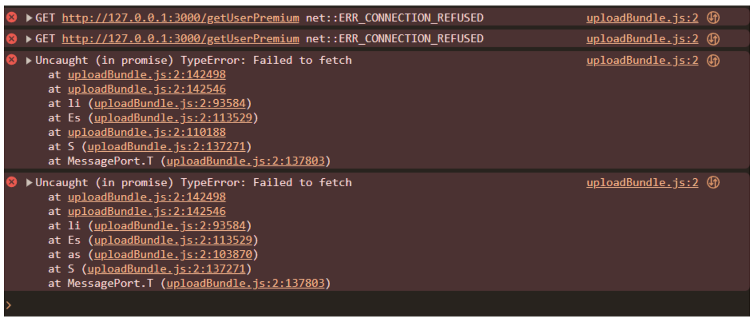
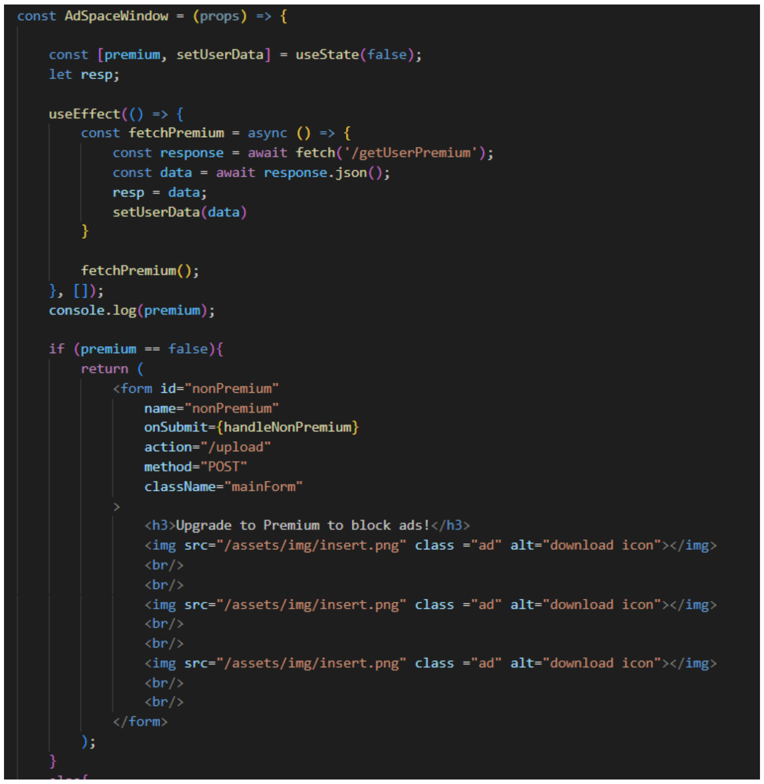
- What is the intended purpose of the application?
    - The purpose of the application reflects a need that my team at work was looking for. We are looking for a solution to online learning attendance that has to be taken for training purposes. The user inserts excel sheets into a database in which they are able to search a user's name and see which courses they have taken. For this project, you can use this for various purposes, essentially allowing the user to create a database off of excel sheets and apply search functionality as needed.
- React
    - React is being used to display the bulk of the HTML as well as handle advertisements.
    - Components: Delete Account, Reset Password, Premium Signup, Ad space, Upload window (error), 404 (error), Login, Signup
- MongoDB
    - Mongo is storing all excel data as well as account data
- What went right?
    - Excel usage
        - The excel node package and express work fantastically. The search functionality functions properly as well, making my core application functional
    - React
        - For the majority of my components, react was functional and things such as deletion and premium account sign ups work correctly.
    -
- What went wrong? (a lot)
    - Xlsx syntax errors
        - For the majority of this project, I spent time on the core functionality and dealing with excel spreadsheets using express and the xlsx library. Due to this long hold up and an extreme lack of resources, much of the project was delayed until I could utilize xlsx properly. It was not until a week before the project was due until I stumbled upon a Github post highlighting my very niche issue
    - Transfering express functionality into React
        - Me and another student worked to figure out how to transfer file data from React to the server. While the other student had figured this issue out, I ran into a roadblock where I had received a plethora of the following errors below:

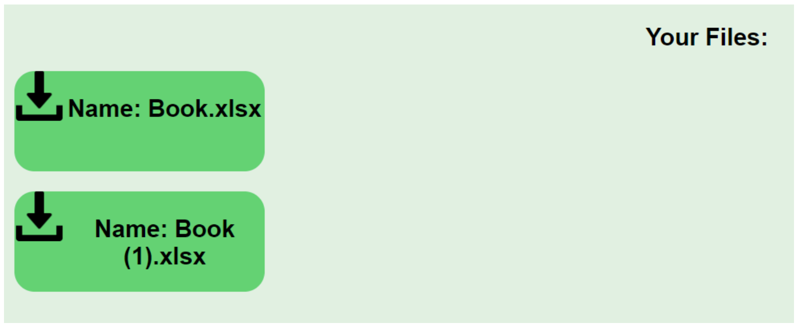


```
▶GET http://127.0.0.1:3000/getUserPremium net::ERR_CONNECTION_REFUSED          uploadBundle.js:2
▶GET http://127.0.0.1:3000/getUserPremium net::ERR_CONNECTION_REFUSED          uploadBundle.js:2
▶Uncaught (in promise) TypeError: Failed to fetch                              uploadBundle.js:2
    at uploadBundle.js:2:142498
    at uploadBundle.js:2:142546
    at li (uploadBundle.js:2:93584)
    at Es (uploadBundle.js:2:113529)
    at uploadBundle.js:2:110188
    at S (uploadBundle.js:2:137271)
    at MessagePort.T (uploadBundle.js:2:137803)
▶Uncaught (in promise) TypeError: Failed to fetch                              uploadBundle.js:2
    at uploadBundle.js:2:142498
    at uploadBundle.js:2:142546
    at li (uploadBundle.js:2:93584)
    at Es (uploadBundle.js:2:113529)
    at as (uploadBundle.js:2:103870)
    at S (uploadBundle.js:2:137271)
    at MessagePort.T (uploadBundle.js:2:137803)
>
```

