Development of Login and Registration Feature using Auth0 and Microsoft Identity.

Introduction:

Auth0: It is a popular identity and access management (IAM) platform that provides solutions for authentication, authorization, and user management in web and mobile applications. It aims to simplify the process of adding secure user authentication and authorization to your applications, saving developers time and effort in implementing these critical functionalities.

Key features and components of Auth0 include:

1. **User Authentication:** Auth0 supports various authentication methods, including social logins (like Google, Facebook, and Twitter), username/password authentication, and multi-factor authentication. Developers can integrate Auth0 to enable users to sign up, log in, and manage their accounts securely.
2. **Authorization:** Auth0 helps manage user permissions and access control in applications. It supports role-based access control (RBAC) and fine-grained authorization policies, allowing developers to control what users can and cannot access within their applications.
3. **Identity Providers:** Auth0 integrates with numerous identity providers, allowing users to authenticate using their existing credentials from platforms such as Google, Microsoft, and others. This reduces the need for users to create new accounts for each application.
4. **Security Features:** Auth0 is designed with security in mind, providing features such as secure password storage, encryption, and protection against common security threats like cross-site scripting (XSS) and cross-site request forgery (CSRF).
5. **Single Sign-On (SSO**): Auth0 supports Single Sign-On, allowing users to log in once and access multiple applications without having to re-

enter their credentials. This enhances the user experience and simplifies authentication across different services.

6. **Customization and Branding:** Developers can customize the appearance and branding of the authentication pages to maintain a consistent user experience that aligns with their application's design.
7. **Analytics and Reporting:** Auth0 provides analytics and reporting tools to monitor user activity, track login attempts, and gain insights into the application's security and usage.
8. **Developer-Friendly APIs and SDKs:** Auth0 offers APIs and Software Development Kits (SDKs) for various programming languages, making it easy for developers to integrate and use Auth0 in their applications.
9. **Extensibility:** Auth0 allows developers to extend its functionality through custom scripts, hooks, and rules, providing flexibility in meeting specific business requirements.

Microsoft Identity: It refers to the set of services and technologies provided by Microsoft to manage and secure digital identities in applications and services. Microsoft's identity platform encompasses a range of tools, services, and standards designed to help developers integrate authentication, authorization, and user management into their applications.

Key components of Microsoft Identity include:

Azure Active Directory (Azure AD): Azure AD is Microsoft's cloud-based identity and access management service. It serves as the backbone for authenticating and authorizing users in applications and services. Azure AD supports various authentication methods, including username/password, multi-factor authentication, and integration with external identity providers.

Microsoft Authentication Library (MSAL):MSAL is a set of libraries and APIs that developers can use to integrate Microsoft Identity into their applications. MSAL supports multiple platforms and programming languages, including .NET, JavaScript, Python, and more.

Azure AD B2C (Business to Consumer): Azure AD B2C is a specialized service within Azure AD that focuses on providing identity and access management solutions for consumer-facing applications. It allows developers to build applications that can handle user authentication and authorization for a large number of external users.

Azure AD B2B (Business to Business): Azure AD B2B enables organizations to securely share their applications and services with external partners. It allows guest users to access resources within an organization's Azure AD, extending collaboration while maintaining security.

Microsoft Graph API: Microsoft Graph API provides a unified programmability model to access a wide range of Microsoft 365 services, including user and group management, calendar, emails, and more. It allows developers to leverage identity information and other Microsoft services in their applications.

Identity Providers: Microsoft Identity supports integration with various identity providers, including social identity providers (Microsoft Account, Google, Facebook) and other enterprise identity providers. This enables users to sign in using their existing credentials.

Identity Security and Compliance: Microsoft Identity includes features and tools for enhancing the security of identities, such as conditional access policies, Identity Protection, and reporting tools. These help organizations monitor and respond to potential security threats.

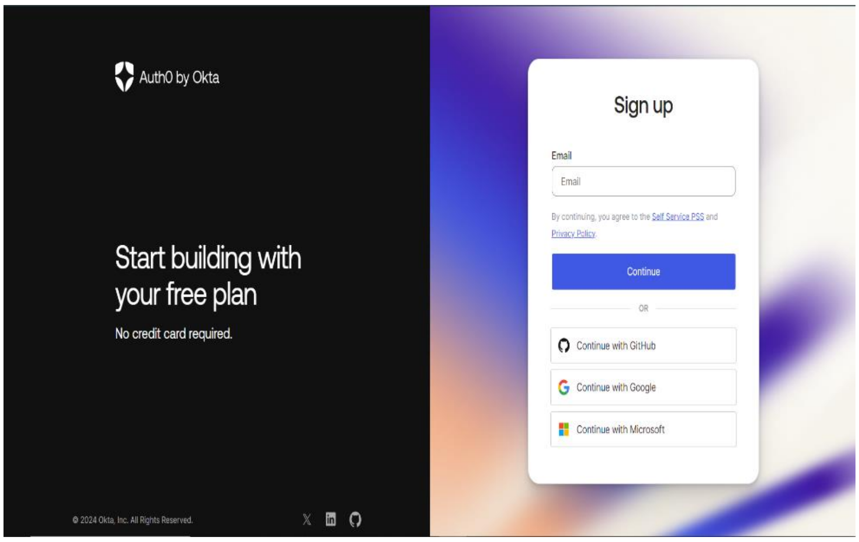
Active Directory (On-Premises): While Azure AD is cloud-based, Microsoft Identity also includes the traditional on-premises Active Directory. Organizations can use a combination of Azure AD and on-premises AD to manage identities across both cloud and on-premises environments.

Objective*:* This report outlines the comprehensive setup process of Auth0 and Microsoft Identity for integrating authentication into a web application, including key configurations and steps required to achieve seamless integration.

1. Auth0 Setup Process:

Step 1: Create an Auth0 Account

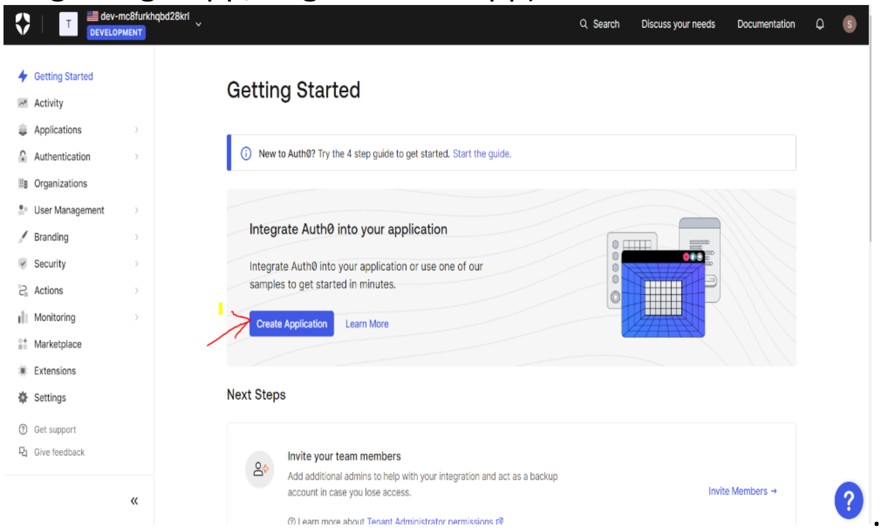
1. Visit the Auth0 website (https://auth0.com/) and sign up for an account.


