Quiz Section 3: React – Solutions

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/cse331-25sp/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.)

It is supposed to load the current list of books from the server. When it gets that response, it removes the loading message and displays the books. That request/response must be failing for some reason.

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

We are getting a 404 message for the request. That means we typed an incorrect URL.

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

Change `"/lists"` to `"/list"` in `doRefreshTimeout`.

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?

It is supposed to send an `/add` request to the server and, once that completes successfully, add it to the list of books displayed in the UI.

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

The network request completes successfully, and refreshing the page shows the item on the list, so the request is working properly. The problem is that the client isn't updating its state to show the new book.

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

Add the following code at the end of `doAddJson`:

```
const books = this.state.books.concat([{
    name: data.name,
    upvotes: data.upvotes,
    downvotes: data.downvotes,
    updating: false,
}]);
this.setState({ books: books, newName: "" });
```

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?

> It is supposed to send a `/vote` request to the server with the vote type (upvote or downvote) and, once that completes successfully, update the book's rating in the UI.

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

> The network request fails with a 400 status code, indicating that the client send a bad request. The body of the response indicates a problem with "name", but the client code isn't sending a name, it is sending a "bookName" field.

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

> Change `bookName` to `name` in the `doVoteClick`.

Task 4 – 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?

> this.state.books is the still old list of books, not the new one with an added book.

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

> Change `this.state.books` to `books`, the variable storing the new list.

(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?

In render, `this.state.items` would be the correct value.

More to the point, the UI generated by `renderBooks` is currently using `this.state.books`, so if `renderMessage` was changed to also use `this.state.books`, the two parts of the UI would be guaranteed to match each other. (The mismatch was really the failure that users saw.)