Architecture

Django Rest Framework, with djoser Api token-based authentication.

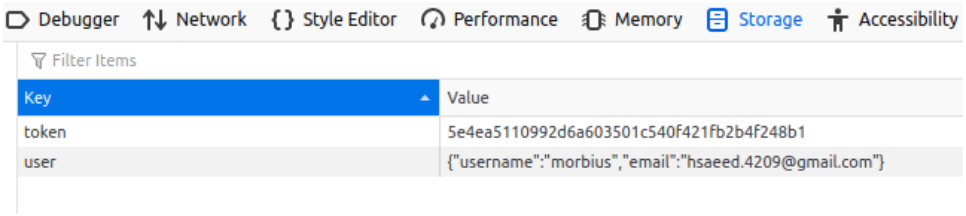
Method + Endpoints	Description
Post: api/users/	Creating a new user
Get: api/users/me/	Gets user instance
Post: api/token/login/	Retrieve user token. (Used hooks – Frontend)
Post: api/token/logout/ (Unused)	Logs user out
Post: api/users/set_passowrd/	Changes users password from settings page
Post: api/users/set_username/	Changes uerers username from settings page

Sample Usage

Request made to /api/token/login/ (login.js). From the React front end, axios makes a post request to the server, submitting the user's data. Username and password are required credentials for a user to be logged in. The response is the users token which is then stored in the data variable. This is how authentication is done.

```
export default function Login() {
  const { setUserDetails } = useAuthCtx();
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);
  const submitHandler = async (formData) => {
    try {
      const { username, password } = formData;
      setLoading(true)
      const { data } = await axios.post('/api/token/login/', { username, password })
      if (data?.auth_token) {
        const token = `${data?.auth_token}`;
        window.localStorage.setItem('user_token', token)
        setUserDetails({
          username,
          token,
        });
      }
      setLoading(false)
    } catch (err) {
      setLoading(false)
      const data = err?.response?.data || {};
      const msgs = Object.entries(data)
        .map(([key, arr]) => Array.isArray(arr) ? `${key}: ${arr[0] || 'is required'}` : `${key}: is required`)
        .join("\n");
      setError(`${msgs}`);
    }
  };
```

The following is an image of local storage setting the user in the browser after clicking on inspect. The token value along with the username and email is set (this is from initial build of authentication system).



The unused token/logout/ logout route was unused in the production build, instead of making the request to the server with that url the following was done:

```
  const logout = () => {
    setAxiosAuthToken(undefined);
    setUser(null)
    window.localStorage.removeItem("user_token")
  };
  const isAuthenticated = !!user?.token;
  return (
    <AuthContext.Provider
      value={{ user, isAuthenticated, setUserDetails, logout }}
    >
      {children}
    </AuthContext.Provider>
  );
};
```

The auth-token is set to null and removed from local storage in the browser. The user is then set to null, and values from local storage are removed. [Hint: closing the browser before logging out may keep the token value in local storage and prevent sign in]