Assignment 1 Reflection

Aiden Burgess - 600280511 - abur970

How must the marker setup their Auth0 Application and API to work with your solution?

Create a simple single page application with auth0, you will need the domain and client ID.

Then, under "Application URIs" in settings, add http://localhost:3000/ to "Allowed Callback URLs", "Allowed Logout URLs" and "Allowed Web Origins"

What files need to be changed in your solution (and where in those files), so that the marker can use their own Auth0 Application and API?

In `backend/config.js`, replace the issuer with your issuer address.

```
const auth0 = {
    audience: "http://localhost:3001/",
    // Replace issuer here
    issuer: "https://dev-exampleissuer.au.auth0.com/",
};
```

In `frontend/.env` replace the `REACT_APP_AUTH0_DOMAIN` and `REACT_APP_AUTH0_CLIENT_ID`, with your respective domain and client id.

```
# Replace with your domain
REACT_APP_AUTH0_DOMAIN=dev-exampleissuer.au.auth0.com
# Replace with your client id
REACT_APP_AUTH0_CLIENT_ID=XXXiX4tXjBxXX2XxqdcXsm5qtp4EyWTt
REACT_APP_AUDIENCE=http://localhost:3001/
```

What changes did you make to the frontend in order to implement Task One? (i.e. which files did you add / change, and why)?

In App.js, added login and logout buttons to the appbar to allow the user to login and logout. Also added state to determine if the user is authenticated. Based on this state, the main content of the page either shows their respective todolist, a login button, or a loading spinner.

Created a login and logout button components, which use auth0 to perform their respective actions.

Created `Auth0ProviderWithHistory` which was added as a wrapper for the main `App` component in `index.js`. This allowed the auth0 state to be reacted to by changing the history, i.e. browsing to other pages. For example when the user logs in via auth0 it redirects them to the homepage.

To allow for this browsing functionality, `App` was also wrapped in a `BrowserRouter` component in `index.js`

What changes did you make to the backend in order to implement Task Two?

Jwt was included in the making of the server in `server.js`. This allowed the app to have all routes authenticated.

The implementation of jwt is found in `jwt.js` which creates a jwt object from the `config.js`.

The Todo schema in `todos-schema.js` was updated with an extra field for the user of a todo item.

The functions to retrieve and interact with the database in `todos-dao.js` were changed to include user as a parameter, as we should only be interacting with todos for a specific user.

The most changes can be found in `todos-routes.js`. We extract the user from each request and use that in conjunction with our DAO to retrieve user-specific information. In cases where the user is trying to access another user's information, we instead return a `401: Unauthenticated` error.

Comment on how easy / difficult it was to add Auth0 authentication to the webapp, and on the documentation you read. What was easily achievable? What was more difficult than anticipated? How effective was the documentation you read in aiding you? What changes, if any, would you recommend, to the documentation for new developers trying to use it?

I followed a Youtube tutorial, and it was easy to setup Auth0 authentication for the application. The official documentation was also readable and helpful when trying to understand how Auth0 worked. The most difficult aspect was actually the backend testing, where I had to create tokens for test requests. The official documentation should have a more straightforward article about backend testing, and mocking as that would be a very common developer need.