Auth0 by Okta
Start building with your free plan
No credit card required.
© 2024 Okta, Inc. All Rights Reserved.
Sign up
Email
Email
By continuing, you agree to the Self Service PSS and Privacy Policy.
Continue
OR
Continue with GitHub
Continue with Google
Continue with Microsoft

2. Upon successful registration, log in to the Auth0 Dashboard.

Step 2: Create a New Application in Auth0

1. In the Auth0 Dashboard, navigate to the "Applications" tab.
2. Click on "Create Application" and choose the type of application (e.g., Single Page App, Regular Web App)



Getting Started

New to Auth0? Try the 4 step guide to get started. Start the guide.

Integrate Auth0 into your application

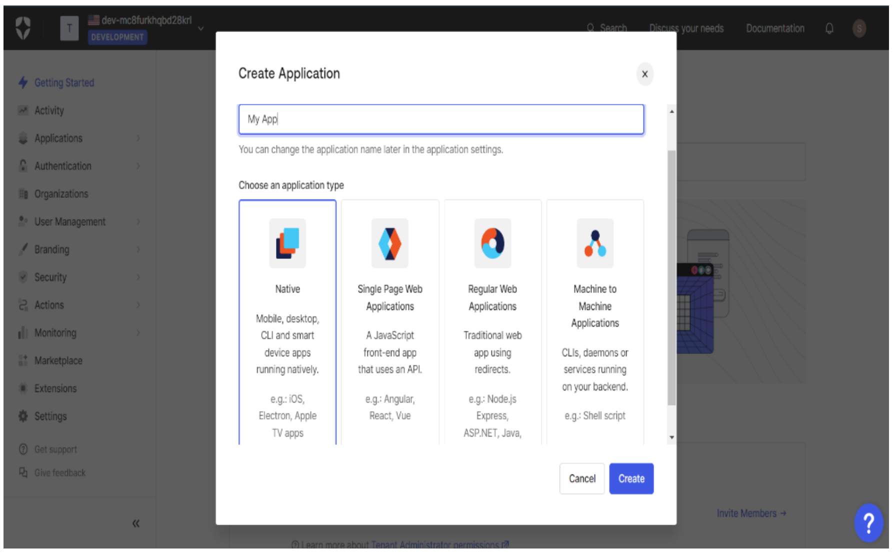
Integrate Auth0 into your application or use one of our samples to get started in minutes.

Next Steps

Invite your team members

Add additional admins to help with your integration and act as a backup account in case you lose access.

Learn more about Tenant Administrator permissions



You can change the application name later in the application settings.

Choose an application type

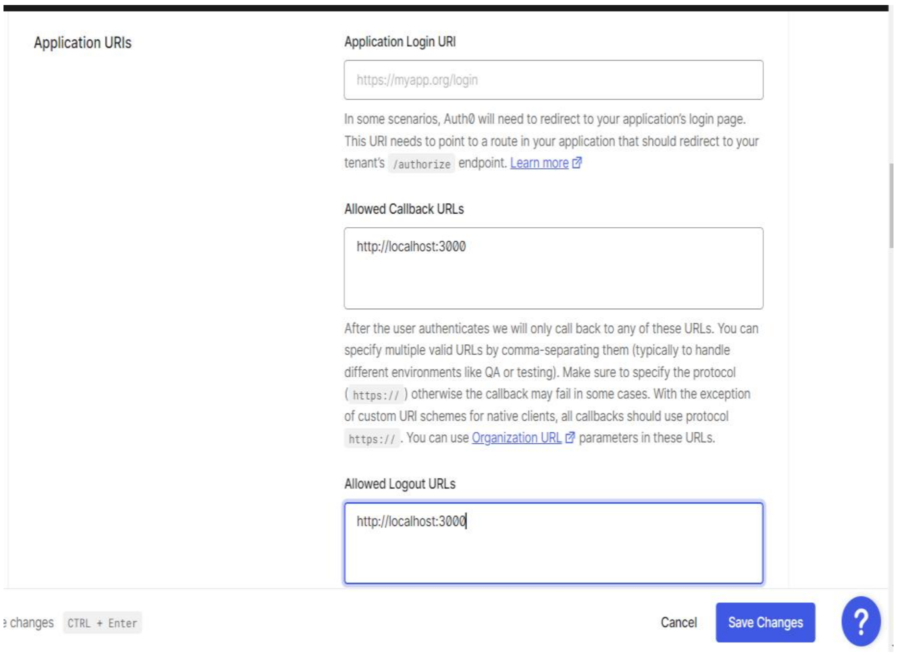
Native — Mobile, desktop, CLI and smart device apps running natively. e.g.: iOS, Electron, Apple TV apps

Single Page Web Applications — A JavaScript front-end app that uses an API. e.g.: Angular, React, Vue

Regular Web Applications — Traditional web app using redirects. e.g.: Node.js Express, ASP.NET, Java,

Machine to Machine Applications — CLIs, daemons or services running on your backend. e.g.: Shell script

3. Configure application settings, such as name and allowed callback URLs.


Application URIs

Application Login URI

https://myapp.org/login

In some scenarios, Auth0 will need to redirect to your application's login page. This URI needs to point to a route in your application that should redirect to your tenant's `/authorize` endpoint. Learn more

Allowed Callback URLs

http://localhost:3000

After the user authenticates we will only call back to any of these URLs. You can specify multiple valid URLs by comma-separating them (typically to handle different environments like QA or testing). Make sure to specify the protocol (`https://`) otherwise the callback may fail in some cases. With the exception of custom URI schemes for native clients, all callbacks should use protocol `https://`. You can use Organization URL parameters in these URLs.

Allowed Logout URLs

http://localhost:3000

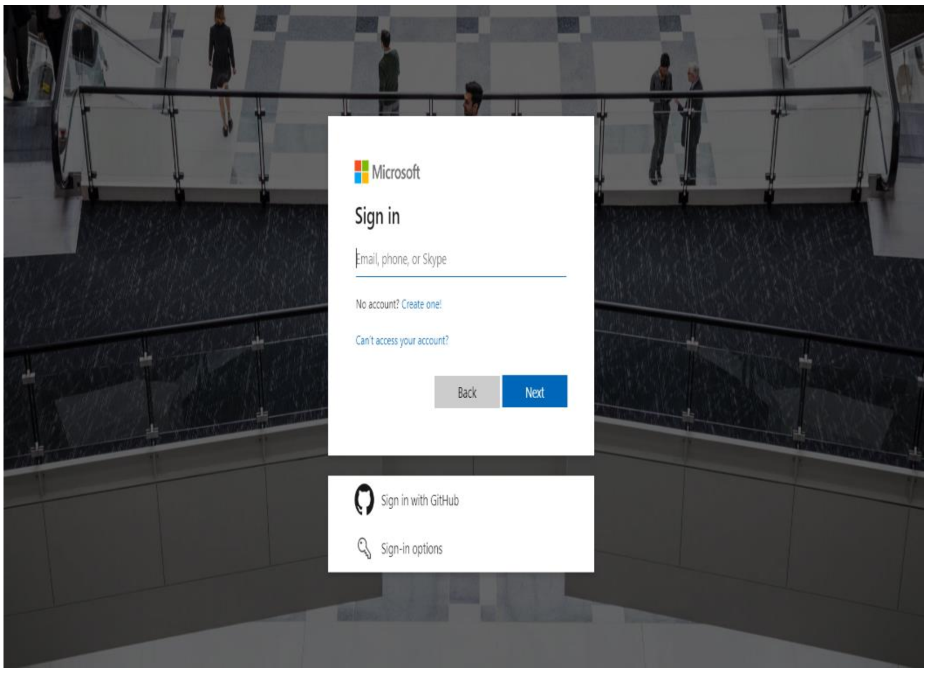
Step 3: Obtain Auth0 Credentials

1. After creating the application, go to the "Settings" tab.
2. Note down the "Client ID" and "Client Secret" – these credentials are crucial for the web application configuration.

