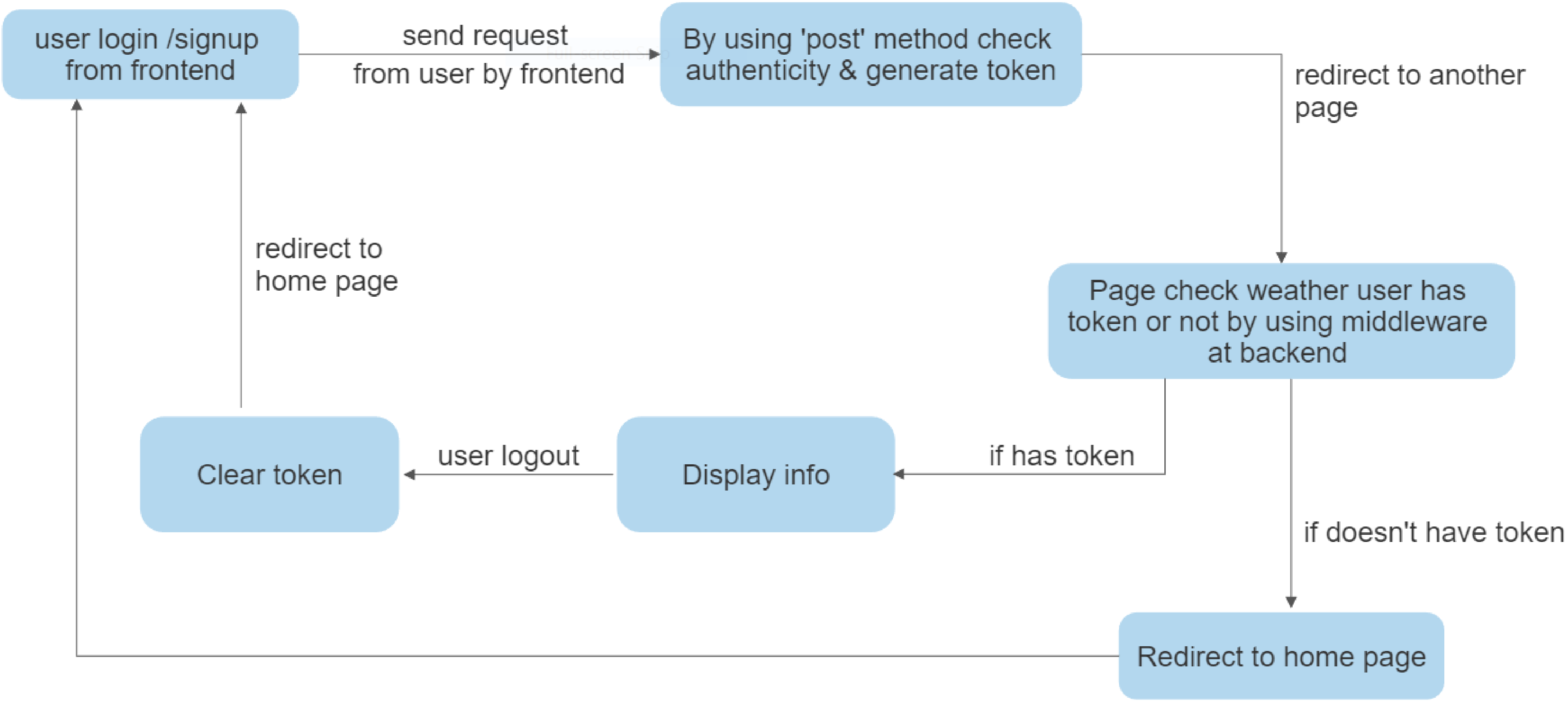
# Authentication using Jason web token



user login /signup from frontend

send request from user by frontend

By using 'post' method check authenticity & generate token

redirect to another page

Page check weather user has token or not by using middleware at backend

if has token

Display info

user logout

Clear token

redirect to home page

if doesn't have token

Redirect to home page

First of all we need to create react app

```
npx create-react-app "App_name“
```

We need to install some kind of packages by using these commands :

```
npm i react-router-dom
npm i react-alert
```

After that we need to add one line of code in package.json file of app above "dependencies" like this:

```
"proxy": "http://localhost:3000/",
```

After that we can run our app by using this command :

```
npm run start
```

Now For backend part we need to create one folder inside our appInside that folder we need to install express, mongoose by these commands :

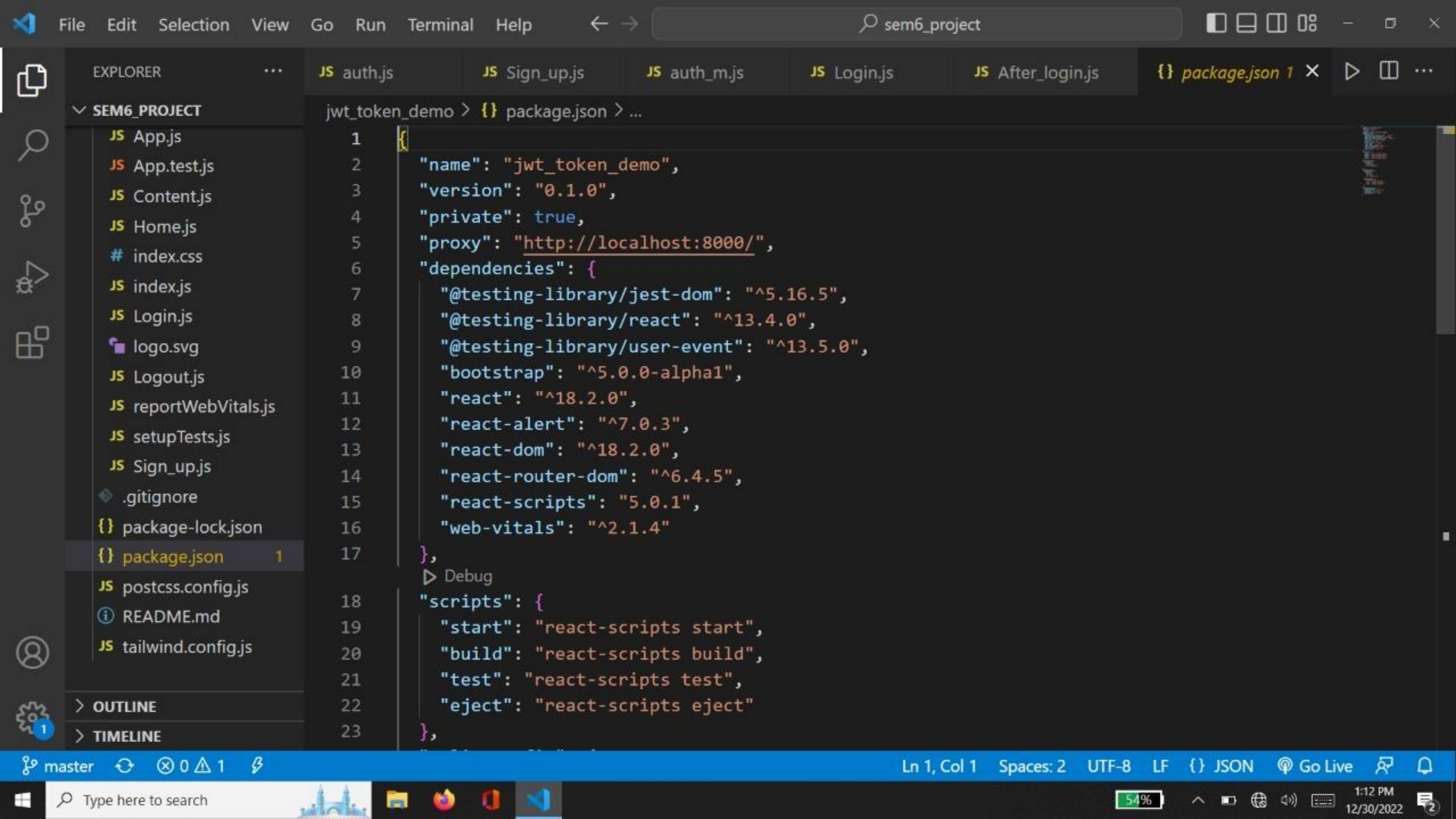
```
npm install express
npm install mongoose
```

