DHRUVA PRASAD UPADHYAYA

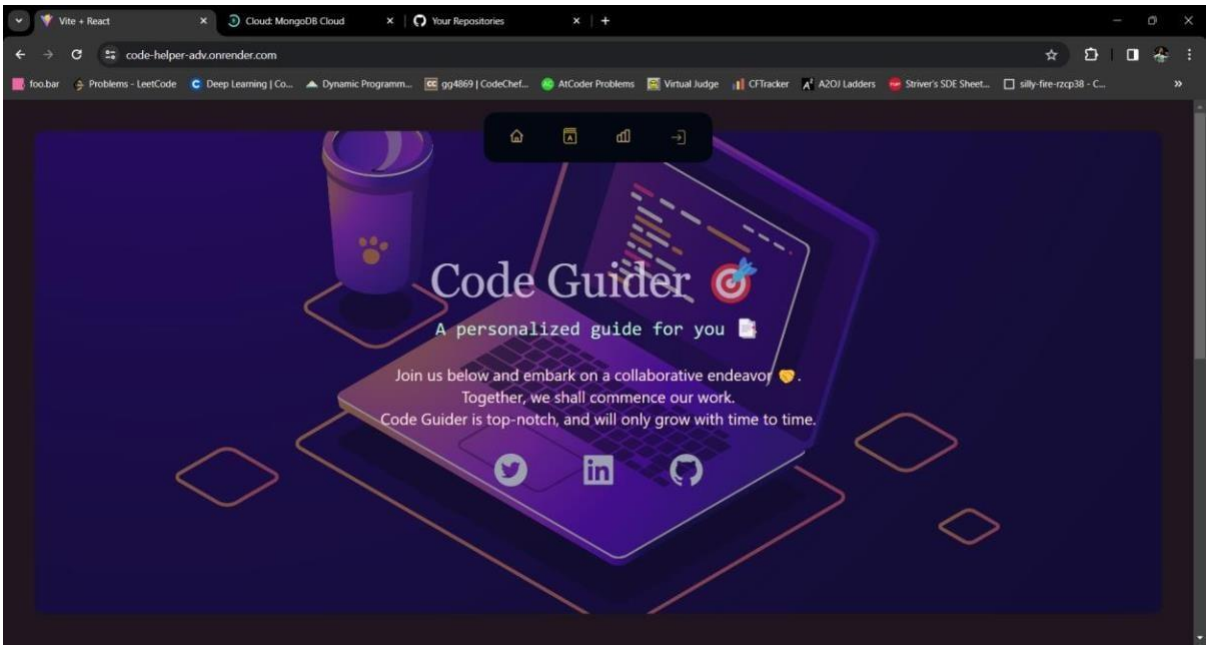
G2

112115046

I HAD IMPROVED MY LAB3 ASSIGNMENT TO MAKE A PROJECT OF MY OWN, BUILT UPON THE SAME LAB3, WHICH IS WHY IT LOOKS DIFFERENT BUT CODE LOGICS STILL REMAIN SAME

DOCKERIZED SSO IMPLEMENTATION FOR A PERSONAL WEBSITE

Github link: https://github.com/dhruva4869/CC-LAB-Docker-SSO



HOMEPAGE WITH LOGIN BUTTON ON TOP RIGHT OF THE HEADER

Docker File used for the project that works is:

client > Dockerfile > ...

```
1  FROM node:18-alpine
2
3  WORKDIR /app
4
5  COPY package.json .
6
7  RUN npm install
8
9  COPY . .
10
11 EXPOSE 8081
12
13 CMD [ "npm", "run", "dev" ]
```

Since me project was built on vite, corresponding changes need to be done in the vite.config file as well. These are the changes:

Dockerfile U vite.config.js M

client > vite.config.js > default

You, 2 seconds ago | 1 author (You)

```
1  import { defineConfig } from "vite";
2  import react from "@vitejs/plugin-react";
3
4  export default defineConfig({
5   base: "/",
6   plugins: [react()],
7   preview: {
8    port: 8081,
9    strictPort: true,
10  },      You, 1 second ago • Uncommitted cha
11  server: {
12   port: 8081,
13   strictPort: true,
14   host: true,
15   origin: "http://0.0.0.0:8081",
16  },
17 });
```

```
> server@1.0.0 dev
> nodemon index.js

[nodemon] 3.0.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Sever running at Port: 8000
Database Connected ac-6hh2kfi-shard-00-00.hfc269j.mongodb.net
```

