**SchedgeToTheEdge**

I present you STTE(Schedge To The Edge) a react web app for managing your schedule in a different way, if until this day you used to struggle with time management say no more, STTE brings a unique approach for managing tasks on daily basis, this unique approach lets you choose a task and a task length and the rest is handled by our algorithm, but that’s not all STTE also lets you set meetings with other STTE users and sync your tasks so it will only take free spaces in time while keeping you healthy with full 8 hour sleep schedule, the app is fully responsive and very comfortable for new users.

**File structure:**

In this project we have 2 main folders:

1. final/node js: this folder contains the backend side of the project.
   1. The main folder contains the server file, which is the main file for the srever (we decided that some middleware will be written here too), the schemas and a .env file.
   2. Inside we have another 2 main folders, controllers and routes which handles the different functions needed for this app to work (including middleware).
2. maybe: this folder contains the frontend side of the project.
   1. Inside this folder we have a src folder with all the required frontend files.
   2. Inside the main folder we have our main files which are the login/signup screen, the main page, the ‘main’ file(AKA index but we preferred main), the different routes, and the main SASS file named “style”.

Also inside this folder we have 3 sub folders.

* + 1. api: this folder contains out axios instances (one uses cookies and one is basic).
    2. comps: contains a custom react component named app bar which is the app bar we see throughout the app.
    3. things for auth: contains all the files needed for a successful and secure authentication, which are:
       1. keepAuth – a file for setting the current user authentication as context which will let us use those variables at a couple of routes without needing to import it.
       2. useAuth – a custom hook that allows us to change (for set/reset) the authentication data.
       3. useAxiosWithJWT – a custom hook for adding and removing interceptors for our authenticated axios requests
       4. useRefreshToken – for extra security we have 2 access tokens, a regular one that expires after a short period of time and another one set as an HTTP only cookie (so it can’t be accessed by js) for refreshing it, this custom hook is used to get a new access token after first one expires.

**Tech used:**

This project is written in react for the frontend, node js for backend and mongodb as database sorce.

In the frontend we also use the following libraries:

1. Material UI (mui) – a library for setting components.

2. react-router-dom – a library for setting routes and navigation.

3. vite – a very powerful approach for creating and running react apps.

4. react-big-calendar – a library for setting calendar layouts (includes moment for date management).

5. axios – a very convenient way for handling http requests.

In the backend we use the following libraries:

1. express - a must have library for opening and managing node server (in my opinion).

2. dotenv – another really convenient library for managing .env files and variables.

3. cookie parser – a library used for handling cookies.

4. jws – one of the most popular for creating comparing and handling authentication.

5. mongoose – a very convenient way for fetching data from mongodb.

6. moment- a library for managing and changing dates.

Testing:

We have already set 2 users and some test tasks, the passwords are 123456789 for both and the usernames are tester and passIs123456789

The database is open for all ip for a week starting today(29/11/2022).

**That’s all from us.**

**This project was created by:**

**David Shelah:** **207933912.**

**&**

**Elia Deri: 323894311.**

**Enjoy 😊**