command will generate package.json file

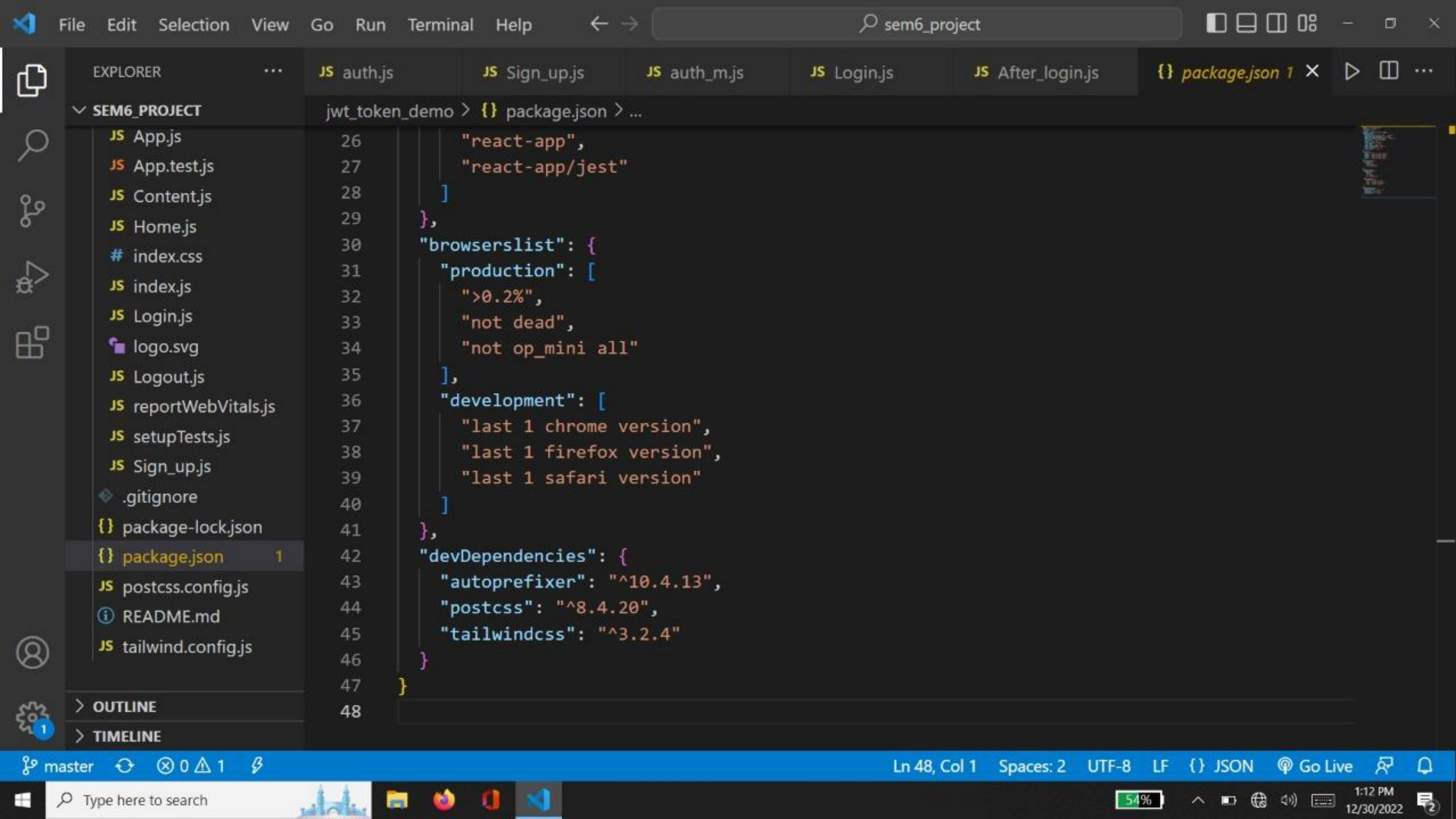
```
npm init –y
```

After that we need to install some packages by using these commands:

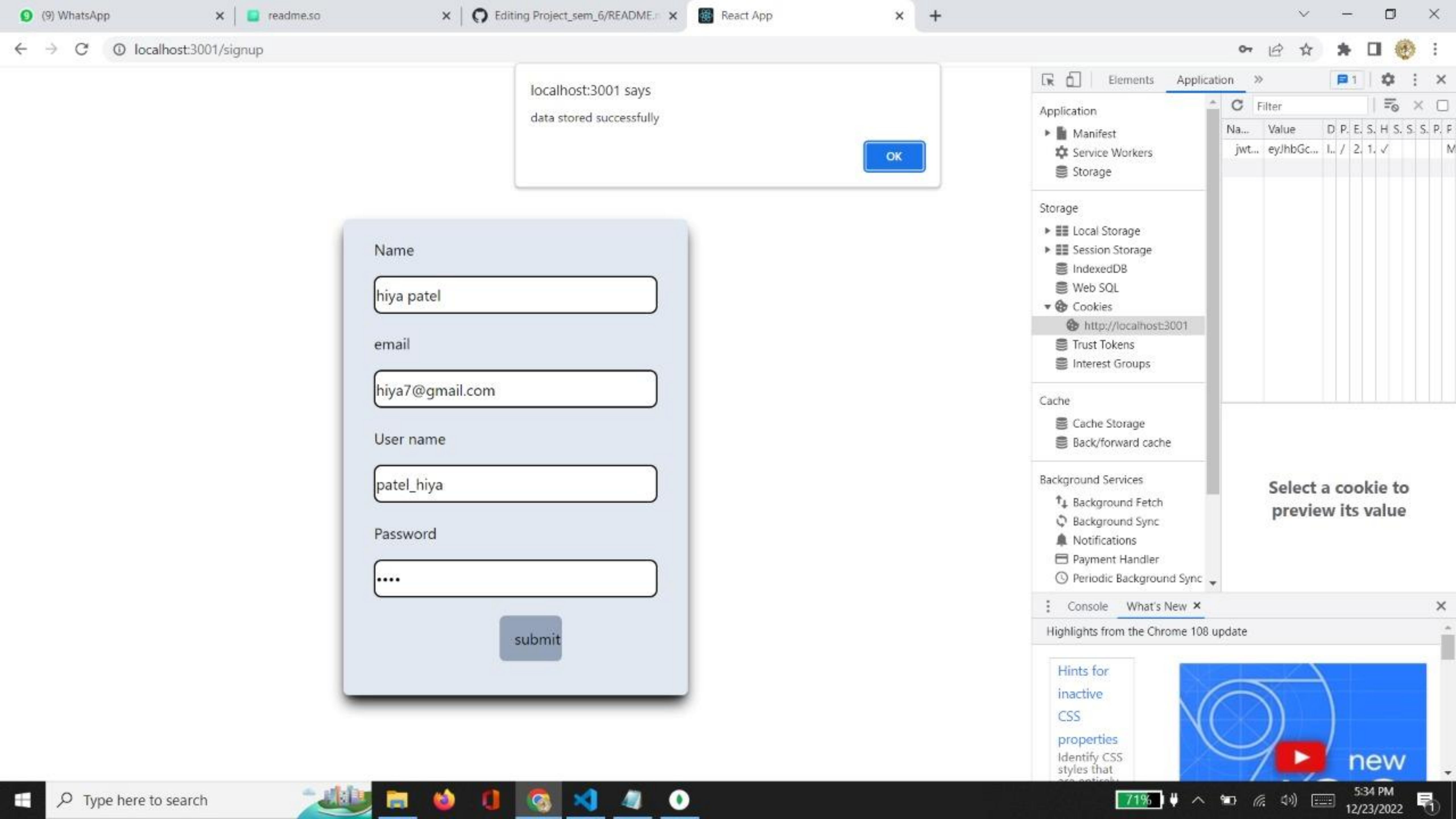
```
npm install bcryptjs
npm install cookie-parser
npm install dotenv
npm install jsonwebtoken
```



```json
{
  "name": "jwt_token_demo",
  "version": "0.1.0",
  "private": true,
  "proxy": "http://localhost:8000/",
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "bootstrap": "^5.0.0-alpha1",
    "react": "^18.2.0",
    "react-alert": "^7.0.3",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.4.5",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
```



```
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "devDependencies": {
    "autoprefixer": "^10.4.13",
    "postcss": "^8.4.20",
    "tailwindcss": "^3.2.4"
  }
}

```



localhost:3001 says

data stored successfully

Name

hiya patel

email

hiya7@gmail.com

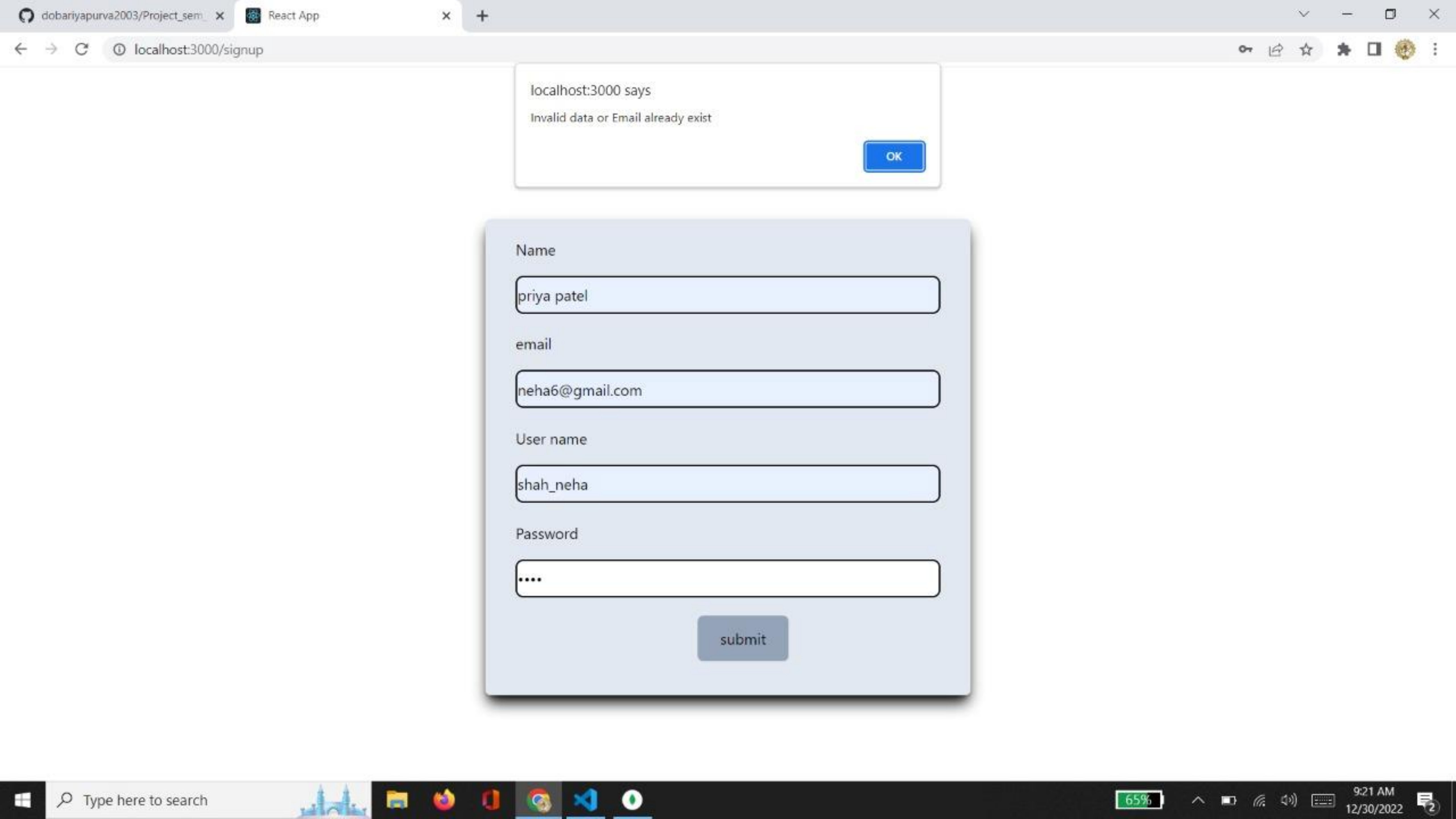
User name

patel_hiya

Password

••••

| Na... | Value | D | P. | E. | S. | H | S. | S. | S. | P. | P |
|---|---|---|---|---|---|---|---|---|---|---|---|
| jwt... | eyJhbGc... | l.. | / | 2. | 1. | ✓ | | | | | M |



