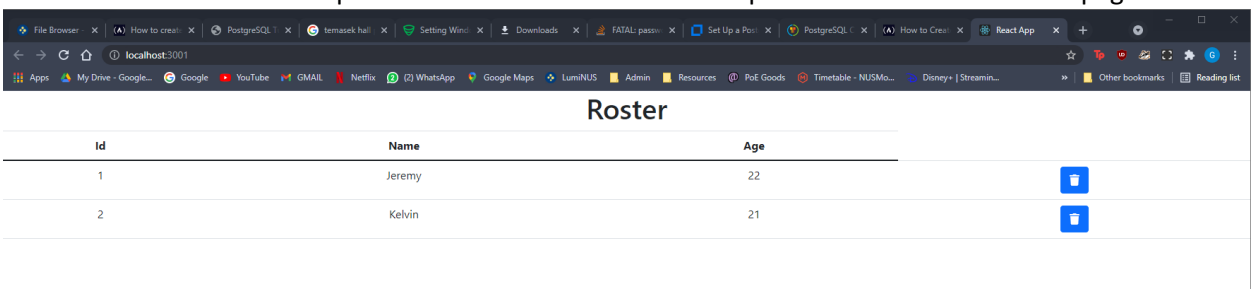
SUBMISSION FOR B4

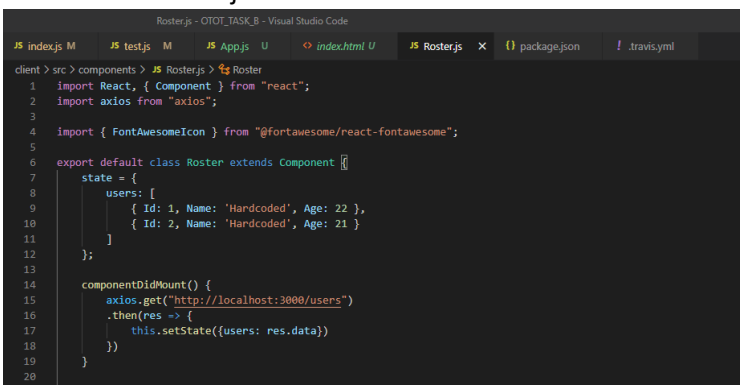
Name: Gordon Foo
Matriculation Number: A0199554L
Github Link: https://github.com/gordonfgz/CS3219_OTOT_TASK_B.git

1. Using React frontend to interact with backend API

Created a Roster class that queries from a backend server and produces the data on a webpage:



Code lies in Roster.js



```
import React, { Component } from "react";
import axios from "axios";

import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";

export default class Roster extends Component {
    state = {
        users: [
            { Id: 1, Name: 'Hardcoded', Age: 22 },
            { Id: 2, Name: 'Hardcoded', Age: 21 }
        ]
    };

    componentDidMount() {
        axios.get("http://localhost:3000/users")
        .then(res => {
            this.setState({users: res.data})
        })
    }
```

If results were not queried properly, then the webpage would display the name as “Hardcoded”.
Evidently, it isn’t hardcoded because the users array in Roster.js is updated by accessing the get api in index.js in the backend by using axios.get()

```
const e = require('express');
const express = require('express');
const cors = require("cors");

const app = express();
app.use(express.json());
app.use(express.urlencoded({extended: false}));

app.use(cors());

const PORT = process.env.PORT || 3000;

module.exports = app

var users = [
    { Id: 1, Name: 'Jeremy', Age: 22},
    { Id: 2, Name: 'Kelvin', Age: 21}
]

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.get('/users', (req, res) => {
    res.status(200).json(users)
})
```

Here is the index.js that has all the backend apis.

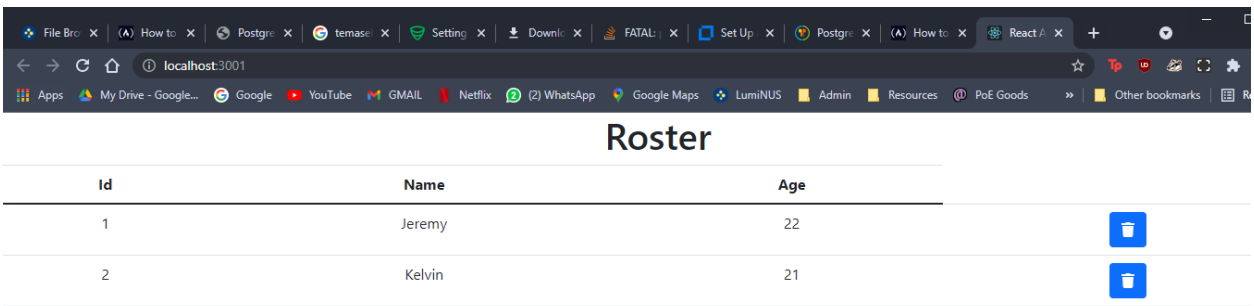
The get request of /users is called in Roster.js and that is how the webpage is able to produce data from the backend.

