Quiz Section 3: React

In this section, we will create a simple Book Review App that will allow a user to add a book to a list of books and rate the books (upvote or downvote). This application includes both a client and server. Remember that **understanding** how all the parts work and interact is necessary before you can start debugging.

To get started with the debugging problems, check out the starter code using the command `git clone https://gitlab.cs.washington.edu/cse331-24au-materials/sec03.git`

To install the modules, `cd server` and run `npm install --no-audit`, then `cd ../client` and run the same. Then, `cd ../server` and start it with `npm run start` (or `npm run build` and `npm run server` on Windows). Then, in a new terminal, `cd client` and start it with `npm run start`. Point your browser at `http://localhost:8080`.

You should now see the "Book Review" title when you open the page.

To access the **Debugging Log**, navigate to https://comfy.cs.washington.edu/service/hw3-practice.

Task 1 – That Doesn't Load Well

The application says "Loading Book Review list..." and the message does not go away. That does not look right!

(a) What should the application be doing here, if it were working properly, to remove the "Loading" message? (Demonstrate that you understand the high level idea of how the application works when it starts up.)

(b) Debug to identify the error in the code that is causing this failure. What is the bug?

(c) Fix the bug identified in part (b). Be sure to add an entry to the Debugging Log.

Task 2 – Breaking Add

Now the loading message disappears, and we see an empty list of books, but when we try to add our first book to the list, it doesn't work.

(a) What should the application be doing here, if it were working properly, to add the new book to the list?

(b) Debug to identify the error in the code that is causing this failure. What is the bug?

(c) Fix the bug identified in part (b). Be sure to add an entry to the Debugging Log.

Task 3 – Voto

Now, books show up properly when we add them, but when we try to upvote or downvote them, it doesn't work as expected.

(a) What should the application be doing here, if it were working properly, to update the book's rating when you upvote or downvote it?

(b) Debug to identify the error in the code that is causing this failure. What is the bug?

(c) Fix the bug identified in part (b). Be sure to add an entry to the Debugging Log.

Task 4 – Updating...

Now, when you when attempt to upvote or downvote a book, the UI updates to "Updating..." and then suddenly drops the book from the list. This is not the intended behavior. It should updating the UI to "Updating..." and then update the book's rating.

(a) What should the application be doing here, if it were working properly, to update the book's rating after 2 seconds?

(b) Debug to identify the error in the code that is causing this failure. What is the bug?

(c) Fix the bug identified in part (b). Be sure to add an entry to the Debugging Log.

(d) This tight coupling between the client and server, with both having to know to pause for 2 seconds, seems problematic. If we were decide that we wanted to keep items around on the server for 3 seconds instead, the client would be broken again!

How could we decouple the client and server so that only one of them (say, the server) has to know how long to pause for?

Task 5 – I Have an Add Feeling About This

To make the application a bit more user-friendly, let's add an informational message after a new book is added.

To do so, add the following code at the end of `doAddJson`:

```
// Add a message about the added book.
this.setState({message: `Added ${data.name}. ` +
    `List now contains ${this.state.books.length} books.`});
```

Unfortunately, when you try it out, something is wrong: it's showing the wrong number of books!

(a) What is the bug?

(b) Fix the bug identified in part (a). Be sure to add an entry to the Debugging Log.

(c) This code is generating what is effectively UI (the informational message) inside of an event handler. It would be more sensible to be generating UI inside of render.

We could do that, for example, by replacing the `message` field of the state with a `justAdded` field that stores the name of the item just added. Then, we would change `renderMessage` to the following:

```
renderMessage = (): JSX.Element => {
    if (this.state.justAdded === "") {
        return <div></div>;
    } else {
        return (<p>Added {this.state.justAdded}.
            List now contains {this.state.books.length} books.</p>);
    }
};
```

Why would doing it that way instead have avoided this bug?