localhost:3000 says

Invalid data or Email already exist

Name

priya patel

email

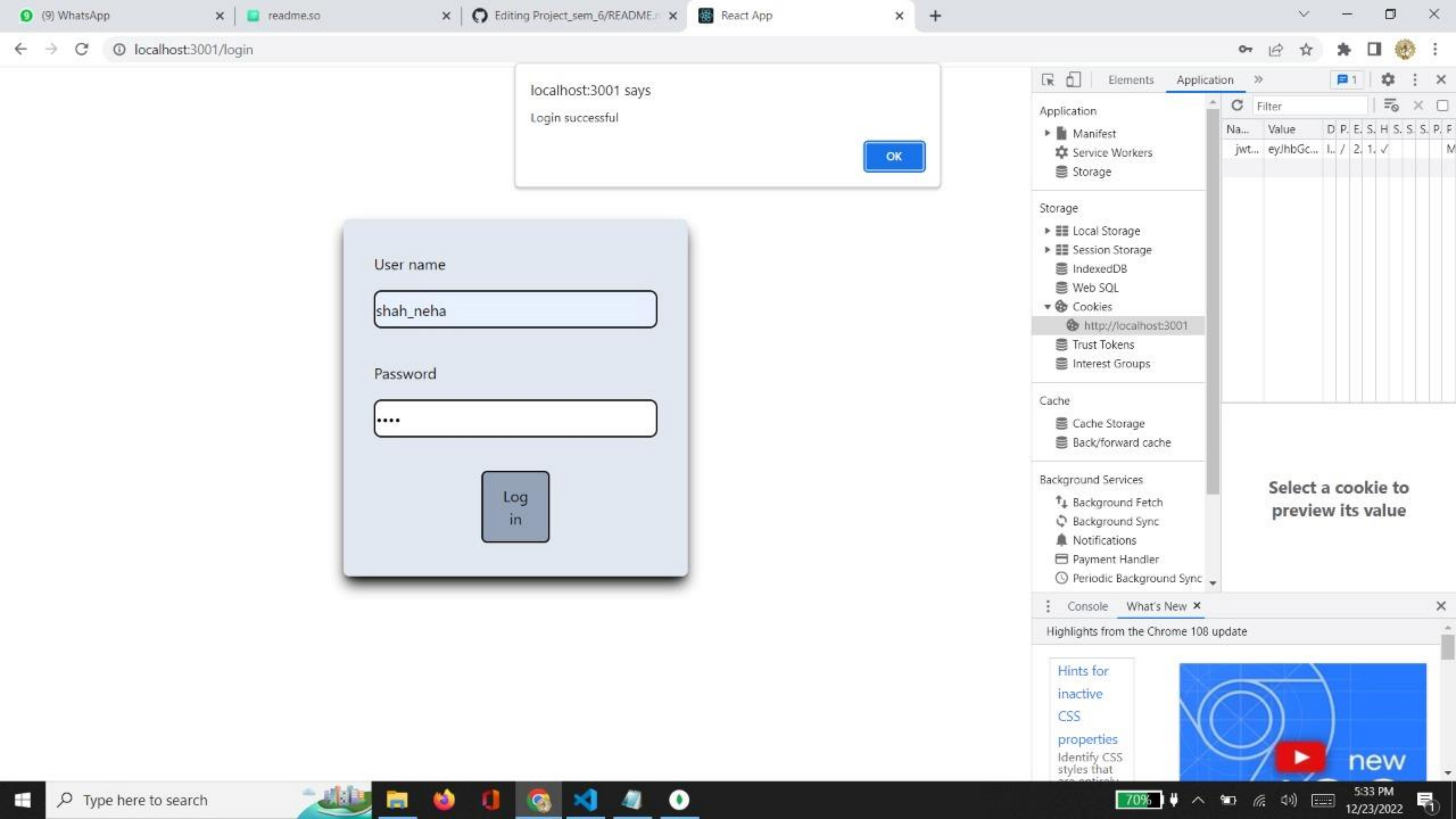
neha6@gmail.com

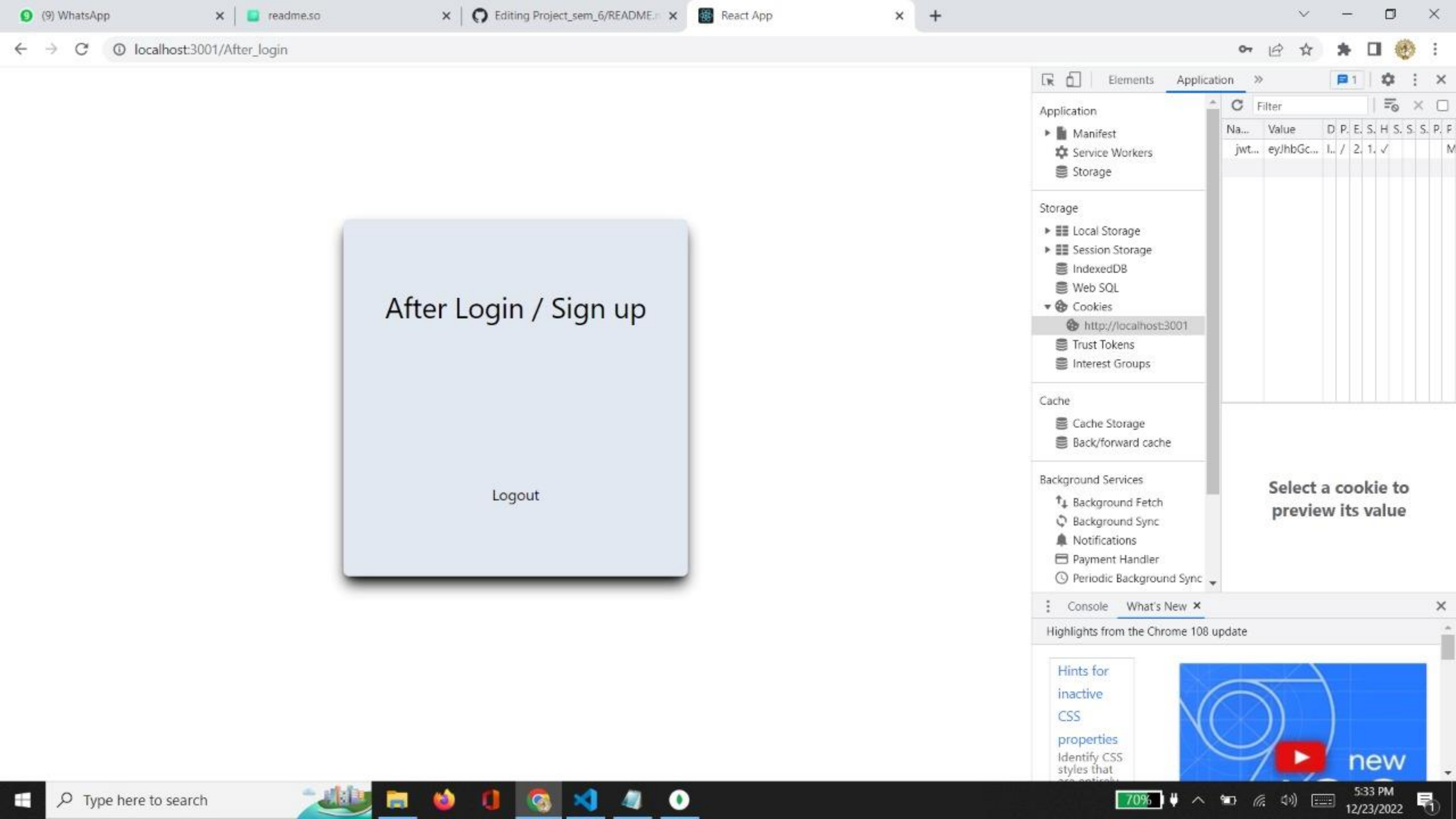
User name

shah_neha

Password

••••





After Login / Sign up

Logout

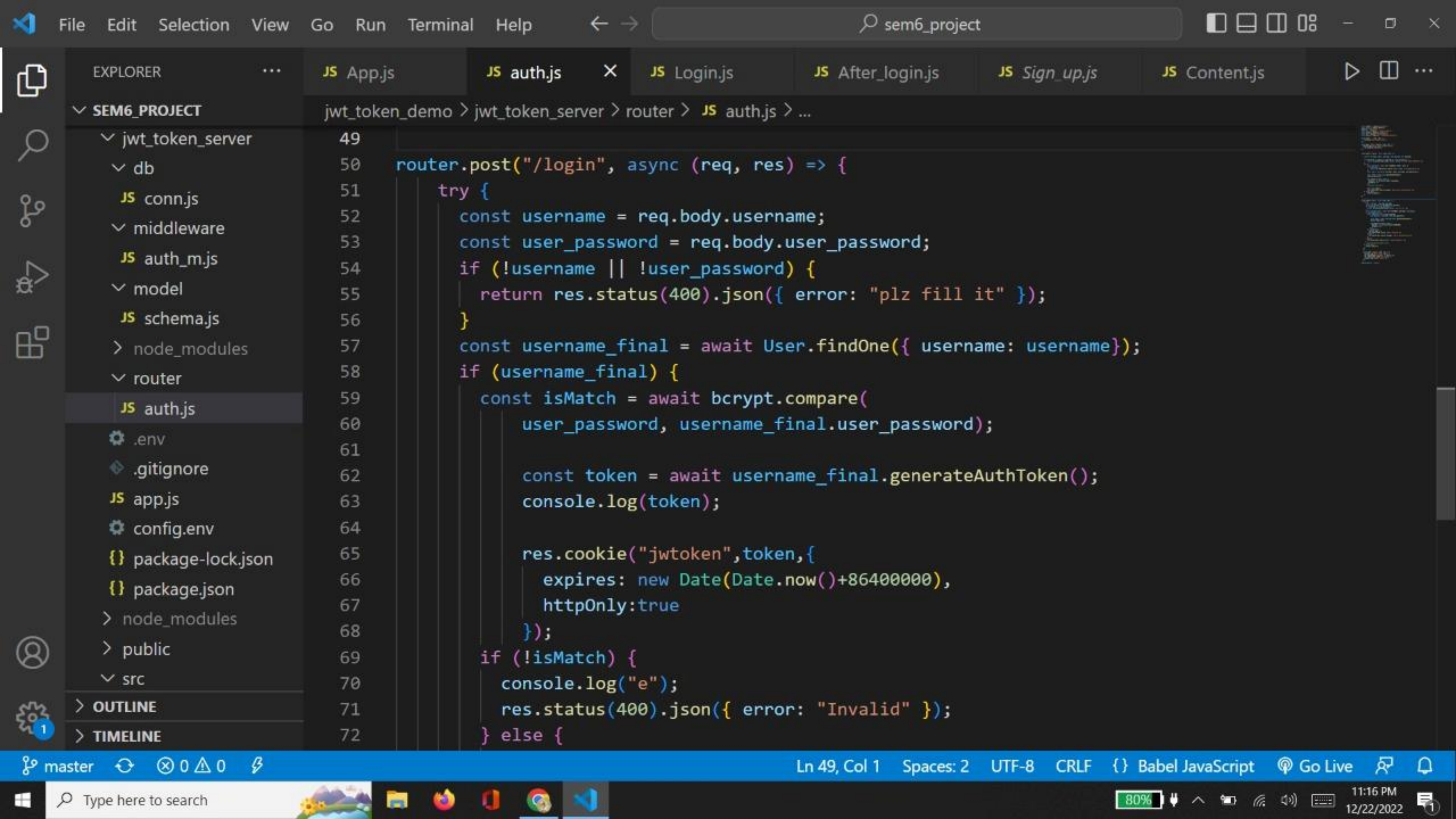
| Na... | Value | D | P. | E. | S. | H | S. | S. | S. | P. | P |
|---|---|---|---|---|---|---|---|---|---|---|---|