Server running

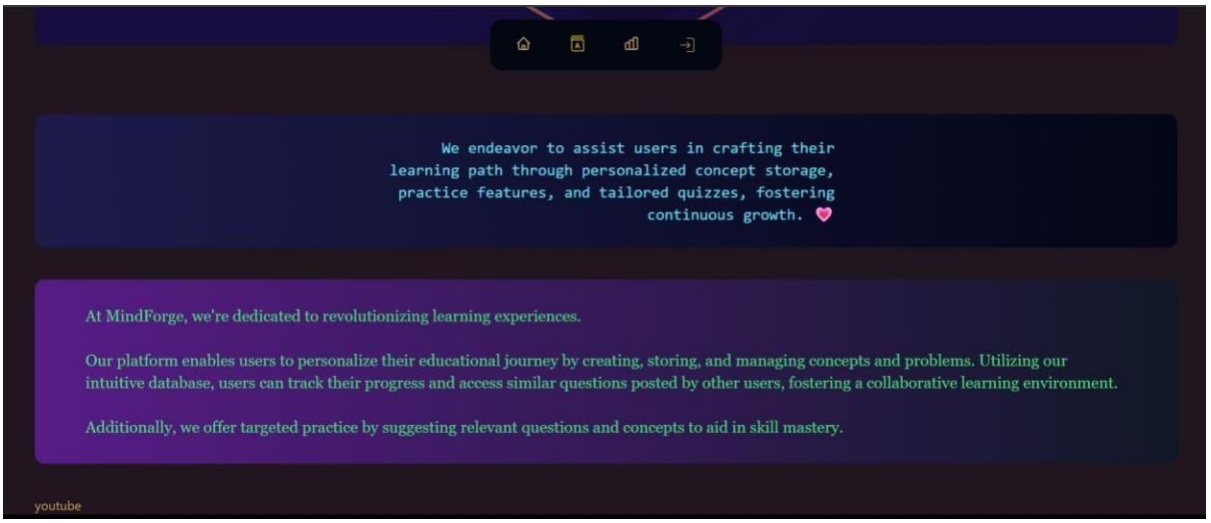
```
okay4869@LAPTOP-S17LHP1H:/mnt/c/Users/Dhruva/Desktop/MERN RADDY-myself/client$ docker images
REPOSITORY       TAG       IMAGE ID       CREATED              SIZE
sample-project   v1.0      b08c3247dcc2   About a minute ago   1.59GB
```

Docker Images and building from the docker. (Dockerization)

```
okay4869@LAPTOP-S17LHP1H:/mnt/c/Users/Dhruva/Desktop/MERN RADDY-myself/client$ docker build . -t "sample-project:v1.0"
[+] Building 9.5s (10/10) FINISHED                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                          0.2s
 => => transferring dockerfile: 174B                                                                          0.1s
 => [internal] load metadata for docker.io/library/node:18-alpine                                             5.8s
 => [internal] load .dockerignore                                                                             0.1s
```

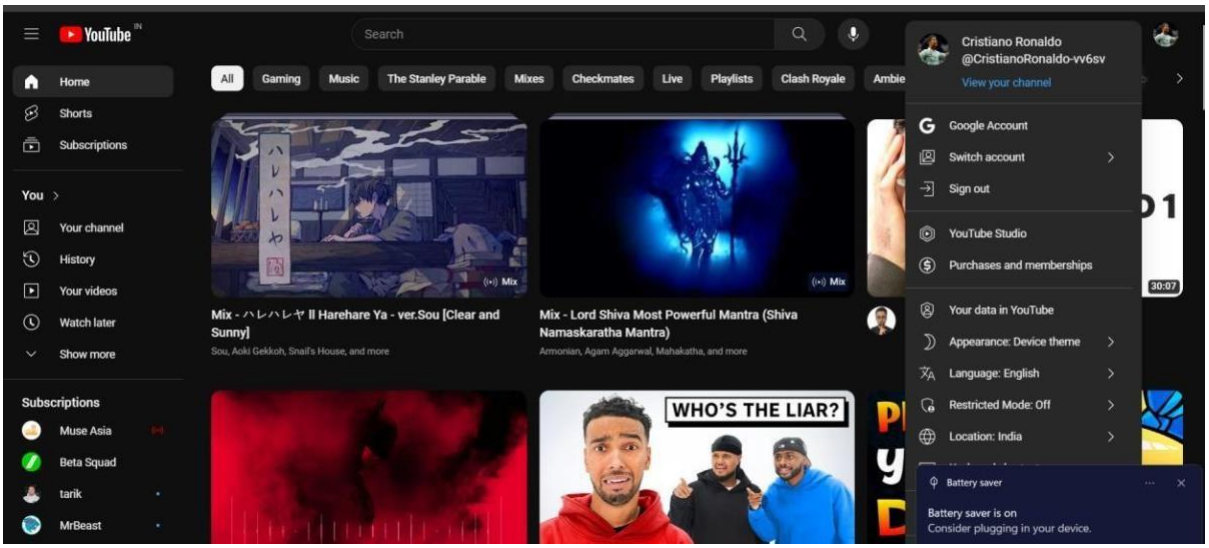
```
okay4869@LAPTOP-S17LHP1H:/mnt/c/Users/Dhruva/Desktop/MERN RADDY-myself/client$ docker run -p 8081:8081 sample-project:v1
.0

> client@0.0.0 dev
> vite


  VITE v5.2.9  ready in 2605 ms

  ➜  Local:   http://localhost:8081/
  ➜  Network: http://172.17.0.2:8081/
```