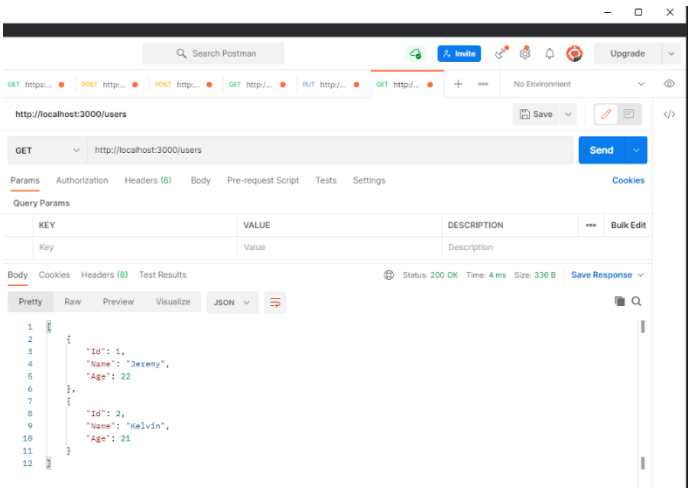
Roster class is imported into React App:

```
import 'bootstrap/dist/css/bootstrap.min.css';
import Roster from './components/Roster';

import { library } from "@fortawesome/fontawesome-svg-core";
import { faTrash, faPlus, faEdit } from "@fortawesome/free-solid-svg-icons";
library.add(faTrash, faEdit, faPlus);


function App() {

  return (
    <div className="App">
      <Roster></Roster>
    </div>
  );
}

export default App;
```

Delete button in webpage works with the backend:

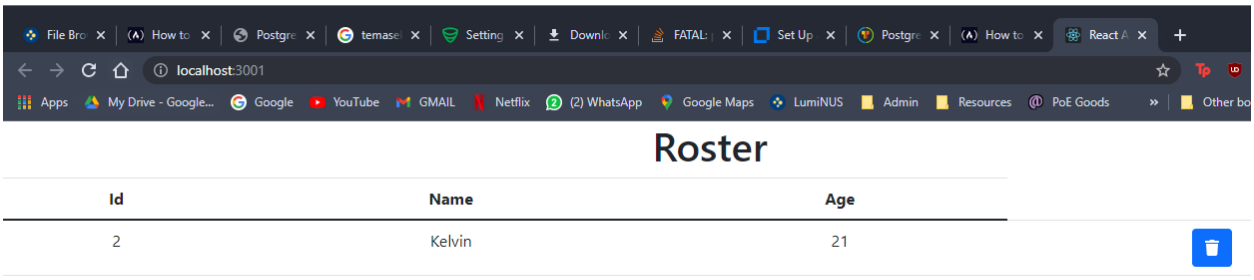
BEFORE PRESSING DELETE:

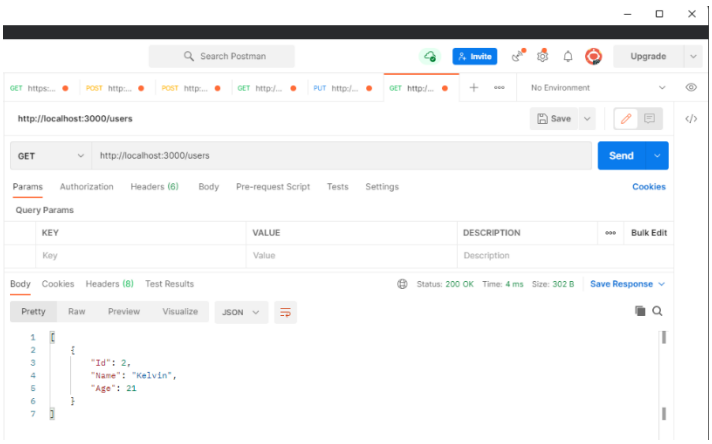




GET request from postman shows that there are 2 entities, inline with what we see in the webpage. Pressing delete on the webpage should result in the next GET request in postman showing only the name that is not deleted. (To prove that deletion happens in the backend and our front end managed to call an API that deals with the data)

