Rapport individuel pour programmable web

Nom: Mazouz
Prénom: Othmane

Développements:

- Identifiant git: **omazouz**
- Tâches effectuées:

  - Front end: Sign up/ Sign In / Routing

  -Back end : Sign Up

- Stratégie employée pour la gestion des versions avec Git:
  - on a divisé le projet en issues et chaque personne choisit l'issue auquel il veut travailler.
  - Pour chaque issue on doit lui créer sa propre branche pour éviter les conflits quand on push dans la branche "dev" .
- Solutions choisies:

  - Pour le Sign Up j'ai utilisé "useFormik" et "Yup"pour bien contrôler les champs ainsi la validation et aussi "useFormik" permet de gerer les champs au lieu d'utiliser "useState" .

  - Pour Sign In j'ai utilisé aussi "useFormik" pour gérer les champs et localStorage pour stocker le token afin de l'utiliser après pour les accès aux pages .

  - Pour les routes j'ai utilisé la librairie "react-router-dom" pour me permettre d'utiliser ses hooks ainsi que pour gérer l'accès aux pages avec "ProtectedRoute" en se basant sur le token.

- ## Difficultés rencontrées:

    - Au début j'avais rencontré des difficultés par rapport au routing vu qu'on doit rendre les pages dans "Appdrawer", mais j'ai opté pour deux router : Le premier dans le main de "Appdrawer" et le deuxieme c'est en general qu'est rendu dans "App.js" et qui gère l'accès aussi.
    - les middlewares en backend avec le signup, mais à l'aide de Tianyang on a réussi à le finir.

- ## Temps de développement / tâche:

    - Routing frontend : 3h
    - Sign Up frontend : 4h
    - Sign In front : 3h
    - Sign Up backend: 8h

- ## Code :

    - Commenter une fonction ou un composant (max 100 lignes) que vous avez écrit et qui vous semble élégant ou optimal

        ### function component : pour protected routes

```
const ProtectedRoute = ({ component: Component, ...rest }) => {
    //stocker dans user le token sinon un objet nul
  const user = JSON.parse(localStorage.getItem('user') || '{}')
  const location = useLocation()
    //une fonction pour vérifier si une personne est connectée
  const isAuthenticated = () => {
      if (!user || !user.token) {
          return false
      }

      try {
          const { exp } = decode(user.token)
          if (exp < new Date().getTime() / 1000) {
```

```
                return false
            }
        } catch {
            return false
        }

        return true
    }
       // si la personne n'est pas connectee elle sera directement
envoyé vers la page de sign in
    if (!isAuthenticated()) {
        return (
            <Redirect
                to={{
                    pathname: '/signin',
                    state: {
                        from: location.pathname,
                    },
                }}
            />
        )
    }
    return (
        <Route
            {...rest}
            render={(props) => {
                return <Component {...rest} {...props} />
            }}
        />
    )
}
```

- Commenter une fonction ou un composant (max 100 lignes) que vous avez écrit et qui mériterait une optimisation ou amélioration:

submitting fonctionne dans sign up:

```
onSubmit: (values, { resetForm, setSubmitting }) => {
            console.log('submitted')
            const {
                firstName,
                lastName,
                email,
                passWord,
                birthDay,
                gender,

            } = values
            axios
                .post('http://localhost:4001/auth/signup', {
                    firstName,
                    lastName,
                    email,
                    passWord,
                    birthDay,
                    gender,

                })
                .then(() => {
                    console.log('Post successful!')
                })
                .catch((err) => {
                    console.log(err)
                })
            resetForm()
            setSubmitting(false)
        },
    })
```

je pense que dans cette fonctionne de submit je peux l'améliorer plus et ajouter d’autres fonctionnalités et aussi pour le projet ca aurait ete mieux si on utilise un contextApi pour gérer les données et l'état du projet