Running the website from the docker directly. Hosted in local host 8081 (visible in ss)



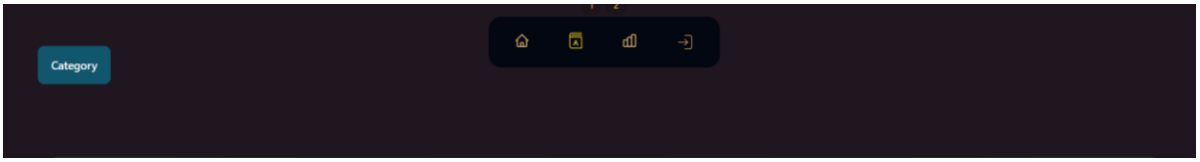
Youtube link. When we log in and click it, it automatically logs us in youtube as well.

Proof:



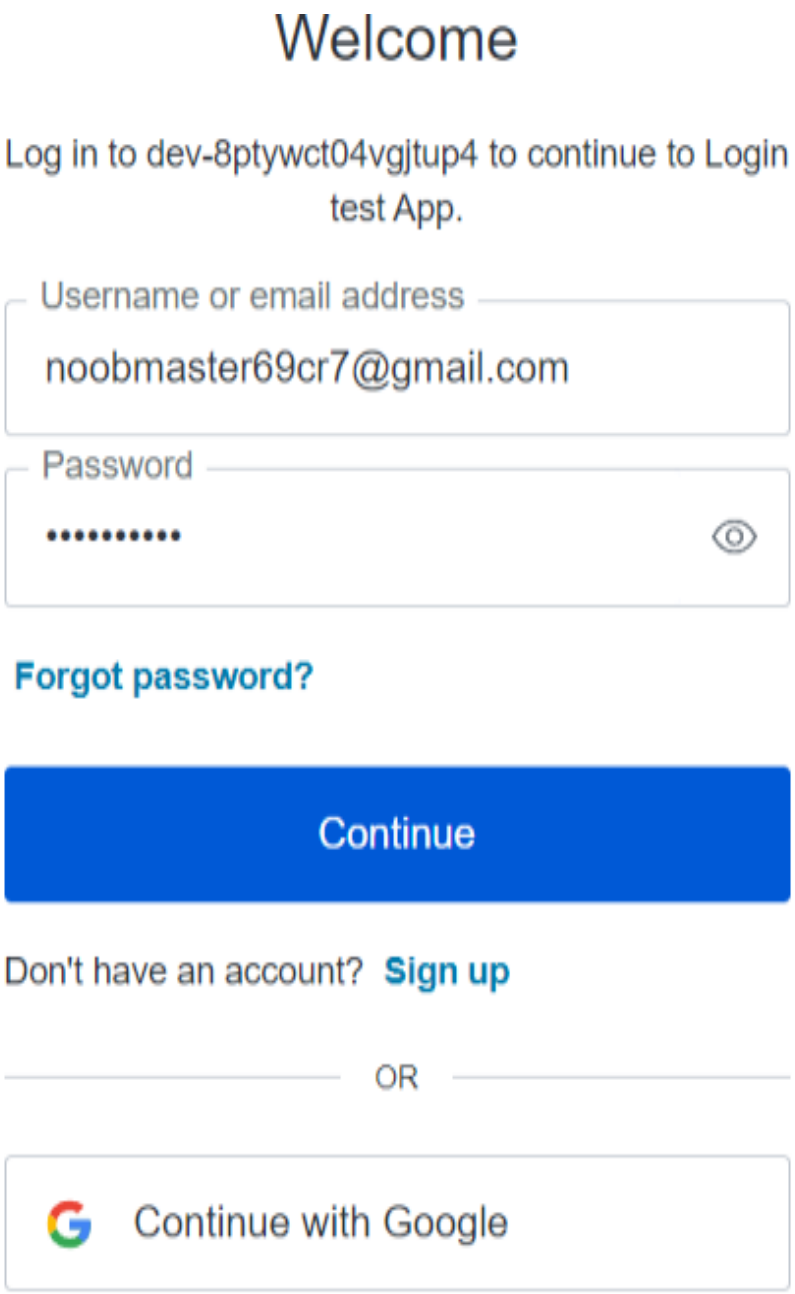
Showing Logged in with my account.