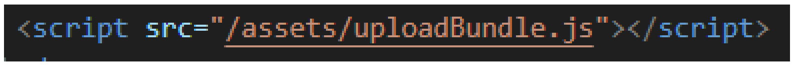
- Advertisements and premium toggling
    - When working with advertisements and premium content, I attempted to use useEffect and useState to gather server information on whether or not the user had a premium account. This would dynamically toggle the premium option depending on the user's account. However, the updating would not properly work, leaving the advertisements off of the page but still there (in the code). This portion of the code is below.
    - 

```
const AdSpaceWindow = (props) => {

    const [premium, setUserData] = useState(false);
    let resp;

    useEffect(() => {
        const fetchPremium = async () => {
            const response = await fetch('/getUserPremium');
            const data = await response.json();
            resp = data;
            setUserData(data)
        }

        fetchPremium();
    }, []);
    console.log(premium);

    if (premium == false){
        return (
            <form id="nonPremium"
                name="nonPremium"
                onSubmit={handleNonPremium}
                action="/upload"
                method="POST"
                className="mainForm"
            >
                <h3>Upgrade to Premium to block ads!</h3>
                <img src="/assets/img/insert.png" class ="ad" alt="download icon"></img>
                <br/>
                <br/>
                <img src="/assets/img/insert.png" class ="ad" alt="download icon"></img>
                <br/>
                <br/>
                <img src="/assets/img/insert.png" class ="ad" alt="download icon"></img>
                <br/>
                <br/>
            </form>
        );
    }
```

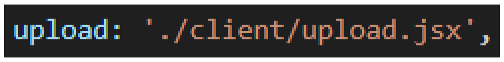
- Dynamic files
    - I had attempted to make it so that when the user searches for files, those files would show at the bottom of the page. This would also work when a file was uploaded. While this did function, I had no luck finding a way to make the file downloadable from the small widgets at the bottom of the page shown below



    - Project scope and time mismanagement
        - I had come to find out quickly that I would need more time on this project. I had spent way too much time working on the main functionality of the project, (ie the search functionality, files and xlsx, getting the dynamic file display to function, etc), and not enough time working on the smaller components such as premium functionality, advertisements, account deletion, etc. This caused extreme time constraints and a lack of support on concepts that I hadn't known that I was not familiar with, leading to missing functionality.
    - React
        - I had spent approximately 6-8 hours on react bug fixes, when I had been missing the following components from my code:



```
<script src="/assets/uploadBundle.js"></script>
```



```
upload: './client/upload.jsx',
```

        - This led to a monumental loss of time due to my lack of knowledge
- What did I learn?
    - Client and server relationships
        - React, handlebars, and node are all very understandable to me know
    - Xlsx and Excel
        - I got to learn a lot about how excel can be parsed as well as how to use more niche libraries
    - Time management
        - My major issue in this project. I had allotted too much time to things that were beyond the scope of this project, leading to a large amount of unfinished work that I could have done so much better on.
- How would I continue to improve?
    - Finish the core pieces of the project.
    - Make the file viewer much more intricate
    - Learn more about React and how I can use it to my advantage
    - Add multiple storage spaces for different excel libraries
    - Use time more efficiently
- How did I go above and beyond?
    - Xlsx node library
    - Express file uploader
- Code fragments
    - Ad space: https://stackoverflow.com/questions/75067533/add-borders-to-sides-of-screen-in-css

Endpoint Documentation

URL: /login
Supported Methods: GET & POST
Query Params: request , response
Descriptions: Displays login page and login info
Return Types: Redirect

URL: /signup
Supported Methods: POST
Query Params: request , response
Descriptions: Displays sign up page and info
Return Types: Redirect

URL: /logout
Supported Methods: GET
Query Params: request , response
Descriptions: Logs user out and takes user back to login page
Return Types: Redirect

URL: /deleteAccount
Supported Methods: POST
Query Params: request , response
Descriptions: Delete user account and redirects to signup page
Return Types: Redirect

URL: /upload
Supported Methods: POST & GET
Query Params: request , response
Descriptions: uploads file and takes user to main page
Return Types: Redirect

URL: /search
Supported Methods: GET
Query Params: request , response
Descriptions: Converts excel sheets to json for parsing and searching
Return Types: File Array (Excel MimeType)

URL: /getUserPremium
Supported Methods: GET
Query Params: request , response
Descriptions: Gets whether or not the user has a premium membership for React usage
Return Types: Boolean

URL: /resetPass
Supported Methods: POST
Query Params: request , response
Descriptions: Resets the user’s password
Return Types: Redirect

URL: /premiumSignup
Supported Methods: POST
Query Params: request , response
Descriptions: Allows user to sign up for premium membership
Return Types: Redirect