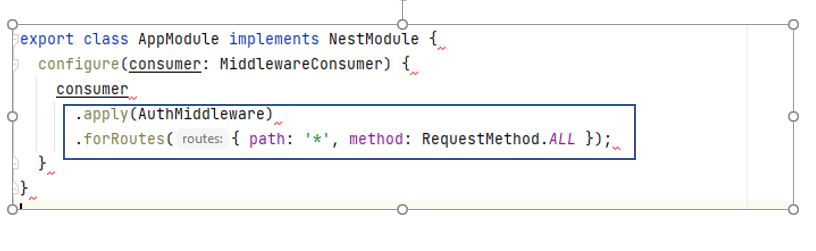
1. Set up a Passport Strategy – This implementation is using Bearer Strategy
2. Using Passport Strategy defined in step 1 construct an Auth-Guard



1. Construct a middleware using AuthGuard and Passport strategy defined in #1 and #2 to
   1. authenticate a request and proceed with api call if authenticated successfully OR return a 401 (unauthorized) error



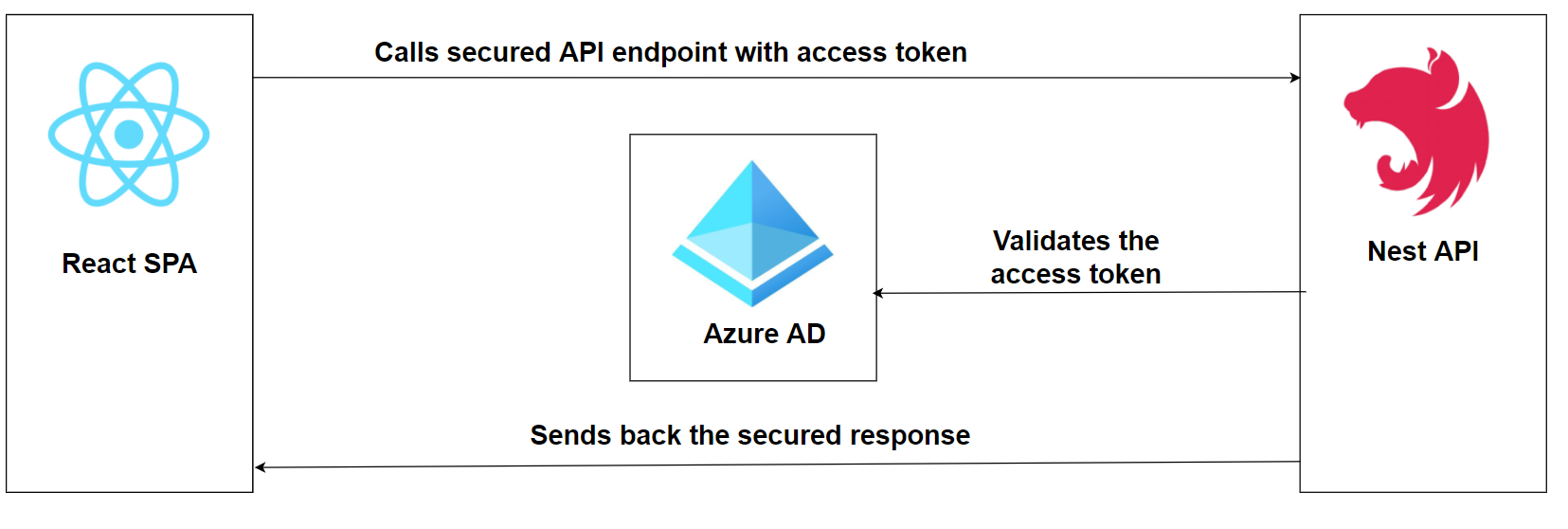
* 1. Apply this middleware globally for all routes



Fore more details, go through documentation at [**Passport**](https://github.com/jaredhanson/passport)

**REACT SPA CLIENT APP – CONFIGURATION TO CALL NEST API**

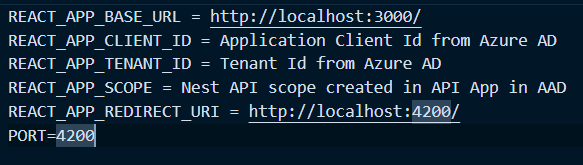
FLOW DIAGRAM BETWEEN REACT SPA à AAD à NESTJS API



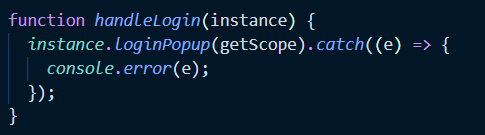
1. Set up config variables to set up communication with Nest API and configure scope to get access to the Web API
   1. Navigate to env-config.json file and Add 2 keys:

**NestAPIScope** – Scope added in NestAPI client app for ReactSPA application.

**NestAPIUrl**": "http://localhost:3001"



1. Use this scope when creating loginRequest on click of Login Button.



1. Run application to test integration with NestAPI

