Project React Report

Tianyang CHEN

28/02/2021

- Github ID

TianYangCHEN0928

- Accomplished tasks

Backend:

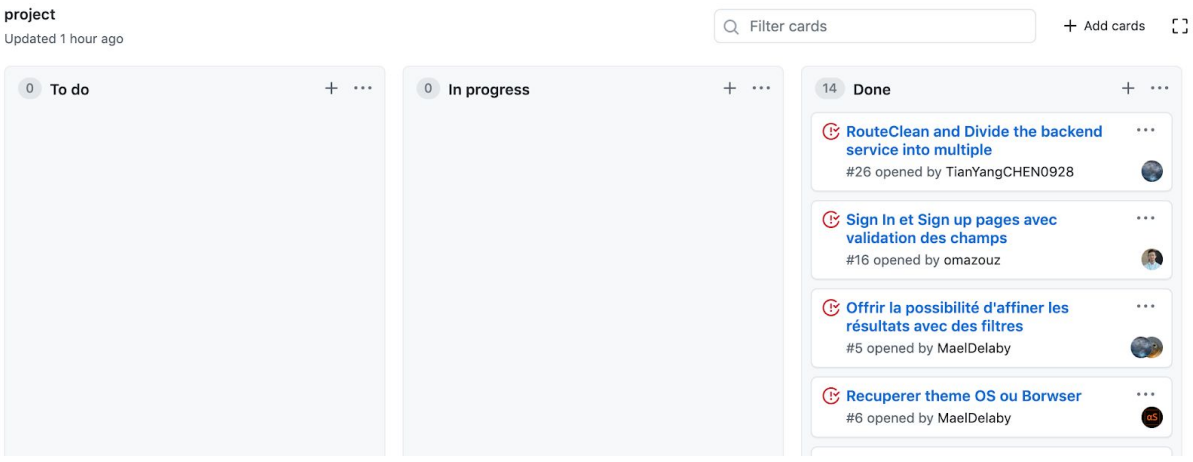
- SignIn Request
- Generate the JWT
- FilterData Request
- Create different Controllers
- Create model for the MongoDB
- Manage the route for different services

Frontend:

- SignUp Page
- SignIn Page
- Get result with filters

- Strategy used for version control with Git

We use the <Kanban> of Github for project schedule management. It is an important means of agile development. All the tasks that need to be completed are made into cards and pasted on a white board. This is our <Kanban>.



According to different stages, we divide the <Kanban> into three columns.

- ❏ Todo
- ❏ In progress
- ❏ Done

In addition, we have also created many branches to do each task. When the task is completed, he also needs to send a pull request to others, and only after others accept it, can it be merged into the dev branch. After all branches have been merged into dev, we will do the test for the project. And if the test passes, we will merge dev into master.

- Solutions chosen

For the DataBase, we choose the MongoDB Atlas which provides database services in the cloud. And it’s free for us because we don’t have a lot of data to store. And another advantage of MongoDB Atlas cloud database is that we only need to focus on the development, instead of the operation and maintenance of the database: deployment, disaster recovery, backup, and monitoring. In addition, I have used Mongodb Atlas in previous project development. So I don't need to spend time learning how to operate and connect it. And it can also set the white list of IP addresses,which improve the safety.There are also some other cloud databases, like Amazon DynamoDB,Apache CouchDB and so on, but we haven’t used them before. So we chose Mongodb Atlas which we are most familiar with. We can connect and operate it easily with “mongoose”.

For the Frontend, we choose the Material-UI which is one of the most popular React interface frameworks in the world which implements the Google Material Design design specification. And it is already known to several members of the project and corresponds to the different visualization constraints present in the project: presentation in the form of a table and a graph in particular.In addition, it helps us save time, so that we don't need to write a lot of css files, and at the same time we can get a very beautiful UI, There are some other frameworks that could have been used like Ant Design, React Bootstrap and so on.

- Difficulties encountered

The first problem I encountered is that after the user SignIn, the backend will return a token because we don't want him to submit the username and password every time.And how should I store and manage this token safely. Generally, it is fine to put it directly in localStorage, but localStorage is vulnerable to attacks.If security is considered, it should be placed in IndexedDB or stored after encryption.

The second problem is about getting the data by filtering. We don't want to get all data from the database, and we want to get the data by some filtering (like departement,jour,etc). And how could we send the filter data request in the frontend page by clicking the filter options. If we can't handle this problem, every time the frontend will get all the data from the database, and just do some filter in the frontend page. And it will take a lot of time to load so many datas in the page.

- Development time / task

Task	Time
SignIn Request	1h
Generate the JWT	2h
Create different Controllers	5h
Create model for the MongoDB	1h30
Manage the route for different services	2h
FilterData Request	2h
SignUp Page	4h
SignIn Page	3h
Get result with filters	4h

- Code

Comment on a function or component (max 100 lines) that you have written and that seems elegant or optimal to you

```
onSubmit: async (values) => {
        const { email, passWord } = values
        await axios
                .post('http://localhost:4001/auth/signin', {
                        email,
                        passWord,
                })
                .then((response) => {
                        if(response.data.success){
                        const { token } = response.data
                        localStorage.setItem('user', JSON.stringify({ token }))
                                        history.push(history.location.state?.from.pathname || '/')
                                }else{
                                        alert('incorrect indentifiants')
                                }
                                })
                                .catch((err) => {
                                        setOpen(true)
                                })
                },
        })
```

The code here, I have used hooks and localStorage. I will check when we have the response if the succes is true “if(response.data.success)”, then i will get the token from response and stock it on local storage. After that,I will push the last url or homepage.

Comment on a function or a component (max 100 lines) that you have written and which deserves an optimization or improvement

```
const signIn = async(req,res)=>{
    let userResult = await User.findOne({email:req.body.email}).exec();
    const accessTokenSecret = "webproject"
    if(userResult != undefined) {

        if (req.body.passWord == userResult.passWord) {
            const accessToken = jwt.sign({ email : userResult.email} , accessTokenSecret);

            return {
                status: 200,
                success:true,
                res_msg: "Login Successfully!",
                token: accessToken
            };
        } else{
            return {
                status:403,
                res_msg: "Incorrect account or password!",
            };
        }
    }else{
        return {
            status:404,
            res_msg:"Account does not exist!"
        };
    }
}
```

The JWT sign here is so simple, and it should be improved in the future. We may sign with the RS256 asymmetric algorithm which uses a public/private key pair. And we could use the private key to generate the JWT, and use the public key to verify the JWT. It will improve the security of the token.