| jwt... | eyJhbGc... | l.. | / | 2. | 1. | ✓ |  |  |  |  | M |

Select a cookie to preview its value

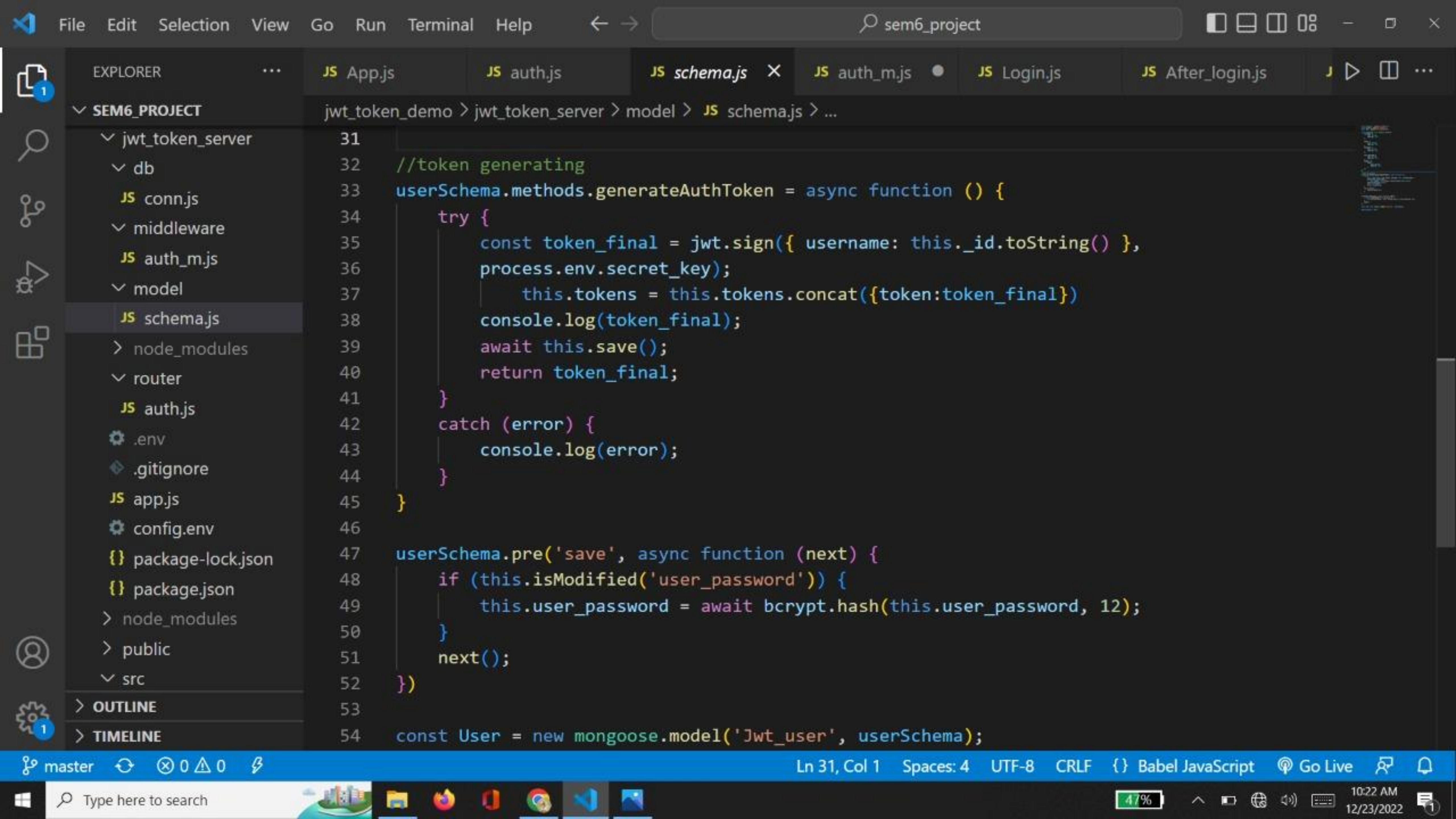
- At backend part by using ‘post’ method we authenticate weather user has correct username and password or not.
- If user has correct username and password we generate a token.
- That token is add in cookie by using inbuilt function
- Cookie(“name of token”,token,{

