

CS385 Mobile Application Development 2023/2024

Lab Session #4 - 27th October 2023

You must upload your solutions to Moodle (The link is on the Moodle Section for this Lab) by 15:59 Tuesday 7th November 2023. Remember, you can upload partial solutions. You do not need to upload fully working solutions. Please name your files as .txt files when you are uploading to Moodle.

Lab Description

Please refer to CS385 Lecture 9 materials for guidance. It is strongly suggested that you use this material as guidance rather than using Google or StackOverflow.

There is a piece of template code for the lab – it is available on Moodle and the filename is “**Lab4-Template-Code.js**”. This file is the basis for the lab.

There is a [GitHub folder available](https://github.com/petermooney/cs385/tree/main/trains) for this lab containing the relevant JSON data files. Please select ONE of these files for your API link (remember – you need to click on the file and then choose the ‘raw’ option).

<https://github.com/petermooney/cs385/tree/main/trains>

You are STRONGLY ADVISED to carefully examine the JSON response from our API links. The JSON will look like this:

```
{ "trainService" : [
  {
    "arrival" : "13:16",
    "departure" : "Frankfurt Höchst station",
    "destination" : "Hamburg Airport station",
    "details" : {
      "bikes" : "limited",
      "cars" : 20,
      "operator" : "Thalys International",
      "wifi" : "no"
    },
    "platform" : "1A",
    "serviceNumber" : "TR6995"
  },

```

TASK 1 – adapting the code to process the API links above

For this task you will need to perform the following steps:

- You will need to change the **URL** within `useEffect`. You can choose any of the JSON files within the GitHub folder provided.

- You will also need to ensure that the **setData** hook is provided with the correct JSON array. This was clearly explained at the end of Lecture 9 (please check the lecture notes)
- Add the necessary code to the **map** function in **TrainDisplayComponent** to render the **"operator"** property for each train journey.
- You are encouraged to try out different **JSON files** from the **GitHub folder** in order to test if your code is working properly.

Save your code in a text file called **Lab4-Task1-SOLUTION.js** – Alternatively, you can combine all of your tasks into one file if you wish.

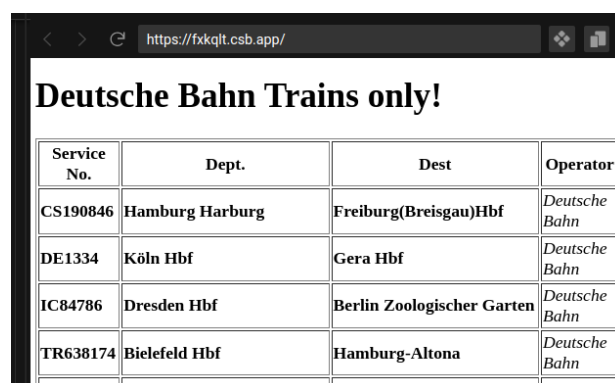
TASK 2 – adding a filter function to a new Component

We want to create a new component in our application. In this component we only want to display train services where the **operator** property from the JSON data contains **"Deutsche Bahn"**. We do not want to consider any other operator string value or even a variant of this.

For this task you must do the following:

- Create a new component called **DeutscheBahnComponent** which receives the JSON response as a **prop** variable.
- Within this component **create a filter function** which will only allow us to display or render train services from the operator **"Deutsche Bahn"**.
- For clarity you should delete the invocation of the **TrainDisplayComponent**. When you delete the line to invoke **TrainDisplayComponent**, CodeSandBox (or your IDE of choice) may complain that this component is declared but never used. It is OK, for now, to ignore this message.
- You are encouraged to try out different **JSON files** from the **GitHub folder** in order to test if your filter function is working properly.

Save your code in a text file called **Lab4-Task2-SOLUTION.js** – Alternatively, you can combine all of your tasks into one file if you wish. Your rendering should look something like the following:



Service No.	Dept.	Dest	Operator
CS190846	Hamburg Harburg	Freiburg(Breisgau)Hbf	Deutsche Bahn
DE1334	Köln Hbf	Gera Hbf	Deutsche Bahn
IC84786	Dresden Hbf	Berlin Zoologischer Garten	Deutsche Bahn
TR638174	Bielefeld Hbf	Hamburg-Altona	Deutsche Bahn
			Deutsche

TASK 3 – Using a button to allow the user select their chosen component.

After Task 2, you will have a new Component called `DeutscheBahnComponent` in addition to our original component `TrainDisplayComponent`.

We are going to follow a set of steps now to use a button to control conditional rendering of one of these components. This task involves you following a set of steps involving inserting new code into your existing application.

1. In the parent **App** component declare a new state variable called `showComp`. This will be a boolean variable. This variable should have an initial value of `false`. The code line will be `const [showComp, setShowComp] = useState(false);`
2. In the parent App component – go to the section where there is an if statement. We have an else statement for display of the component. Add the initial code for a button here as follows `<button>Change Display</button>`
3. Now we need to connect this button with an action or event. Our event is going to be simple – if the button is clicked change the boolean value of `showComp` to the opposite (if it is `true` then turn it `false` and vice versa). We need to write our button handler method or function in the App component. Do not write it within `useEffect`.
`function toggleDisplay() {`
`// here we toggle the boolean value of showComp`
`setShowComp(!showComp);`
`}`
4. Now we must connect this function or method with the `onClick` event for our button. We need to make a change in the button code.
`<button onClick={() => toggleDisplay()}>Change Display</button>`
5. Finally, we need to add some conditional rendering AFTER the button within the **App** component. We want to conditionally render the `DeutscheBahnComponent` component if `showComp` is true while we would like to render the `TrainDisplayComponent` component if `showComp` is false. Our code will look like the following:

```
<button onClick={() => toggleDisplay()}>Change Display</button>
{showComp && <DeutscheBahnComponent APIData={data} />}
{!showComp && <TrainDisplayComponent APIData={data} />}
```

6. Save you code and then reload your application – test out the button. If you have followed the steps correctly, you should able able to toggle the display of the components by using the button. This could be very useful to you in your project for CS385.

Save your code in a text file called **Lab4-Task3-SOLUTION.js** – Alternatively, you can combine all of your tasks into one file if you wish.

UPLOADING YOUR SOLUTIONS

- You can upload separate files (if you wish) for each task or a single file into Moodle. It is strongly advised that you simply copy your code from CodeSandBox.io into a notepad (or otherwise) text editor. Save your files as txt files. This is the simplest way to do this. DO NOT USE WORD DOCUMENTS.