On top of that sso can do many other functionalities. I have shown them all below how SSO helps in my own application

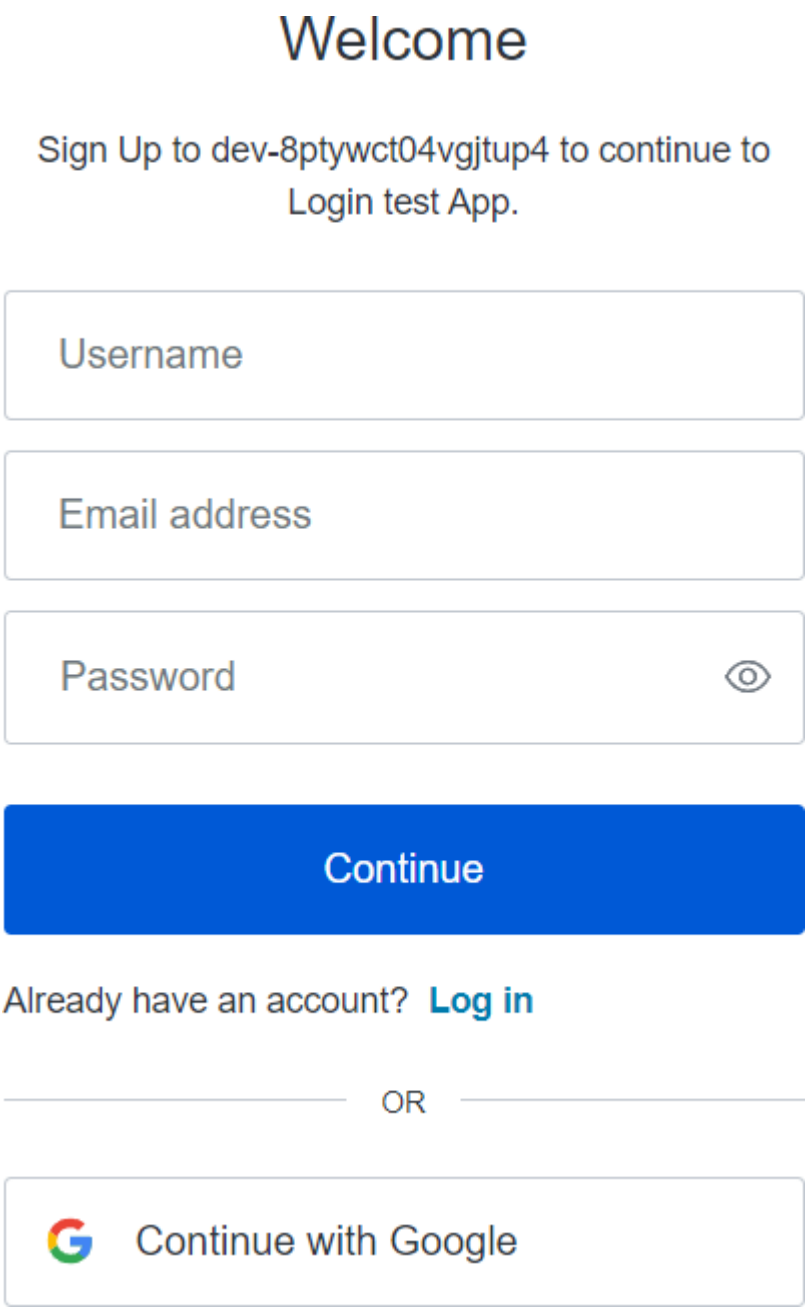


In the problems, page no button to add a new problem available.