  expires(time),httponly:true

  })

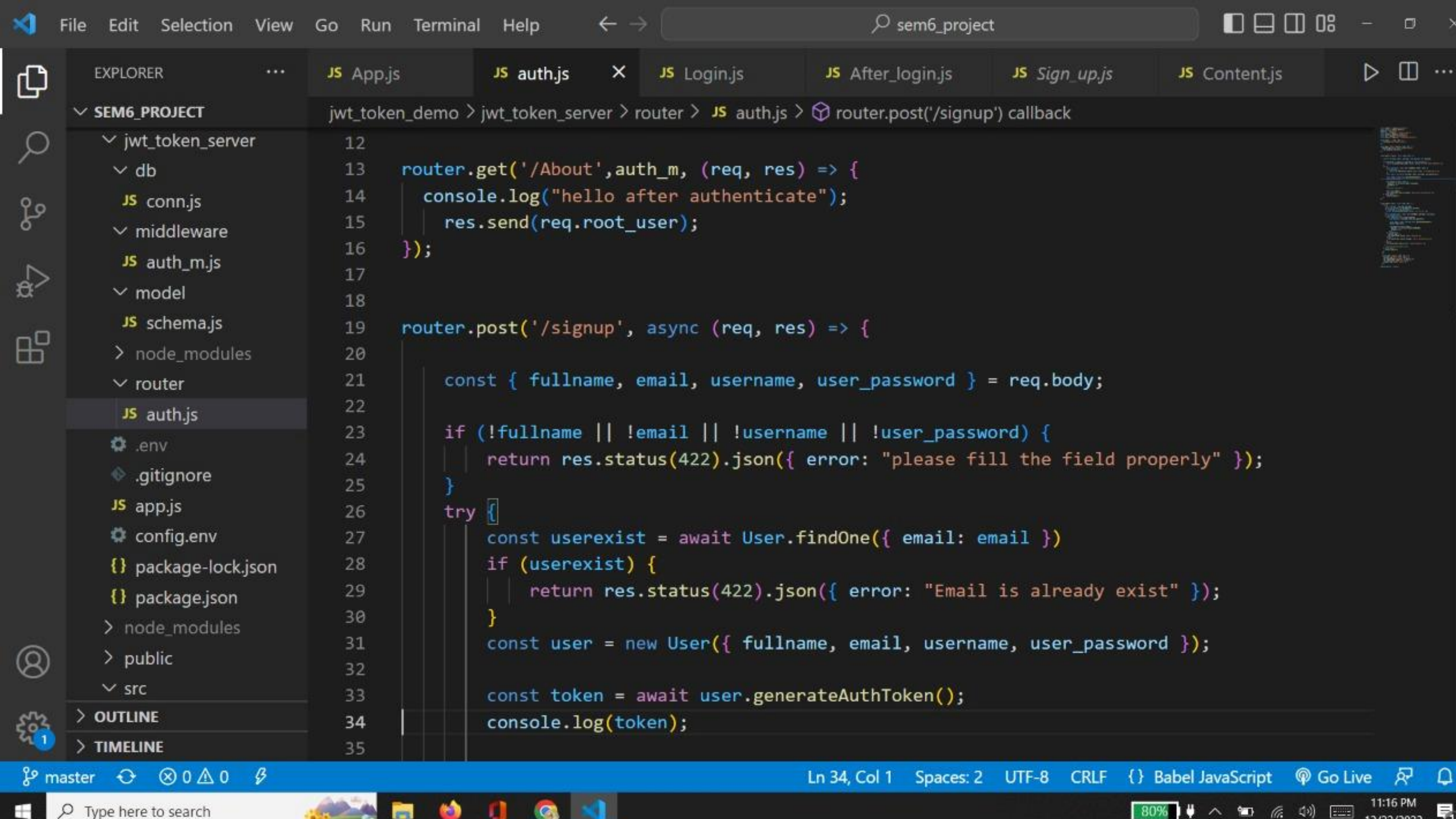


```
router.post("/login", async (req, res) => {
    try {
      const username = req.body.username;
      const user_password = req.body.user_password;
      if (!username || !user_password) {
        return res.status(400).json({ error: "plz fill it" });
      }
      const username_final = await User.findOne({ username: username});
      if (username_final) {
        const isMatch = await bcrypt.compare(
            user_password, username_final.user_password);

            const token = await username_final.generateAuthToken();
            console.log(token);

            res.cookie("jwtoken",token,{
              expires: new Date(Date.now()+86400000),
              httpOnly:true
            });
        if (!isMatch) {
          console.log("e");
          res.status(400).json({ error: "Invalid" });
        } else {
```