2. Microsoft Identity Setup Process:

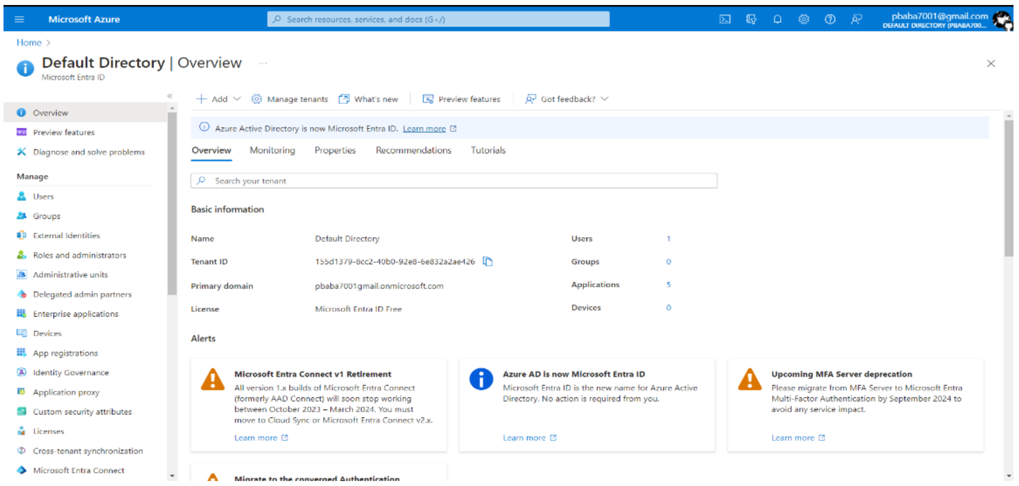
Step 1: Access the Azure Portal

1. Log in to the Azure portal (https://portal.azure.com/).



Step 2: Create a New App Registration in Azure AD

1. Navigate to "Azure Active Directory" > "App registrations."



Default Directory | Overview

Microsoft Entra ID

Azure Active Directory is now Microsoft Entra ID. Learn more

Basic information

Name	Default Directory	Users	1
Tenant ID	155d1379-8cc2-40b0-92e8-6e832a2ae426	Groups	0
Primary domain	pbaba7001gmail.onmicrosoft.com	Applications	5
License	Microsoft Entra ID Free	Devices	0

Alerts

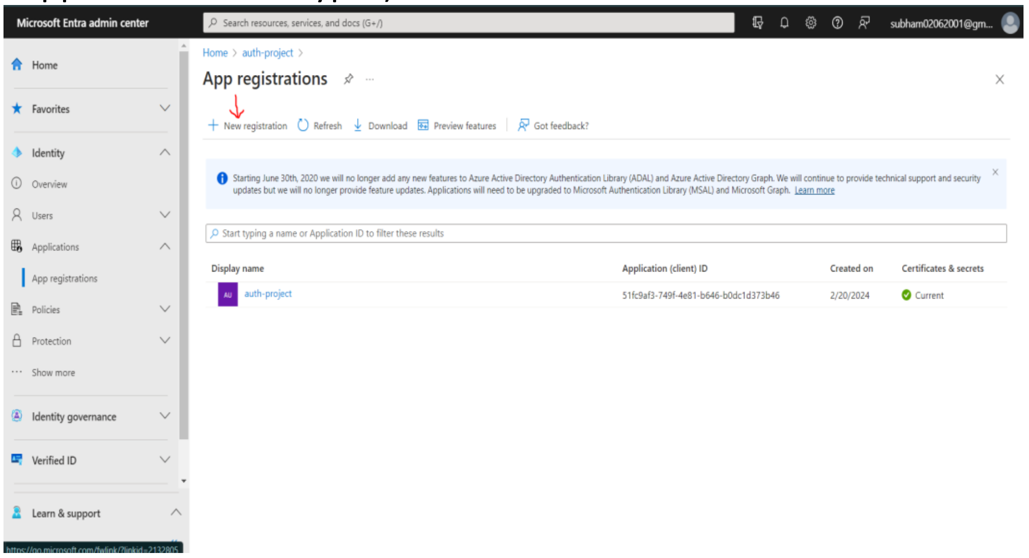
Microsoft Entra Connect v1 Retirement
All version 1.x builds of Microsoft Entra Connect (formerly AAD Connect) will soon stop working between October 2023 – March 2024. You must move to Cloud Sync or Microsoft Entra Connect v2.x.
Learn more

Azure AD is now Microsoft Entra ID
Microsoft Entra ID is the new name for Azure Active Directory. No action is required from you.
Learn more

Upcoming MFA Server deprecation
Please migrate from MFA Server to Microsoft Entra Multi-Factor Authentication by September 2024 to avoid any service impact.
Learn more

Migrate to the converged Authentication

2. Click on "New registration" and fill in required details (e.g., Name, Supported account types).

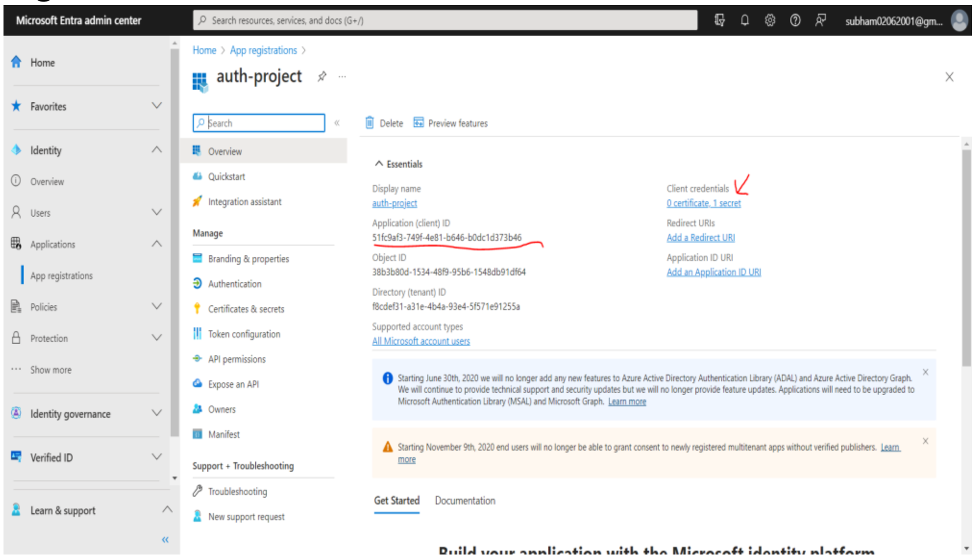


App registrations

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more

Display name	Application (client) ID	Created on	Certificates & secrets
AU auth-project	51fc9af3-749f-4e81-b646-b0dc1d373b46	2/20/2024	✓ Current

