Auth0 Integration Guide - SPA and API:

React:

To integrate Auth0 authentication into your React app u must first head to https://auth0.com/signup and create an account. Once signed up head to **Applications** on the left sidebar and then from the **Applications** sub menu press Applications. From there create a Single-page application (SPA). Fill out the form by giving it a name (E.g. CinemaBookingFrontend) and fillout the necessary urls.

Official guide:
https://developer.auth0.com/resources/code-samples/spa/react/basic-authentication/v17-javascript-react-router-5

The urls are:

1. **Allowed callback urls**: A comma separated list of allowed urls Auth0 can redirect to upon successful login
2. **Allowed callback urls:** A comma separated list of allowed urls Auth0 can redirect to upon logout
3. **Allowed callback urls:** A comma separated list of allowed origins, that can make requests to the authentication app.

After this defining the urls press **Save Changes.**

Head into your React project and create .env at the root if one doesnt already exists. Define the environment variables:

```
REACT_APP_API_SERVER_URL=your_api_url
REACT_APP_AUTH0_DOMAIN=auth0_domain_value
REACT_APP_AUTH0_CLIENT_ID=your_client_value
REACT_APP_AUTH0_CALLBACK_URL=callback_url
REACT_APP_AUTH0_AUDIENCE=server_api_audience_value
```

An example of the values could be:

```
REACT_APP_API_SERVER_URL=http://api.cinema-booking.com
REACT_APP_AUTH0_DOMAIN=dev-kyyqmmxci28c240u.us.auth0.com
REACT_APP_AUTH0_CLIENT_ID=vdTwJxDFGL3mAW0xm6OJYPF0o8XIAdj
REACT_APP_AUTH0_CALLBACK_URL=http://api.cinema-booking.com/landing-page
REACT_APP_AUTH0_AUDIENCE=http://api.cinema-booking.com
```

The AUTH0_DOMAIN and CLIENT_ID value can be found by heading to your application using the Auth0 Dashboard then go to the **Settings tab** and then under the ‘basic information’.

The callback can be found under ‘application URIs’ under the same tab by scrolling down a bit.

For now leave the AUTH0_AUDIENCE value empty. We will create that later in the guide.

Head back your React project and install the Auth0 SDK:

```
# Using npm
npm install @auth0/auth0-react

# Using yarn
yarn add @auth0/auth0-react
```

Then create a file called Auth0Provider and copy paste this code:

```
import { Auth0Provider, AppState } from "@auth0/auth0-react";
import { ReactNode } from "react";
import { useNavigate } from "react-router-dom";

interface Auth0ProviderWithNavigateProps {
 children: ReactNode;
}

const Auth0ProviderWithNavigate = ({
 children,
}: Auth0ProviderWithNavigateProps) => {
 const navigate = useNavigate();

 const domain = process.env.REACT_APP_AUTH0_DOMAIN;
 const clientId = process.env.REACT_APP_AUTH0_CLIENT_ID;
 const redirectUri = process.env.REACT_APP_AUTH0_CALLBACK_URL;
 const audience = process.env.REACT_APP_AUTH0_AUDIENCE;

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

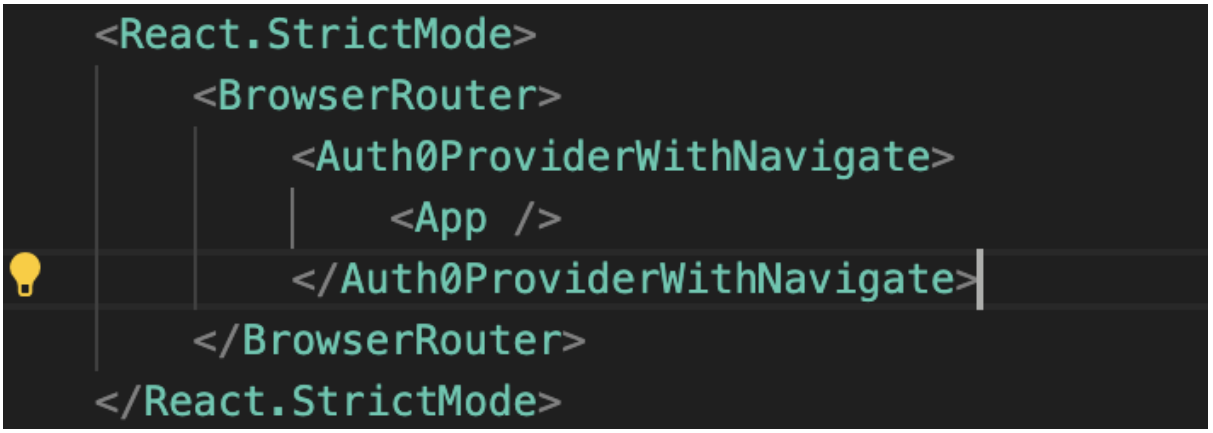
```
 if (!(domain && clientId && redirectUri && audience)) {
   return null;
 }