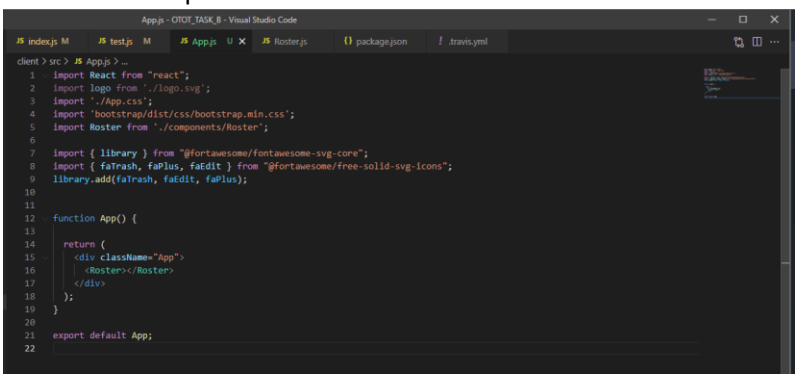
AFTER PRESSING DELETE UNDER JEREMY'S NAME:



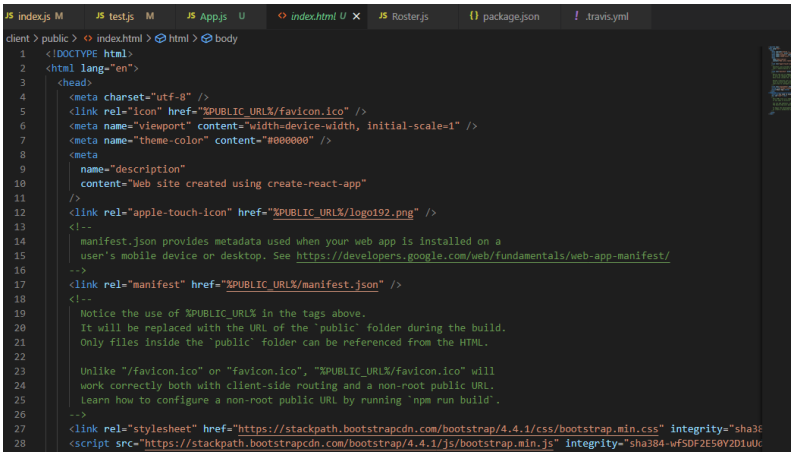


As you can see, after pressing the delete button, GET request only returns 1 entry. This means that the delete button on the webpage successfully called an API to update the "database" that an entry is deleted.

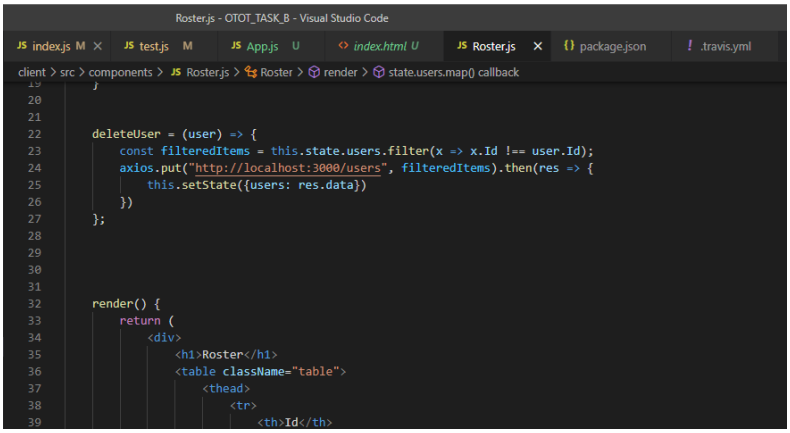
2.Used boostrap



imported boostrap in App.js



added boostrap into index



used bootstrap styling in Roster.js

Special thanks to:
https://www.freecodecamp.org/news/how-to-create-a-react-app-with-a-node-backend-the-complete-guide/
https://jinalshah999.medium.com/reactjs-step-by-step-tutorial-series-part-4-build-todo-application-using-reactjs-ab219c9b3608