3. Note down the "Application (client) ID" and "Directory (tenant) ID" upon registration.



Essentials

Display name: auth-project

Application (client) ID: 51fc9af3-749f-4e81-b646-b0dc1d373b46

Object ID: 38b3b80d-1534-48f9-95b6-1548db91df64

Directory (tenant) ID: f8cdef31-a31e-4b4a-93e4-5f571e91255a

Supported account types: All Microsoft account users

Client credentials: 0 certificate, 1 secret

Redirect URIs: Add a Redirect URI

Application ID URI: Add an Application ID URI

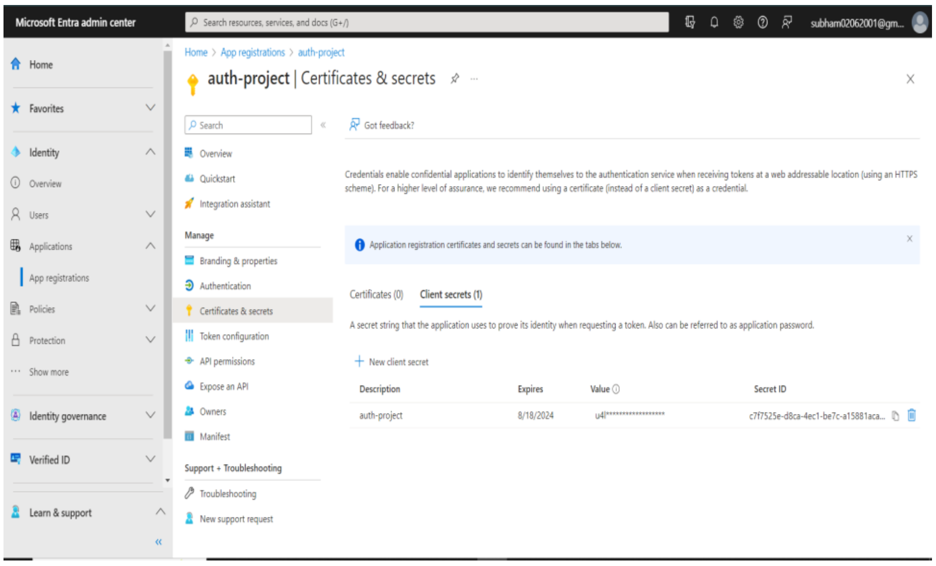
Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more

Starting November 9th, 2020 end users will no longer be able to grant consent to newly registered multitenant apps without verified publishers. Learn more

Build your application with the Microsoft identity platform

Step 3: Generate Client Secret

1. In the App Registration settings, go to "Certificates & Secrets."
2. Create a new client secret and note down the generated secret value.



Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

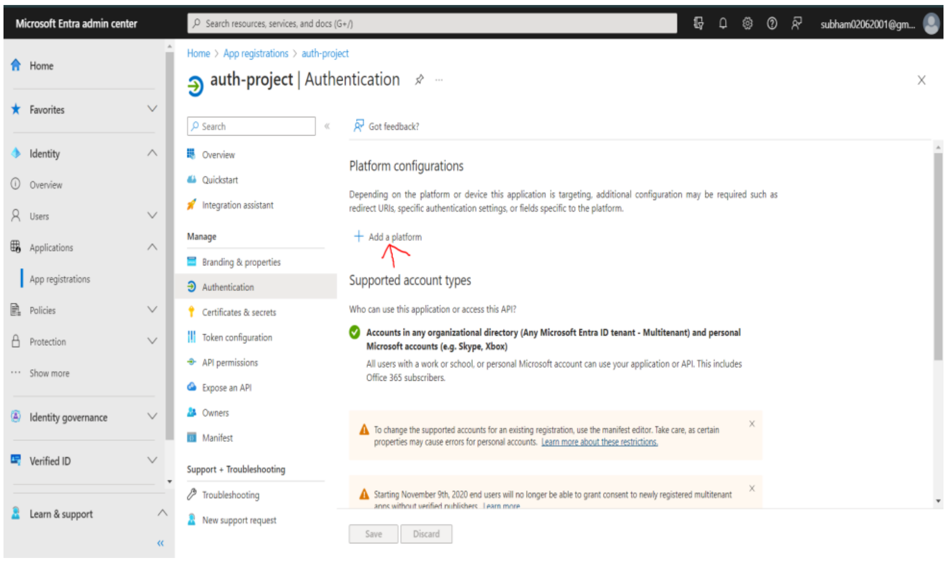
Application registration certificates and secrets can be found in the tabs below.

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

Description	Expires	Value	Secret ID
auth-project	8/18/2024	u4l******************	c7f7525e-d8ca-4ec1-be7c-a15881aca...

Step 4: Configure Authentication Redirect URIs

1. Under App Registration in the Azure portal, navigate to "Authentication."



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.

Supported account types

Who can use this application or access this API?

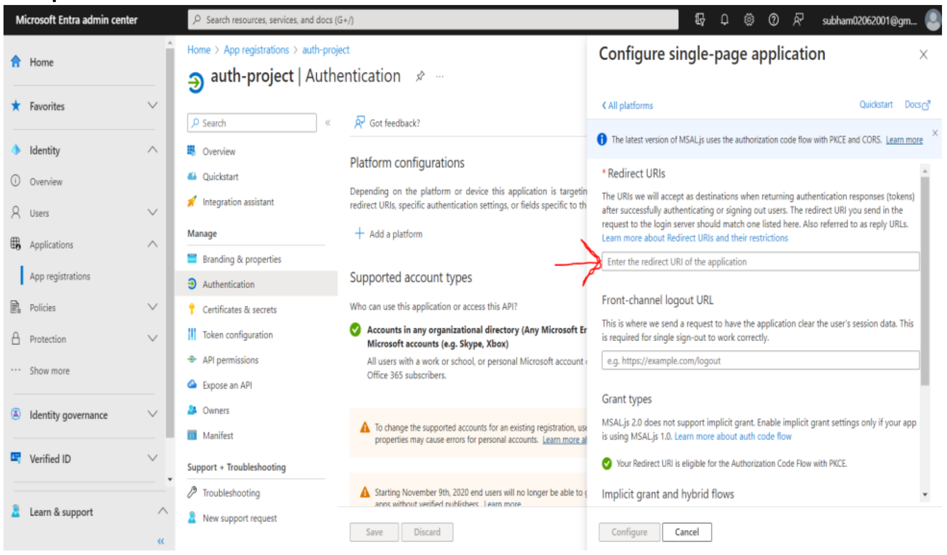
Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

All users with a work or school, or personal Microsoft account can use your application or API. This includes Office 365 subscribers.

To change the supported accounts for an existing registration, use the manifest editor. Take care, as certain properties may cause errors for personal accounts. Learn more about these restrictions.

Starting November 9th, 2020 end users will no longer be able to grant consent to newly registered multitenant apps without verified publishers. Learn more

2. Configure the redirect URIs where Microsoft will send authentication responses


Microsoft Entra admin center
Search resources, services, and docs (G+/)
subham02062001@gm...
Home > App registrations > auth-project
auth-project | Authentication
All platforms
Quickstart
Docs
Enter the redirect URI of the application
e.g. https://example.com/logout
Configure
Cancel

Configure single-page application

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. The redirect URI you send in the request to the login server should match one listed here. Also referred to as reply URLs. Learn more about Redirect URIs and their restrictions

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.