 return (
   <Auth0Provider
     domain={domain}
     clientId={clientId}
     authorizationParams={{
       audience: audience,
       redirect_uri: redirectUri,
     }}
     onRedirectCallback={onRedirectCallback}
   >
     {children}
   </Auth0Provider>
 );
};

export default Auth0ProviderWithNavigate
```

After that head to the place in your React project where your <App/> component is defined. This could be index.tsx. Wrap the app component with the Auth provider from above. The Auth provider must be a child or grandchild to the <BrowserRouter>:



Now we're ready to use the useAuth0 hooks inside our react application to handle login, sign out and sign ins.

For your login button assign this handler to the button's 'onClick' attribute:
To access objects, properties and function from the useAuth0 Hooks simply add this to your component:

```
const { isAuthenticated, user, loginWithRedirect, logout, getAccessTokenSilently } = useAuth0();
```

After this i can attach the loginWithRedirect function to my handleLogin handler:

```
const handleLogin = async () => {
  await loginWithRedirect({
    appState: {
    },
  });
};
```

And then use it on my login button:

```
    <Link
      onClick={handleLogin}
      fontSize="25px"
      fontWeight="500"
      color={textLight}
      _hover={{ color: crispYellow }}
      id="login-button"
    >
      Login
    </Link>
```

Functions to handle signup and logins:

```
const handleSignUp = async () => {
  await loginWithRedirect({
    appState: {
    },
    authorizationParams: {
      screen_hint: "signup",
    },
  });
};

const handleLogout = () => {
  logout({
    logoutParams: {
      returnTo: window.location.origin,
    },
  });
};
```

Using loginWithRedirect will redirect the user to a sign in/sign up Auth0 page.

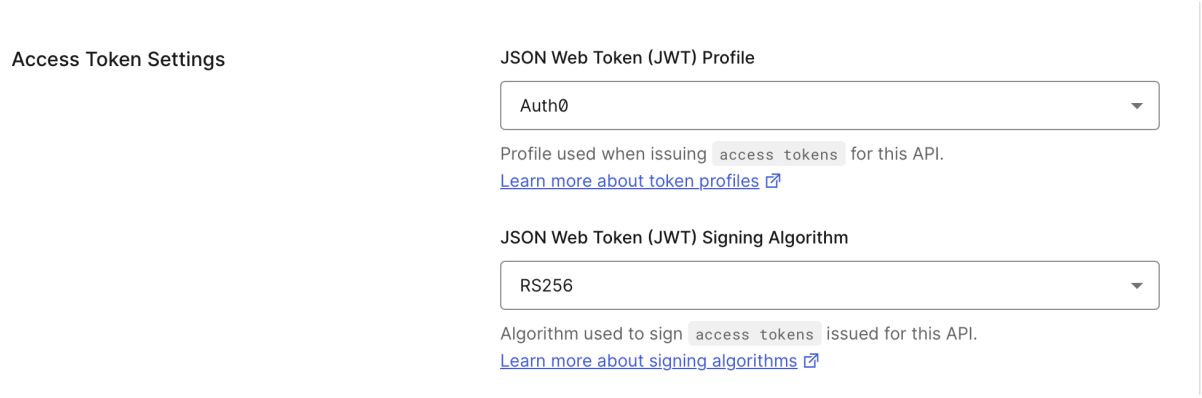
Express.js API Integration:

To make an authenticated user on our frontend able to perform requests to protected routes on the API we must set up an auth0 API server like we did with the frontend.

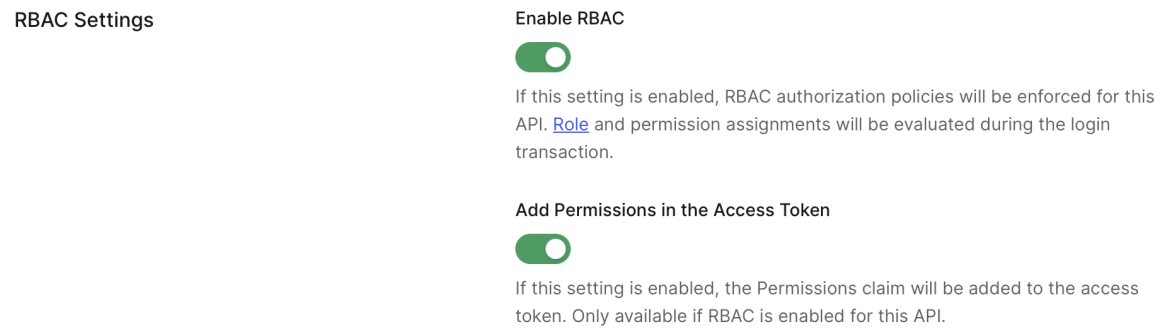
So head to the Auth0 dashboard → choose **Applications** from the left sidebar → from the sub menu choose **API's.** From there press the **Create API** button.

Fill out the form. The audience field is important in this context as that is the missing environment value in our .env in the React project. Once the API application has been created copy paste the audience value to **REACT_APP_AUTH0_AUDIENCE** environment value. This will be used to fetch the access token used for authorization on the API.

Additionally head to the Auth0 API application you just created and press the **Settings** tab. Scroll down to **Access Token Settings** and ensure they look like this:



Scroll further down to **RBAC Settings** and ensure they look this:



After that head to your Express.js code and insert this to your .env file:

