**CREATING UI AND IMPLEMENTING VARIOUS COMPONENTS USING REACT**

**TEAM** **MEMBERS**:

ASIF FAROOK.K (211520205016) (**GITHUB** **ID**: <https://github.com/ASIF10504>)

ARSHAT AHAMED RAFIQ A.S (211520205015)(**GITHUB** **ID**: <https://github.com/AGTAHAMED>)

**CREATING UI AND IMPLEMENTING VARIOUS COMPONENTS USING REACT**

$ npx create-react-app count-ui

npx: installed 67 in 7.295s

Creating a new React app in /Users/cherrybomb/count-ui.

Installing packages. This might take a couple of minutes.

Installing react, react-dom, and react-scripts with cra-template...

[...]

Installing template dependencies using npm...

+ @testing-library/jest-dom@5.16.4

+ @testing-library/user-event@13.5.0

+ web-vitals@2.1.4

+ @testing-library/react@13.3.0

added 52 packages from 109 contributors in 9.858s

[...]

Success! Created count-ui at /Users/cherrybomb/count-ui

[...]

We suggest that you begin by typing:

cd count-ui

npm start

As you can see, the npx command has created a new template with a folder structure, an awesome README file, and a Git repository. Here's the structure:

$ cd count-ui/

/Users/cherrybomb/count-ui

$ ls -A -1

.git

.gitignore

README.md

node\_modules

package-lock.json

package.json

public

src

This process also initialized the Git repo and set the branch to master, which is a pretty cool trick. Next, install the npm packages:

$ npm install axios http-proxy-middleware

[...]

npm WARN @apideck/better-ajv-errors@0.3.4 requires a peer of ajv@>=8 but none is installed. You must install peer dependencies yourself.

+ http-proxy-middleware@2.0.6

+ axios@0.27.2

added 2 packages from 2 contributors, updated 1 package and audited 1449 packages in 5.886s

Now that those are set up, add your services, and main.js file:

$ mkdir src/services

src/services

$ touch src/services/main.js

Preconfiguration is now complete, so you can now work on coding.

**Code a UI from start to finish**

Now that you have everything preconfigured, you can put together the service for your application. Add the following code to the main.js file:

import axios from 'axios';

const baseURL = '<http://localhost:5001/api>';

export const get = async () => await axios.get(`${baseURL}/`);

export const increment = async () => await axios.post(`${baseURL}/`);

export default {

get,

increment

}

This process creates a JavaScript file that interacts with the API you created in my previous article.

**Set up the proxy**

Next, you must set up a proxy middleware by creating a new file in the src directory.

$ touch src/setupProxy.js

Configure the proxy with this code in setupProxy.js:

const { createProxyMiddleware } = require('http-proxy-middleware');

module.exports = function(app) {

app.use(

'/api',

createProxyMiddleware({

target: '[http://localhost:5000](http://localhost:5000/)',

changeOrigin: true,

})

);

};

In this code, the app.use function specifies the service to use as /api when connecting to the existing API project. However, nothing defines api in the code. This is where a proxy comes in. With a proxy, you can define the api function on the proxy level to interact with your Express API. This middleware registers requests between both applications because the UI and API use the same host with different ports. They require a proxy to transfer internal traffic.

[**SKIP TO CONTENT**](https://opensource.com/article/22/7/code-first-react-app)

**Programming and development**

[**Red Hat Developers Blog**](https://developers.redhat.com/?intcmp=7016000000127cYAAQ)

[**Programming cheat sheets**](https://opensource.com/downloads/cheat-sheets?intcmp=7016000000127cYAAQ)

[**Try for free: Red Hat Learning Subscription**](https://www.redhat.com/en/services/training/learning-subscription?intcmp=7016000000127cYAAQ)

[**eBook HYPERLINK "https://opensource.com/downloads/bash-programming-guide?intcmp=7016000000127cYAAQ": An introduction to programming with Bash**](https://opensource.com/downloads/bash-programming-guide?intcmp=7016000000127cYAAQ)

[**Bash shell scripting cheat sheet**](https://developers.redhat.com/cheat-sheets/bash-shell-cheat-sheet?intcmp=7016000000127cYAAQ)

[**eBook HYPERLINK "https://developers.redhat.com/e-books/modernizing-enterprise-java?intcmp=7016000000127cYAAQ": Modernizing Enterprise Java**](https://developers.redhat.com/e-books/modernizing-enterprise-java?intcmp=7016000000127cYAAQ)

[**An open source developer's guide to building applications**](https://opensource.com/downloads/building-applications?intcmp=7016000000127cYAAQ)

**JavaScript imports**

In your base src directory, you see that the original template created an App.js, and you must add main.js (in the services directory) to your imports in the App.js file. You also need to import React on the very first line, as it is external to the project:

import React from 'react'

import main from './services/main';

**Add the rendering function**

Now that you have your imports, you must add a render function. In the **App()** function of App.js, add the first section of definitions for **react** and **count** before the **return** section. This section gets the **count** from the API and puts it on the screen. In the **return** function, a button provides the ability to increment the count.

function App() {

const [count, setCount] = React.useState(0);

React.useEffect(()=>{

async function fetchCount(){

const newCount = (await main.get()).data.count;

setCount(newCount);

}

fetchCount();

}, [setCount]);

return (

<div className="App">

<header className="App-header">

<h4>

{count}

</h4>

<button onClick={async ()=>{

setCount((await main.increment()).data.count);

}}>

Increment

</button>

</header>

</div>

);

}

To start and test the app, run npm run start. You should see the output below. Before running the application, confirm your API is running from the Express app by running node ./src/index.js.

$ npm run start

> count-ui@0.1.0 start /Users/cherrybomb/count-ui

> react-scripts start

[HPM] Proxy created: / -> [http://localhost:5000](http://localhost:5000/)

(node:71729) [DEP\_WEBPACK\_DEV\_SERVER\_ON\_AFTER\_SETUP\_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.

(Use `node --trace-deprecation ...` to show where the warning was created)

(node:71729) [DEP\_WEBPACK\_DEV\_SERVER\_ON\_BEFORE\_SETUP\_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.

Starting the development server...