Grant types

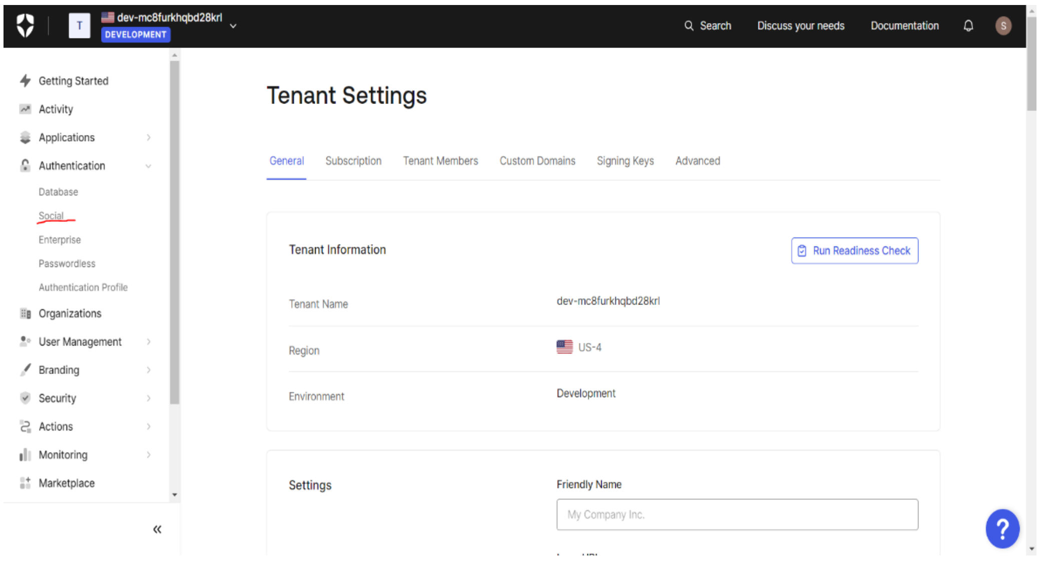
MSAL.js 2.0 does not support implicit grant. Enable implicit grant settings only if your app is using MSAL.js 1.0. Learn more about auth code flow

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

Implicit grant and hybrid flows

Step 5: Configure Microsoft Identity in Auth0

1. Return to the Auth0 Dashboard.
2. Navigate to "Authentication" > "Social" > "Microsoft."



Tenant Settings

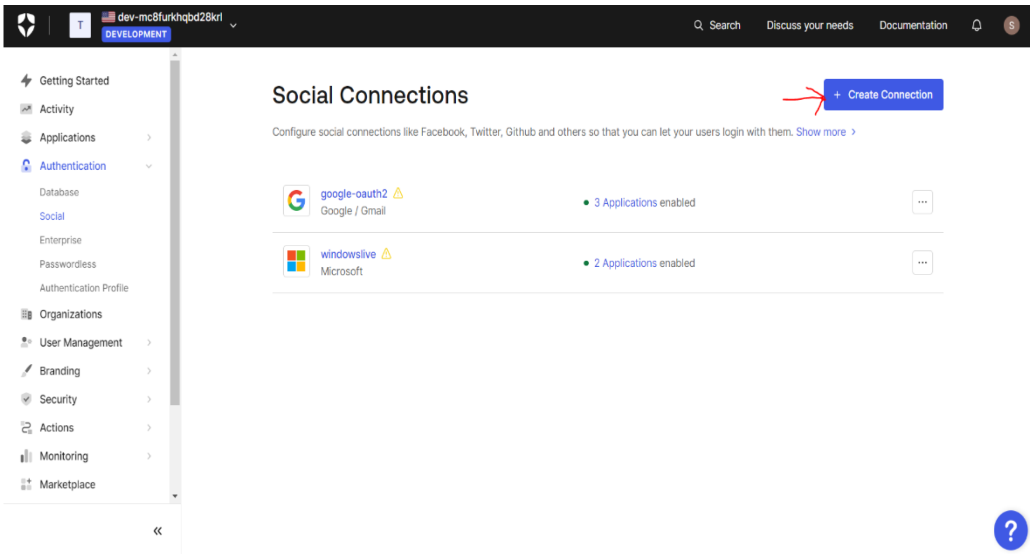
Tenant Information

Tenant Name	dev-mc8furkhqbd28krl
Region	US-4
Environment	Development

Settings

Friendly Name

My Company Inc.

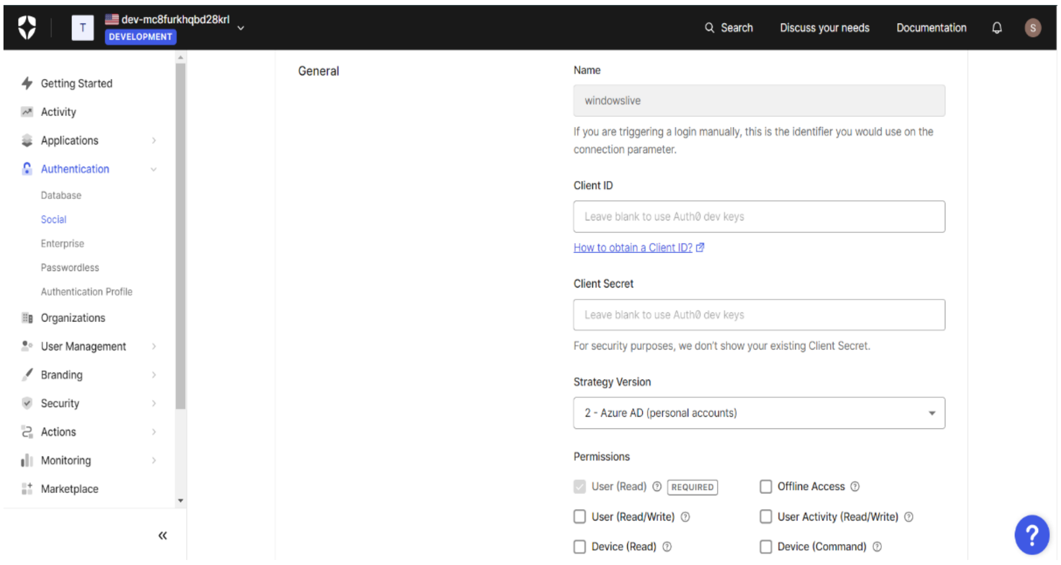


Social Connections

Configure social connections like Facebook, Twitter, Github and others so that you can let your users login with them. Show more ›

google-oauth2 ⚠ Google / Gmail	● 3 Applications enabled
windowslive ⚠ Microsoft	● 2 Applications enabled

3. Enable the Microsoft connection and provide the Application (client) ID, Directory (tenant) ID, and Client Secret obtained from Azure.



General

Name

windowslive

If you are triggering a login manually, this is the identifier you would use on the connection parameter.

Client ID

Leave blank to use Auth0 dev keys

How to obtain a Client ID?

Client Secret

Leave blank to use Auth0 dev keys

For security purposes, we don't show your existing Client Secret.