```
PORT=6060
CLIENT_ORIGIN_URL=http://localhost:4040
AUTH0_AUDIENCE=AUTH0-AUDIENCE
AUTH0_DOMAIN=AUTH0-DOMAIN
```

AUTH0_DOMAIN and **AUTH0_AUDIENCE** must match the values defined in the React .env file. CIENT_ORIGIN_URL is the origin of your React application.

After this copy the contents of this package.json file to your package.json file:

```
{
  "name": "api_express_javascript_hello-world",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node src/index.js",
    "dev": "nodemon src/index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^10.0.0",
    "express": "^4.17.1",
    "express-oauth2-jwt-bearer": "^1.1.0",
    "helmet": "^4.6.0",
    "nocache": "^3.0.1"
  },
  "devDependencies": {
    "nodemon": "^2.0.15",
    "prettier": "^2.4.1"
  }
}
```

Make sure to use npm install after to install the packages

Create a file called middleware.json and copy paste this code:

```
const { auth } = require("express-oauth2-jwt-bearer");
const dotenv = require("dotenv");

dotenv.config();

const validateAccessToken = auth({
    audience: 'http://localhost:3001',
    issuerBaseURL: 'https://dev-kyyqmmxci28c240u.us.auth0.com/',
    tokenSigningAlg: 'RS256'
  });

module.exports = {
  validateAccessToken,
};
```

After this head to the file that contains your express routers and import the middleware:

```
const { validateAccessToken } = require("../middleware/auth0.middleware.js");
```

Now attach the middleware to your protected routes like this:

```
router.get("/protected", validateAccessToken, (req, res) => {
  const message = getProtectedMessage();

  res.status(200).json(message);
});
```

Now we have setup authorization on your backend using Auth0 provider. To pass the access token to the protected endpoints from React do the following:

Fetch the getAccessTokenSilently function from the useAuth0 hook:

```
const { getAccessTokenSilently } = useAuth0();
```

Retrieve this environment variable from the .env file:

```
const audience = process.env.REACT_APP_AUTH0_AUDIENCE;
```

And then use it to fetch the access token from Auth0 and pass it to the authorization header on your request:

```
// Get access token from Auth0
const accessToken = await getAccessTokenSilently({
  authorizationParams: {
    audience: audience,
    scope: "read:messages",
  },
});

// Call your backend with token in Authorization header
const response = await fetch("http://localhost:3001/api/messages/create-payment", {
  method: "GET",
  headers: {
    Authorization: `Bearer ${accessToken}`,
    "content-type": "application/json",
  },
});
```