This is because we need to login to get this functionality



Login Page here with a signup functionality and forgot password button too.



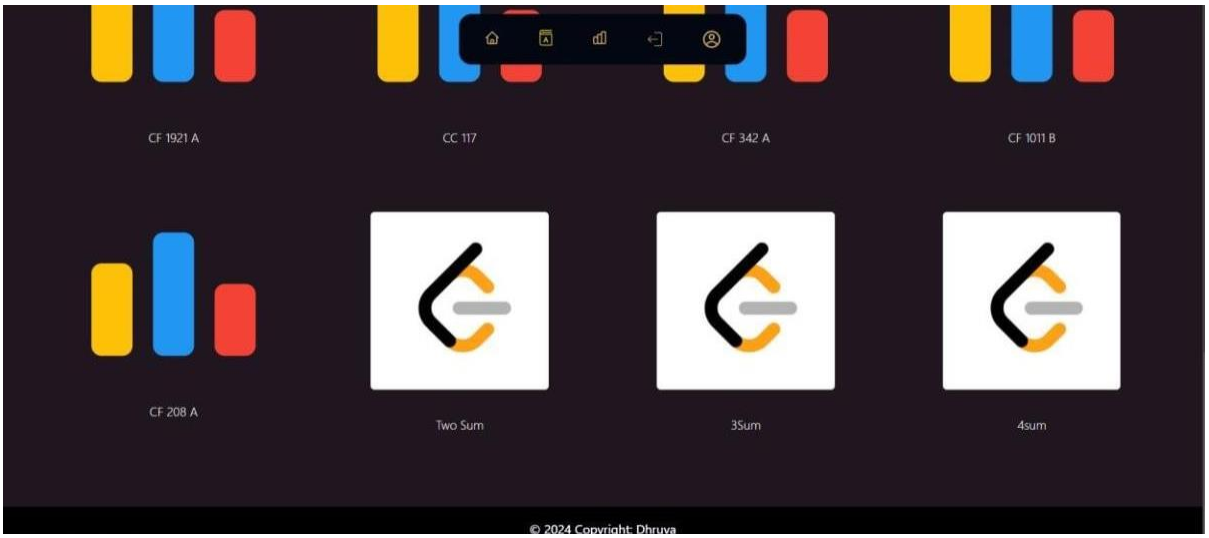
Signup Page with login with google available too.

After login, the header changes to profile and logout symbols

Back to homepage after logging in

Now Add Problem part is visible. Users can add their problems and the details freely.

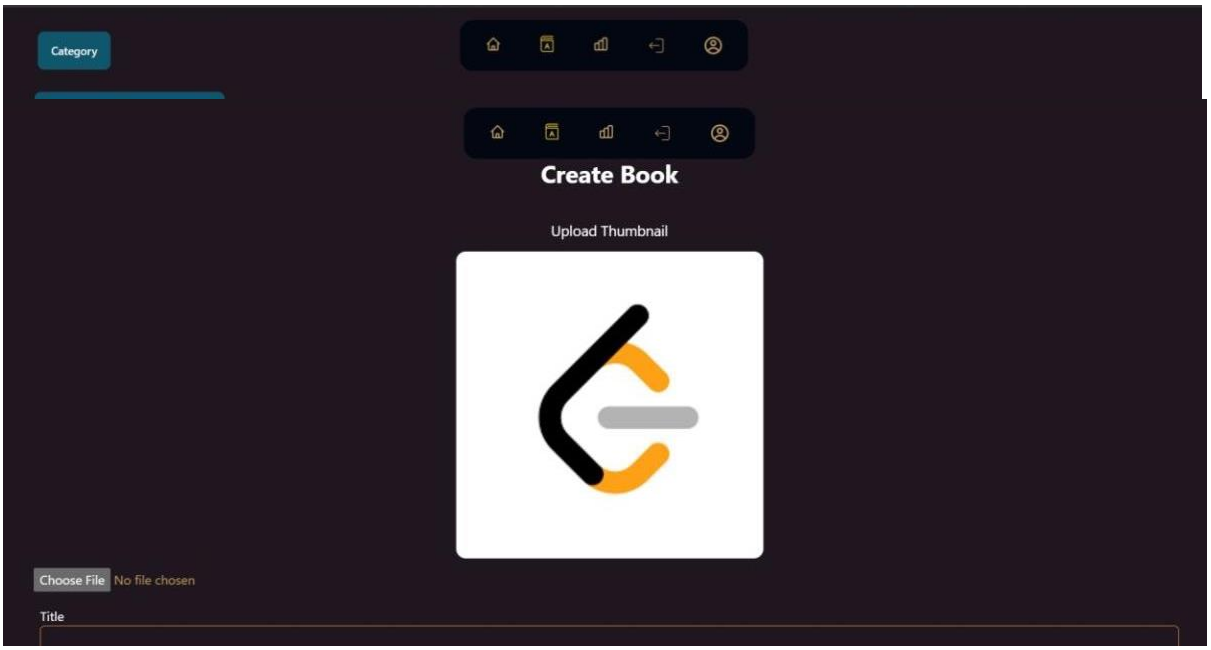
Problems that I have added for 1 user. Some of the problems are shown by