Strategy Version

2 - Azure AD (personal accounts)

Permissions

- [✓] User (Read) REQUIRED
- [ ] Offline Access
- [ ] User (Read/Write)
- [ ] User Activity (Read/Write)
- [ ] Device (Read)
- [ ] Device (Command)

Step 6: Test Connection in Auth0

1. Use the Auth0 Dashboard to test the Microsoft connection to ensure successful integration.

Key Configurations for Auth0:

1. **Client ID:**
    - *Definition:* The Client ID is a unique identifier assigned to each Auth0 application.
    - *Purpose:* It serves as a public identifier for the application and is used during the authentication process to associate the request with the correct Auth0 application.
    - *Implementation:* Developers need to include the Client ID in the application's configuration to establish a secure connection with Auth0.
2. **Client Secret:**
    - *Definition:* The Client Secret is a confidential key known only to the application and Auth0.
    - *Purpose:* It is used to authenticate the application with Auth0 during the process of obtaining tokens. This ensures that only authorized applications can interact with Auth0 on behalf of the users.
    - *Implementation:* Developers must securely store and manage the Client Secret on the server-side, preventing exposure to unauthorized entities.
3. **Allowed Callback URLs:**
    - *Definition:* These are the URLs to which Auth0 can redirect users after successful authentication.
    - *Purpose:* Specifying allowed callback URLs prevents attackers from redirecting users to malicious endpoints after authentication. It ensures that the authentication response is directed only to trusted locations.

- *Implementation:* Developers configure these URLs in the Auth0 Dashboard to define the valid callback destinations for their application.

Key Configurations for Microsoft Identity:

1. **Application (client) ID:**
    - *Definition:* The Application ID (also known as the Client ID) is a unique identifier assigned to an application registered in Azure Active Directory (Azure AD).
    - *Purpose:* It uniquely identifies the application when interacting with Microsoft Identity services, establishing a connection between the application and the Azure AD instance.
    - *Implementation:* Developers obtain this ID from the Azure portal after registering their application.

2. **Directory (tenant) ID:**
    - *Definition:* The Directory ID (also known as the Tenant ID) is a unique identifier for the Azure AD tenant associated with the application.
    - *Purpose:* It specifies the Azure AD instance to which the application belongs, ensuring correct routing of authentication requests within the specified tenant.
    - *Implementation:* Developers retrieve this ID from the Azure portal after registering the application, and it is used as a parameter in authentication requests.

3. **Client Secret:**
    - *Definition:* The Client Secret is a secure key used for authenticating the application with Azure AD.

- *Purpose:* It ensures that only authorized applications can request tokens and access resources on behalf of users. It acts as a shared secret between the application and Azure AD.
- *Implementation:* Developers generate a Client Secret in the Azure portal during the application registration process, and it must be securely stored on the server-side.

Integrating Auth0 with Microsoft Identity involves several steps to ensure a seamless and secure authentication process. Here's a detailed guide on the steps required:

1. Create an Auth0 Account and Application:

- Sign up for an Auth0 account if not already done.
- In the Auth0 Dashboard, create a new application.
- Note down the Client ID and Client Secret provided by Auth0 for the application.

2. Set Up Microsoft Identity in the Azure Portal:

- Log in to the Azure portal (https://portal.azure.com/).
- Navigate to "Azure Active Directory" > "App registrations."
- Create a new app registration for your application.
- Note down the Application (client) ID, Directory (tenant) ID, and create a Client Secret.

3. Configure Authentication Redirect URIs:

- In the Azure portal, under your app registration, navigate to "Authentication."
- Configure the Redirect URIs where Microsoft Identity will send authentication responses. Ensure these match the callback URLs configured in Auth0.

4. Configure Microsoft Identity in Auth0:

- In the Auth0 Dashboard, navigate to "Connections" > "Social" > "Microsoft."
- Enable the Microsoft connection.
- Provide the Application (client) ID, Directory (tenant) ID, and Client Secret obtained from the Azure portal.

5. Test the Connection:

- Use the Auth0 Dashboard to test the Microsoft connection to ensure successful integration.
- Verify that authentication requests initiated by Auth0 are correctly processed by Microsoft Identity.

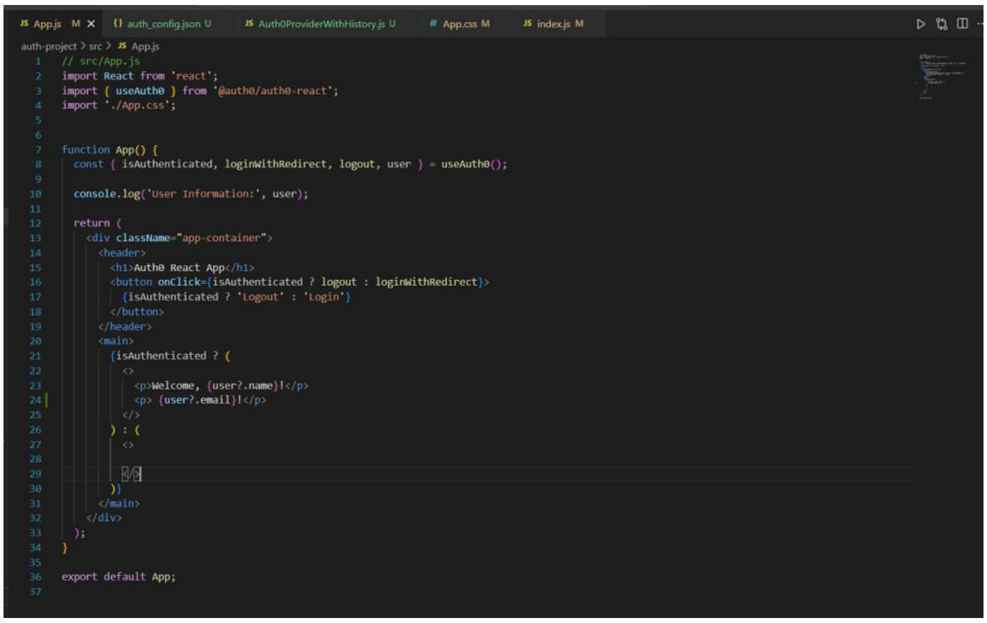
6. Update Web Application Configuration:

- In your web application's authentication configuration, set Auth0 as the primary authentication provider.
- Ensure the application is configured to redirect users to Auth0 for authentication.

