1/08 Class1Group1 Assignment 3: TOTP:

Workflow:

1. Go to url
2. Put in credentials and click "login"
3. Run command line app and put generated, printed code into page, then "submit"
4. They should be taken to the new page

common.js

* Updated login() function to contain a line to redirect to newpage.html after successfully logging in (status code 200)
* Created a generateTOTP function which calls index.js to call the totp-generator.js file to create a TOTP on the user side.
* Created a generateTOTP2 function which calls index.js to call index.js’s helper file to create a TOTP on the server side (AKA displays to console directly)

index.js:

* Has access to the SECRET environment variable for TOTP generating functions to use: ALGAE
* Uses crypto npm dependency to create a 6 numerical character TOTP
* Updated to have two functions:

1. generateTOTP: Acting as the user’s TOTP: generates a TOTP calling the totp-generator.js
2. generateTOTP2: Acting as the sever’s/backend’s TOTP: uses a helper function defined within index.js to generate a TOTP simultaneously with the user’s TOTP.

* Updated to have verify() function, which is called once the button is updated and clicked in newpage.html

newpage.html

* Has a text area for a generated TOTP, and is initially disabled until the button changes to Verify TOTP
* Has a button that initially calls the generateTOTP functions simultaneously. After initial click, the button changes function to verify TOTP and the text box is enabled again.
* An alert displays if TOTP’s match exactly

totp-genrator.js:

* Added a SECRET environment variable for TOTP generating functions to use: ALGAE
* Uses crypto npm dependency to create a 6 numerical character TOTP

docker-compose.yml:

* Added a SECRET environment variable for TOTP generating functions to use: ALGAE

dockerfile:

* Modified to copy EVERYTHING within the container so that the totp-gnerator.js file is accounted for. Takes about 10 seconds to compose up the server. Not ideal, but functional.
* Modified the commands to include “node” and “index.js” so that index.js can execute files by itself.

12/20 Class1Group1 Assignment 2: Login

index.html:

* Updated to contain two text area elements and a button.
* Updated mode to: cors to allow crossing of data
* One text area is for the username the user wishes to log in with
* One text area is for the password the user wishes to log in with
* The button calls the common.js’s login() function

common.js

* Updated to contain a login() function, which makes a POST request to the login function in the index.js file.
* Updated mode to: cors to allow crossing of data
* Receives status code 200 if the login was successful, code 500 if the login was not successful do to server errors, or a code 400 error if either username or password that was input by the user on the index.html page was incorrect when comparing it to the single entry in the sql server.

index.js

* Contains a login() function
* Accesses PEPPER value stored in Docker environment file.
* Uses bcrypt() to create its own hash from the salt in the database, the text input for the password from index.html, and the PEPPER stored in the Docker environment file.
* Uses bcrypt to compare the hash it just created to the hash stored in the sql database.

users.sql

* Updated so that the salt used is stored in the “salt” column and the hashed password is stored in the “password” column

12/13 Class1Group1 Assignment 1: App structure

The app can be broken down into three major components:

1. The Docker Container
2. The SQL Database
3. A folder called “server” that contains two separate folders to
   1. Display the browser and its media contents (front end)
   2. Connect to the database hosted on the docker container (back end)

The SQL database simply contains user information with 3 columns of information: username, password, and the email associated with the user. The database’s contents loaded into a docker container upon initialization. With the added login page, it will now first require a username and password to login and then it will go back to the main page that will give information.

The frontend folder contains the tab icon png, the index.html file that creates the browser page, the common.css file to decorate the browser page, and the common.js file to apply functionality to the browser and its elements, which at this point are just the url and the button to query the database.

The backend folder contains a single index.js file that handles connecting the client and fetching the contents of the SQL database upon initialization.