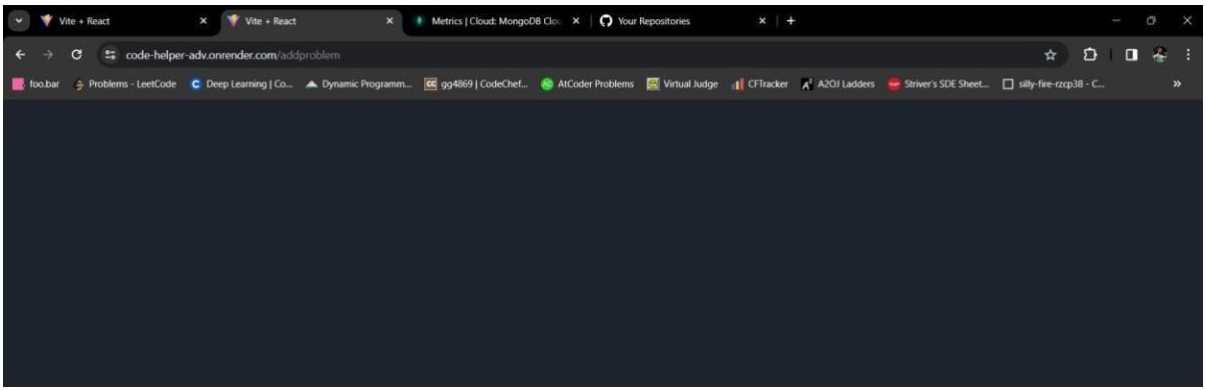
default for each user as well for enhanced experience

Page after refresh in any other tabs will still be logged in, meaning that the login session is found immediately and it is not expired till the user truly logs out.

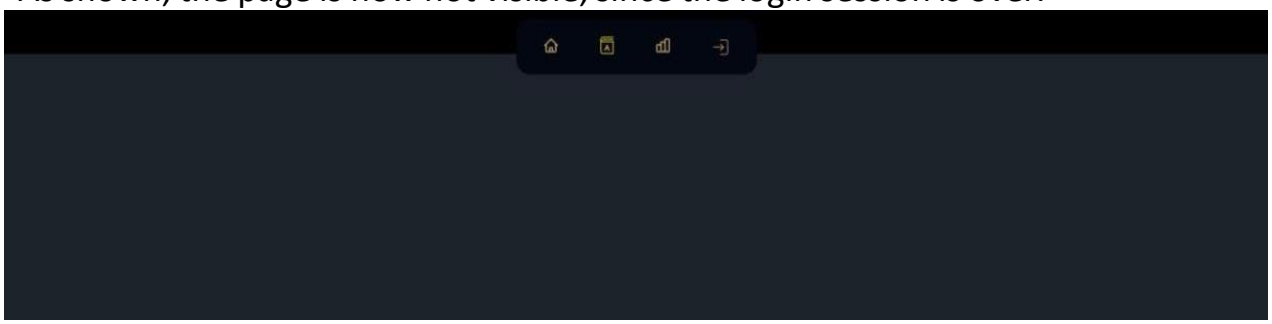
“Add Problem” page:



Now let us logout and check the same page.