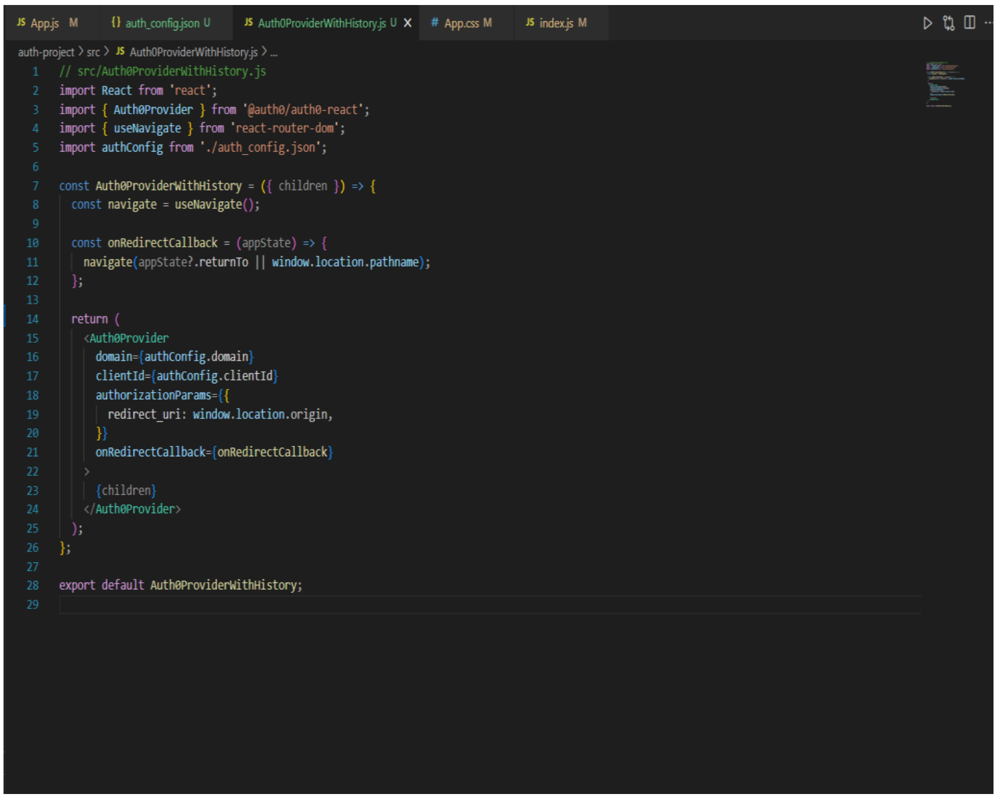
7. Implement Login and Registration UI:

- Develop the frontend interfaces for login and registration within your web application.

  App.js



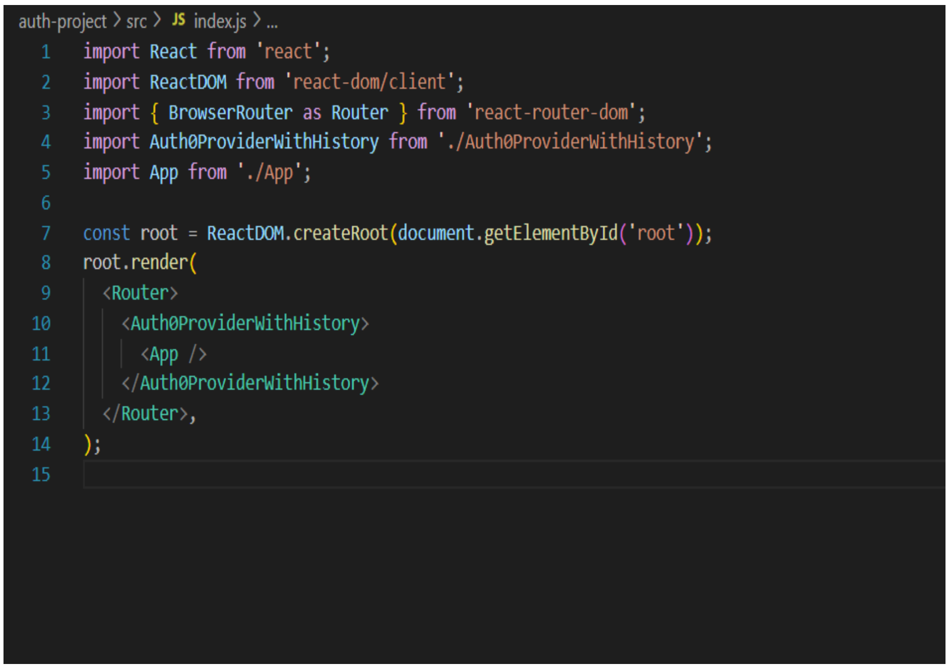
```
// src/App.js
import React from 'react';
import { useAuth0 } from '@auth0/auth0-react';
import './App.css';


function App() {
  const { isAuthenticated, loginWithRedirect, logout, user } = useAuth0();

  console.log('User Information:', user);

  return (
    <div className="app-container">
      <header>
        <h1>Auth0 React App</h1>
        <button onClick={isAuthenticated ? logout : loginWithRedirect}>
          {isAuthenticated ? 'Logout' : 'Login'}
        </button>
      </header>
      <main>
        {isAuthenticated ? (
          <>
            <p>Welcome, {user?.name}!</p>
            <p> {user?.email}!</p>
          </>
        ) : (
          <>

          </>
        )}
      </main>
    </div>
  );
}

export default App;
```


App.js M
auth_config.json U
Auth0ProviderWithHistory.js U
App.css M
index.js M
auth-project > src > Auth0ProviderWithHistory.js > ...

```
// src/Auth0ProviderWithHistory.js
import React from 'react';
import { Auth0Provider } from '@auth0/auth0-react';
import { useNavigate } from 'react-router-dom';
import authConfig from './auth_config.json';

const Auth0ProviderWithHistory = ({ children }) => {
  const navigate = useNavigate();

  const onRedirectCallback = (appState) => {
    navigate(appState?.returnTo || window.location.pathname);
  };

  return (
    <Auth0Provider
      domain={authConfig.domain}
      clientId={authConfig.clientId}
      authorizationParams={{
        redirect_uri: window.location.origin,
      }}
      onRedirectCallback={onRedirectCallback}
    >
      {children}
    </Auth0Provider>
  );
};

export default Auth0ProviderWithHistory;
```

AuthProviderHistory.js

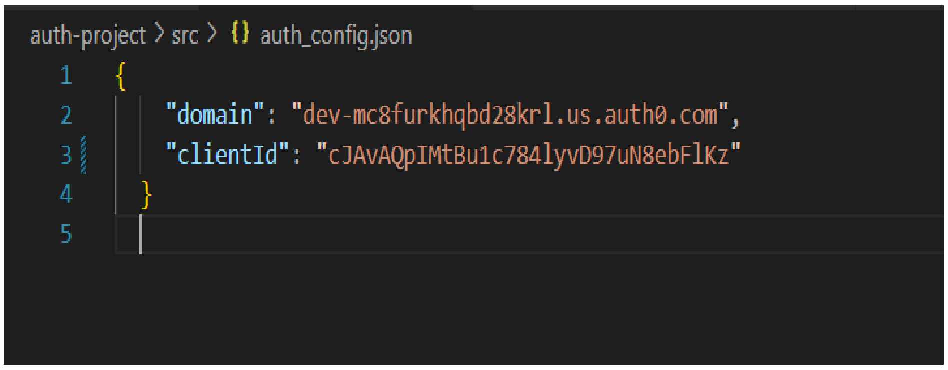
Index.js



auth-project > src > JS index.js > ...

