

Name: Hun Ravit

ID: e20180328

Ex1: client

Home:

```
    },
    methods: {
      logout() {
        axios.get('http://localhost:3001/api/logout')
          .then(response => {
            this.info = response
          })
          .then(() => {
            if(this.info){
              console.log(this.info.data.msg);
              if(this.info.data.msg=='logout success'){
                VueCookies.remove('Token');
                location.href = '/';
              }else{
                alert("Log out false");
              }
            }
          })
      },
      mounted () {
        var Token = VueCookies.get('Token');
        if(!Token){
          location.href = '/';
        }
      }
    }
  },
}
```

Login:


```
    methods: {
      getData(){
        axios.post('http://localhost:3001/api/login?username=${this.username}&password=${this.password}')
          .then(response => {
            this.info = response
          })
          .then(() => {
            if(this.info){
              if(this.info.data.msg=='success'){
                this.message_success = true;
                VueCookies.set('Token', this.info.data.token, "1h");
                location.href = '/home';
              }else{
                this.message_false=true;
                this.command = this.info.data.command;
              }
            }
          })
      },
    },
  },
}
```

Register:

```
methods:{
  writeUser(){
    axios.post(`http://localhost:3001/api/register?username=${this.username}&password=${this.password}&email=${this.email}&firstname=${this.firstname}&lastname=${this.lastname}`)
      .then(response => {
        this.info = response
      })
      .then(()=>{
        if(this.info){
          console.log(this.info.data.msg);
          if(this.info.data.msg=='success'){
            this.message_success = true;
            location.href = '/';
          }else{
            this.message=true;
            this.command = this.info.data.command;
          }
        }
      })
    },
  },
}
```

Result:

Already login.



Username or Email

Password

You cannot sign in again. Please sign out before sign in again.

Log in

Log in

Server:

```
const createUser = async (req, res) => {
  const username = req.query.username;
  try {
    const userExist = await User.findOne({ username: username })
    if(userExist){
      res.status(400).json({msg:'success',command:'Username is already exists'});
    }else{
      //encrypt password
      const hashPass = await bcrypt.hash(req.query.password, 12);

      const user = new User({
        username: req.query.username,
        password: hashPass,
        email:req.query.email,
        firstname:req.query.firstname,
        lastname:req.query.lastname
      });
      console.log(user);
      try {
        const newUser = await user.save();
        res.status(201).json({msg:'success', newUser });
      } catch (err) {
        res.status(500).json({msg:'error', message: err.message });
      }
    }
  } catch (err) {
    res.status(500).json({msg:'error', message: err.message });
  }
}
```

```
const login = async (req,res)=>{
  const { username,password } = req.query;
  // console.log(username,password);
  var nameFind = username;

  try {
    const user = await User.findOne({ username: nameFind})
    if(user){
      var info = user;
      //bcrypt
      const checkPass = await bcrypt.compare(password, info.password);
      if(checkPass){
        const token = await jwt.sign({username},process.env.SECRETE,{ expiresIn: '2h' });
        store.set(process.env.SECRETE,token);

        res.status(200).json({msg:'success','token':token});
      }else{
        res.status(400).json({msg:'Error',command:'wrong password'});
      }
    }else{
      res.status(400).json({msg:'Error',command:'User not exists.'});
    }
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
}
```

middleware

```
const checkLogout = (req,res,next)=>{
  if(req.user){
    // console.log(req.user);
    return next();
  }else{
    return res.json({msg:'error',command:"Attem to sign out fial."});
  }
}

const checkLogin = (req,res,next)=>{
  if(!req.user){
    return next();
  }else{
    return res.json({msg:'error',command:"You cannot sign in again. Please sign out before sign in again."});
  }
}

const checkRegister = (req,res,next)=>{
  if(!req.user){
    return next();
  }else{
    return res.json({msg:'error',command:"You cannot register a user."});
  }
}

const checkHomepage= (req,res,next)=>{
  if(req.user){
    return next();
  }else{
    return res.json({msg:"error",command:"You cannot go to home page. PLEASE log in."});
  }
}
```

Ex2: We add more route

Edit User

```
92 const updateUser = async (req,res)=>{
93   //get from token
94   var Whereusername = req.user.username;
95   const { username } = req.body;
96
97   const user = await User.findOne({ username: username});
98   if(user){
99     res.status(200).json({msg:'error',command:'Username is already exists'});
100   }else{
101
102     try {
103       const where = {username:Whereusername};
104       const setValue = {username:username};
105       const user = await User.updateOne(where,setValue);
106       if(user){
107         store.clear();
108         res.status(200).json({msg:'success',command:'already update please login again with your new username.'});
109       }
110     } catch (err) {
111       res.status(500).json({ message: err.message });
112     }
113   }
114 }
115
116 }
```

Edit Password:

```
const updatePassword = async (req,res)=>{

  var Whereusername = req.user.username;

  const hashPass = await bcrypt.hash(req.body.password, 12);
  try {
    const where = {username:Whereusername};
    const setValue = {password:hashPass};
    const user = await User.updateOne(where,setValue);
    if(user){
      store.clear();
      res.status(200).json({msg:'success',command:'already update please login again with your new password.'});
    }
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
}
```

Delete User:

```
const deleteUser = async (req,res)=>{
  var username = req.user.username;

  try {
    const where = {username:username};
    const user = await User.deleteOne(where);
    if(user){
      store.clear();
      res.status(200).json({msg:'success',command:'Your account is deleted.'});
    }else{
      res.status(500).json({msg:'success',command:'Cennot delete'});
    }
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
}
```