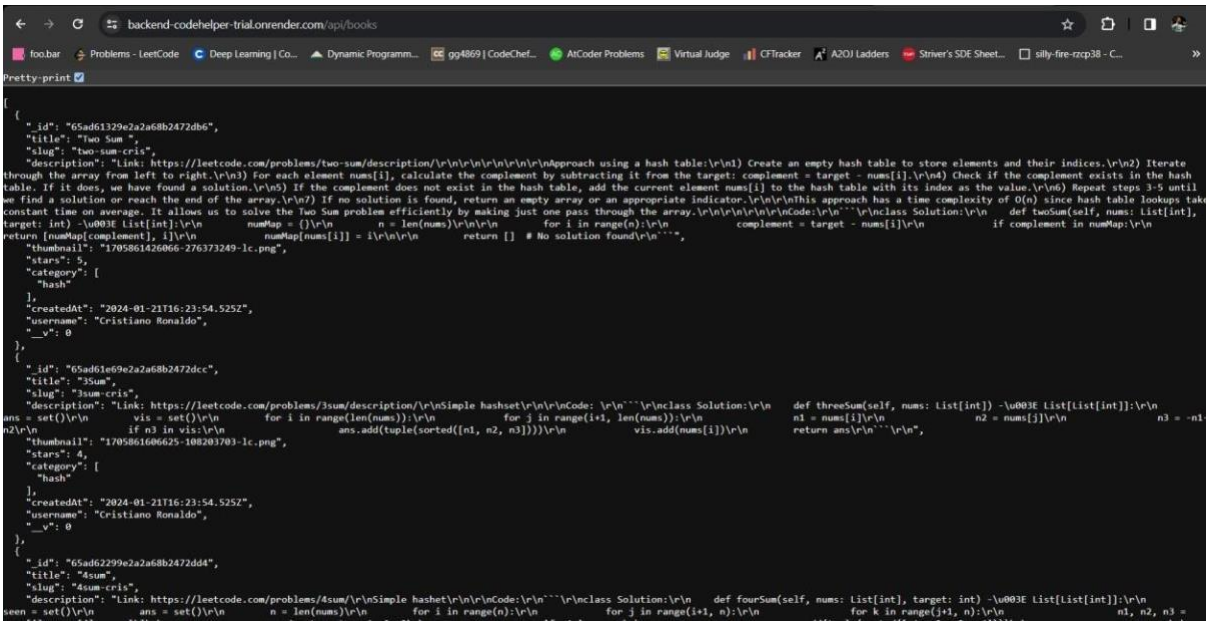
As shown, the page is now not visible, since the login session is over.

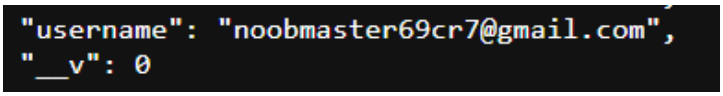


Profile page also not visible since user not logged in.

Let us check the database after we add a problem. It should also have the username in it.

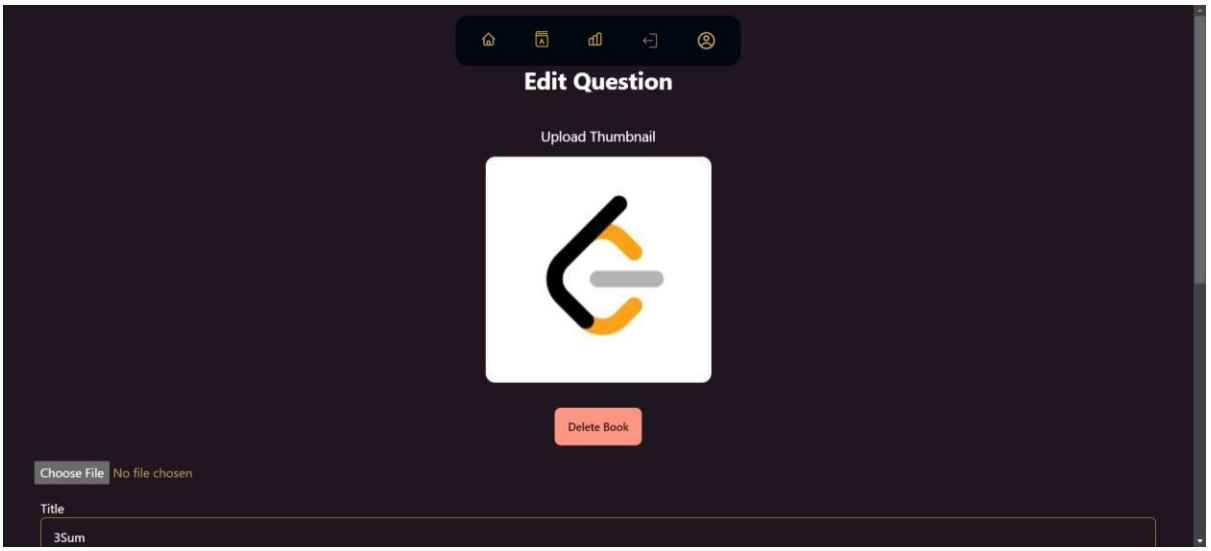
Backend storage: https://backend-codehelper-trial.onrender.com/api/books