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { BrowserRouter as Router } from 'react-router-dom';
import Auth0ProviderWithHistory from './Auth0ProviderWithHistory';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <Router>
    <Auth0ProviderWithHistory>
      <App />
    </Auth0ProviderWithHistory>
  </Router>,
);
```

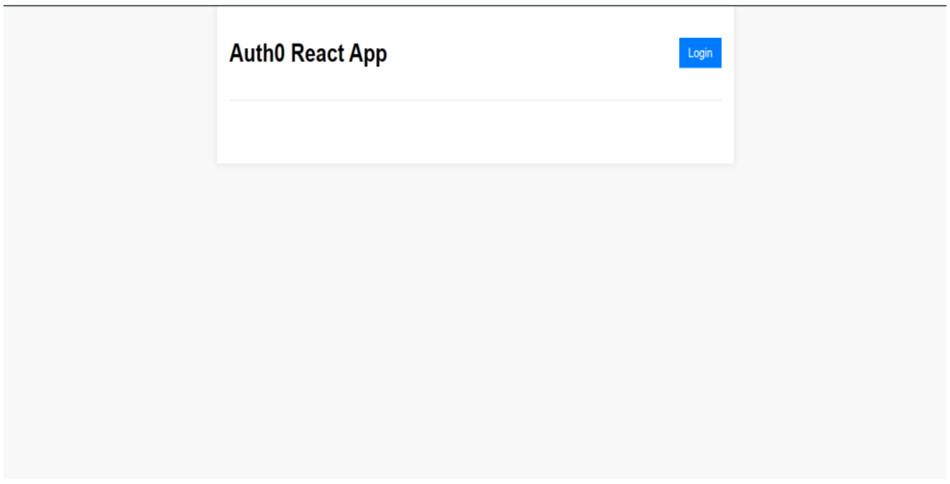
auth_config.json



auth-project > src > {} auth_config.json

```
{
    "domain": "dev-mc8furkhqbd28krl.us.auth0.com",
    "clientId": "cJAvAQpIMtBu1c784lyvD97uN8ebFlKz"
  }
```

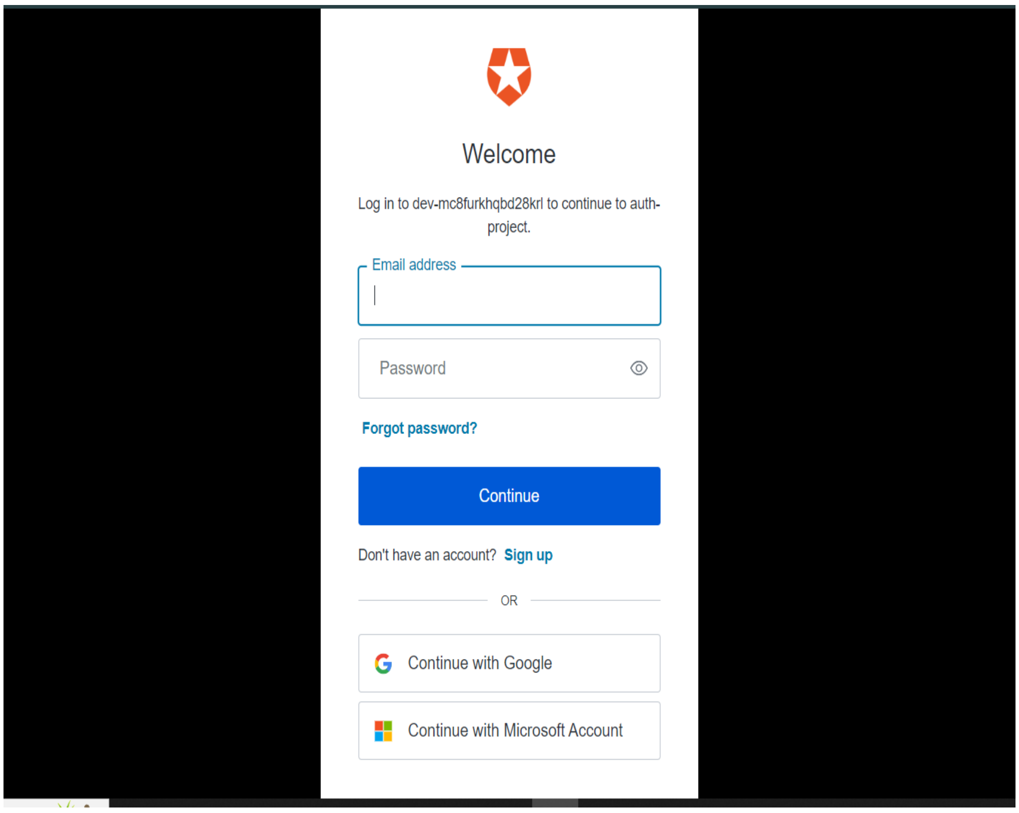
UI



- Implement logic to initiate the authentication flow when users attempt to log in or register.

8. Securely Manage Auth0 and Microsoft Identity Tokens:

- Implement logic on the server-side to securely handle and validate the tokens received from Auth0 and Microsoft Identity during the authentication process.

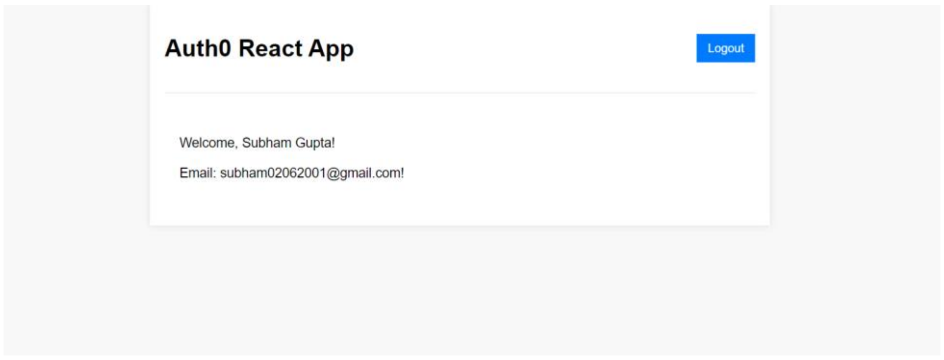


Welcome

Log in to dev-mc8furkhqbd28krl to continue to auth-project.

- Use these tokens to identify and authorize users within your application.

9. Test the End-to-End Flow:



Auth0 React App

Welcome, Subham Gupta!

Email: subham02062001@gmail.com!

- Test the entire authentication flow from your web application, including login, registration, and logout.
- Ensure that users can successfully authenticate using their Microsoft accounts.

10. Document the Configuration:

- Document the key configurations, including Client IDs, Client Secrets, and Redirect URIs for both Auth0 and Microsoft Identity.
- Maintain detailed documentation for future reference and for onboarding new developers.

11. Prepare for Production:

- Ensure that the application is configured securely for production use.
- Monitor and manage any necessary updates or changes to configurations.

Conclusion: In conclusion, successfully setting up Auth0 and Microsoft Identity for our web application's authentication is a big achievement. This process ensures that our authentication solution is both secure and user-friendly. We went through detailed steps, configuring essential elements like Client IDs, Client Secrets, and Redirect URIs, to build a strong foundation for smooth integration.

By using Auth0 as the main authentication provider and Microsoft Identity as the identity provider, we've created a straightforward authentication process. This not only improves how users interact with our application but also makes our application more secure.

Our documentation provides developers with a useful guide, detailing each step and important configurations for future work. The user guide makes it easy for end-users to navigate our login and registration features, ensuring a positive experience.

During the live demo, we showed how everything works together, emphasizing the successful collaboration between Auth0 and Microsoft Identity. Sharing insights about challenges faced and overcome provides valuable lessons for future development.

To sum up, Auth0 and Microsoft Identity together create a dependable and effective authentication solution. This lays the groundwork for a secure and user-friendly web application. Our commitment is to adhere to best practices and deliver a superior authentication experience for both developers and end-users.

References:

- Auth Docs(https://auth0.com/docs)
- Microsoft Identity Docs (https://learn.microsoft.com/en-us/entra/identity-platform/)