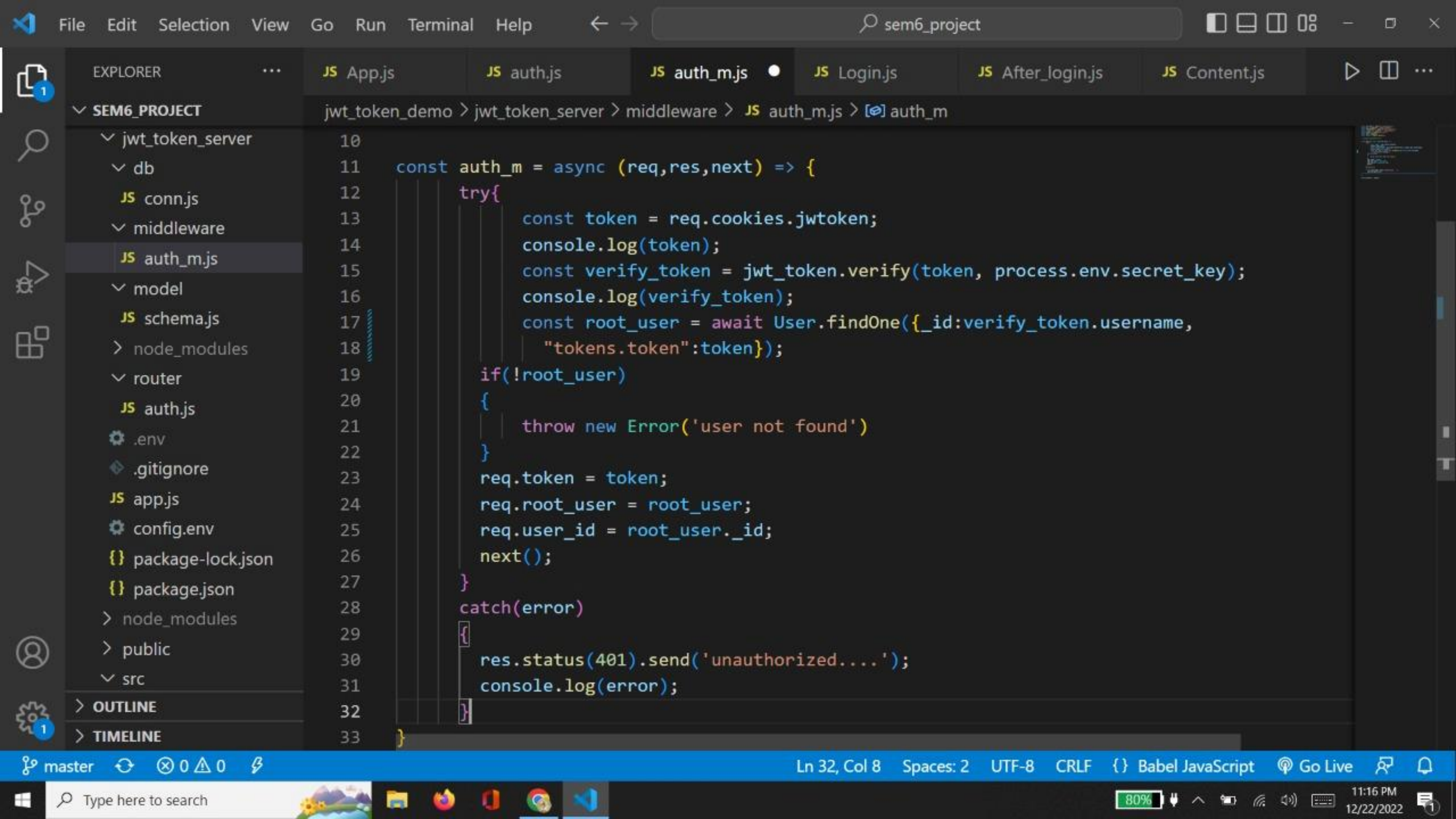
```javascript

//token generating
userSchema.methods.generateAuthToken = async function () {
    try {
        const token_final = jwt.sign({ username: this._id.toString() },
        process.env.secret_key);
            this.tokens = this.tokens.concat({token:token_final})
        console.log(token_final);
        await this.save();
        return token_final;
    }
    catch (error) {
        console.log(error);
    }
}

userSchema.pre('save', async function (next) {
    if (this.isModified('user_password')) {
        this.user_password = await bcrypt.hash(this.user_password, 12);
    }
    next();
})

const User = new mongoose.model('Jwt_user', userSchema);
```

- As we use post method for login in same manner we used for ‘sign up’.
- Here we added one constraint that if user has already signed up once with particular email id that email id can’t be used to again sign up.
- Here also we generated a token & stored it in ‘cookie’.



```

router.get('/About',auth_m, (req, res) => {
 console.log("hello after authenticate");
  res.send(req.root_user);
});


router.post('/signup', async (req, res) => {

  const { fullname, email, username, user_password } = req.body;

  if (!fullname || !email || !username || !user_password) {
    return res.status(422).json({ error: "please fill the field properly" });
  }
  try {
    const userexist = await User.findOne({ email: email })
    if (userexist) {
      return res.status(422).json({ error: "Email is already exist" });
    }
    const user = new User({ fullname, email, username, user_password });

    const token = await user.generateAuthToken();
    console.log(token);

```

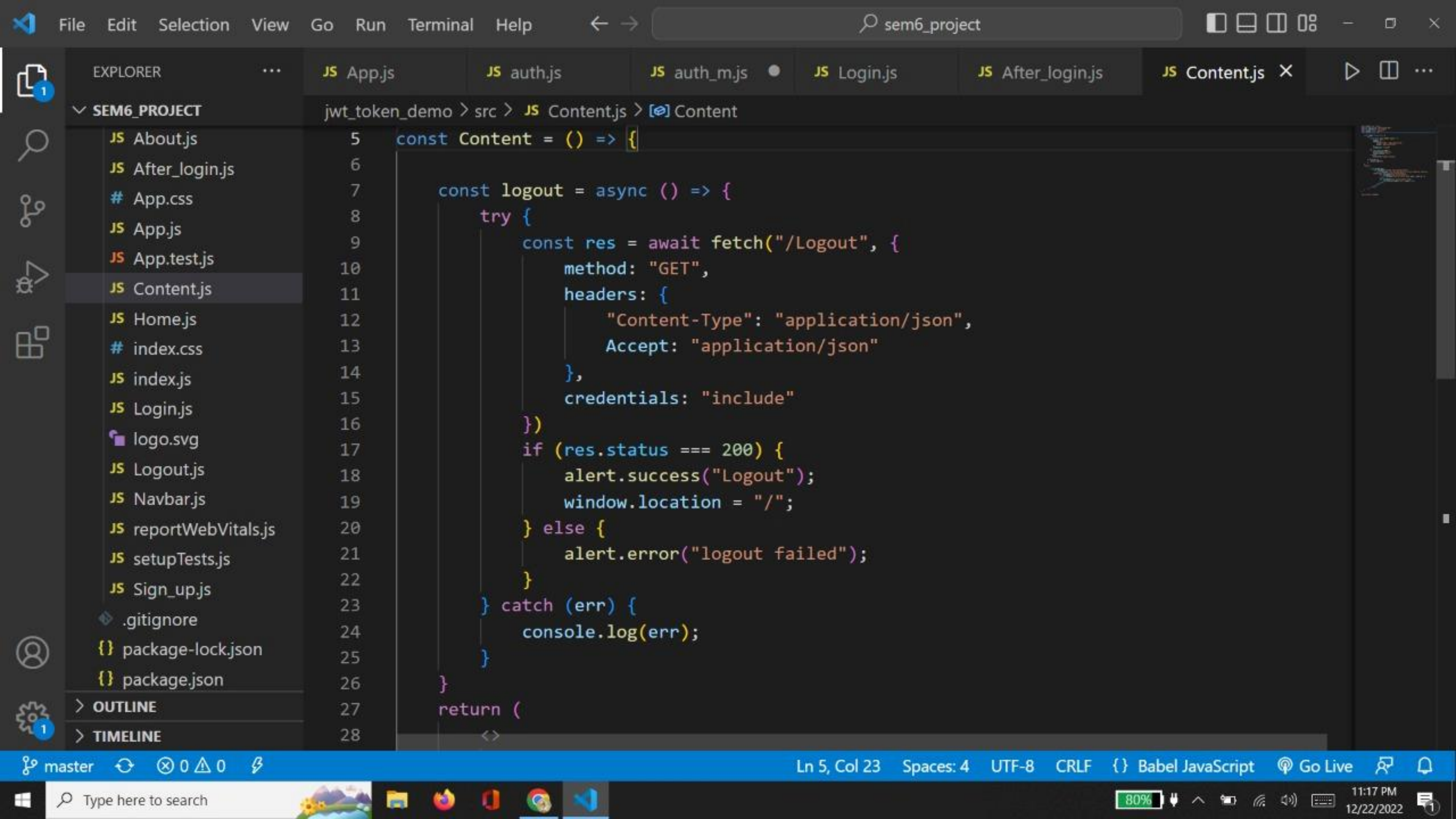
- Now for verifying weather user is having token or not we added middleware by using ‘get’ method.
- That middleware function verifies weather user has token or not by getting data from database where the token is stored.