Username shown as well for different users.

Along with add Problem, Delete and Update functionalities are also integrated.



Edit and Delete functionalities built in as well. Again, SSO provides that a user not logged in cannot access this link.

Code snippets implementing SSO. Already there in the Github link.

```jsx
import { useAuth0 } from "@auth0/auth0-react";
import React from "react";

const Login = () => {
  const { loginWithRedirect, isAuthenticated  } = useAuth0();

  return !isAuthenticated && (
    <button className="btn btn-success text-5xl" onClick={() => loginWithRedirect()}>Log In</button>
  );
};

export default Login;
```

Code Blame 14 lines (12 loc) · 388 Bytes Code 55% faster with GitHub Copilot Raw

```jsx
import React from 'react';
import { useAuth0 } from "@auth0/auth0-react";

function Logout() {
  const { logout, isAuthenticated } = useAuth0();

  // Show the logout button only if the user is authenticated
  return isAuthenticated && (
    <button onClick={() => logout({ logoutParams: { returnTo: window.location.origin } })}>
      Log Out
    </button>
  );
}
export default Logout;
```

Fetch Data:

```
const fetchData = async () => {
  try {

    let url = baseUrl;
    if (selectedCategory) {
      url += `?category=${selectedCategory}`
    }

    const response = await fetch(url);

    if (!response.ok) {
      throw new Error("Failed to fetch data.");
    }

    const jsonData = await response.json();
    setData(jsonData);
    setIsLoading(false);
  } catch (error) {
    console.log(error);
    setError("Error fetching data. Please try again later.");
    setIsLoading(false);
  }
};
fetchData();
}, [selectedCategory, searchTerm]);
```

Data fetch for only the current username

```
const curr = user ? user.name : "";



const newData = data.filter(item => item.username === curr);
```

Checking authentication before displaying the button

```
{
  isAuthenticated && <Link to="/addproblem" className="text-2xl"> <button className="btn btn-accent w-2/12">+Add Problem</button></Link>
}
```

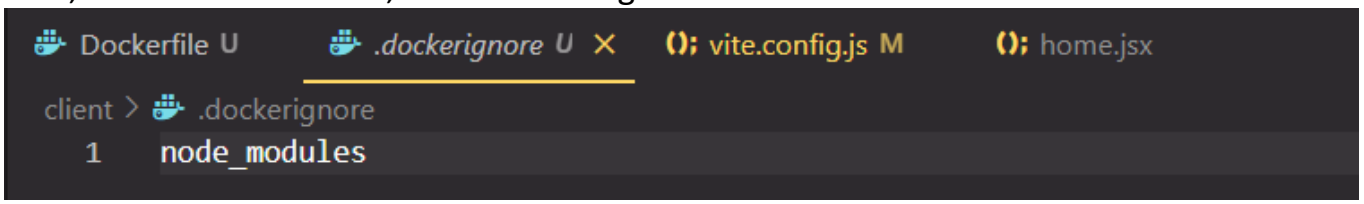
Delete and edit operations:

```
try {
  const response = await fetch(
    "https://backend-codehelper-trial.onrender.com/api/books/" + bookId,
    {
      method: "DELETE",
    }
  );
```

Again, edit problem only visible if authenticated and logged in:

```
<img className="mx-auto py-12 min-h-36 min-w-36 content-center items-center justify-center" src={`https://
alt={data?.title} />
{isAuthenticated && <Link to={`/editproblem/${data.slug}`} className="text-xl">Edit</Link>} <br/><br/>
<Link to="/problems">Books </Link>
```

Also, if docker takes time, add a .dockerignore file as shown:



DONE BY:

DHRUVA PRASAD UPADHYAYA
112115046
G2
CSE

Github Link: https://github.com/dhruva4869/CC-LAB-Docker-SSO