CSE 331: Software Design & Implementation

Section 9

In this section, we will revisit our first example application that includes both a client and a server, the To-Do List, and practice debugging in a more complicated environment. Remember that **understanding** how all the parts work and interact is necessary before you can start debugging. If any parts of the To-Do List application are currently unclear, start by reviewing the material describing how they work.

To get started with the debugging problems, check out the starter code using the command

```
git clone https://gitlab.cs.washington.edu/cse331-24sp/cse331-24sp-materials/sec-debug.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 "To-Do List" title when you open the page.

1. That Doesn't Load Well

The application says "Loading To-Do 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).

2. Breaking Add

Now the loading message disappears, and we see an empty to-do list, but when we try to add our first item to the to-do list, it doesn't work!

(a) What should the application be doing here, if it were working properly, to add the new item 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).

3. A Pain in the Check

Now, items show up properly when we add them, but if we try to mark an item as completed, it doesn't work!

(a) What should the application be doing here, if it were working properly, to mark the item as completed?

(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).

4. Pull Out All the Drops

Now, the item you click on is marked completed in the UI, but it is never dropped from the list unless you refresh the page. It's supposed to be dropped automatically!

(a) What should the application be doing here, if it were working properly, to remove the completed item after 5 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).

(d) This tight coupling between the client and server, with both having to know to pause for 5 seconds, seems problematic. If we were decide that we wanted to keep items around on the server for 8 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?

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 item is added.

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

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

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

(a) What is the bug?

(b) Fix the bug identified in part (a).

(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.items.length} items.</p>);
  }
};
```

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