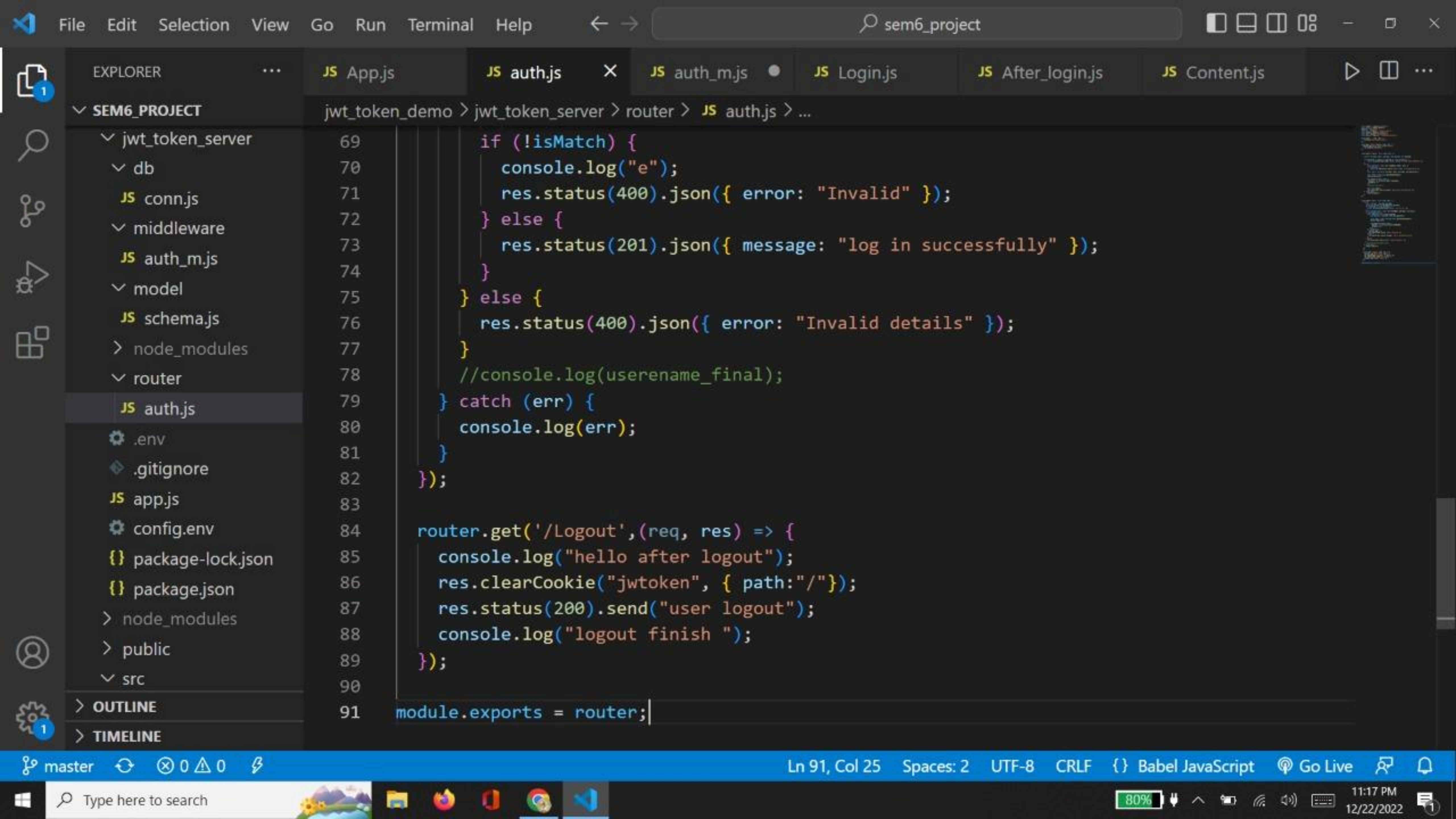
```
const auth_m = async (req,res,next) => {
    try{
            const token = req.cookies.jwtoken;
            console.log(token);
            const verify_token = jwt_token.verify(token, process.env.secret_key);
            console.log(verify_token);
            const root_user = await User.findOne({_id:verify_token.username,
             "tokens.token":token});
        if(!root_user)
        {
            throw new Error('user not found')
        }
        req.token = token;
        req.root_user = root_user;
        req.user_id = root_user._id;
        next();
    }
    catch(error)
    {
        res.status(401).send('unauthorized....');
        console.log(error);
    }
}
```

- Here is the page which should display after authentication.
- By this page user can logout.
- We send request for logout by ‘get’ method at backend.
- At backened if user has logout ,system will clear the token which was stored in cookie by using inbuilt function

```
            clearCookie(“tokenname ”,{path:”/”});
```



```
const Content = () => {

    const logout = async () => {
        try {
            const res = await fetch("/Logout", {
                method: "GET",
                headers: {
                    "Content-Type": "application/json",
                    Accept: "application/json"
                },
                credentials: "include"
            })
            if (res.status === 200) {
                alert.success("Logout");
                window.location = "/";
            } else {
                alert.error("logout failed");
            }
        } catch (err) {
            console.log(err);
        }
    }
    return (
        <>
```



```
    if (!isMatch) {
     console.log("e");
     res.status(400).json({ error: "Invalid" });
    } else {
     res.status(201).json({ message: "log in successfully" });
    }
   } else {
    res.status(400).json({ error: "Invalid details" });
   }
   //console.log(userename_final);
  } catch (err) {
   console.log(err);
  }
 });

 router.get('/Logout',(req, res) => {
  console.log("hello after logout");
  res.clearCookie("jwtoken", { path:"/"});
  res.status(200).send("user logout");
  console.log("logout finish ");
 });

module.exports = router;
```

- You can use this link to access playlist or usefull videos for mern development :
- https://youtube.com/playlist?list=PLwGdqUZWnOp3t3qT7pvAznwUDzKbhEcCc