App.js | ViewAssets.js M | db.json | Update.js M

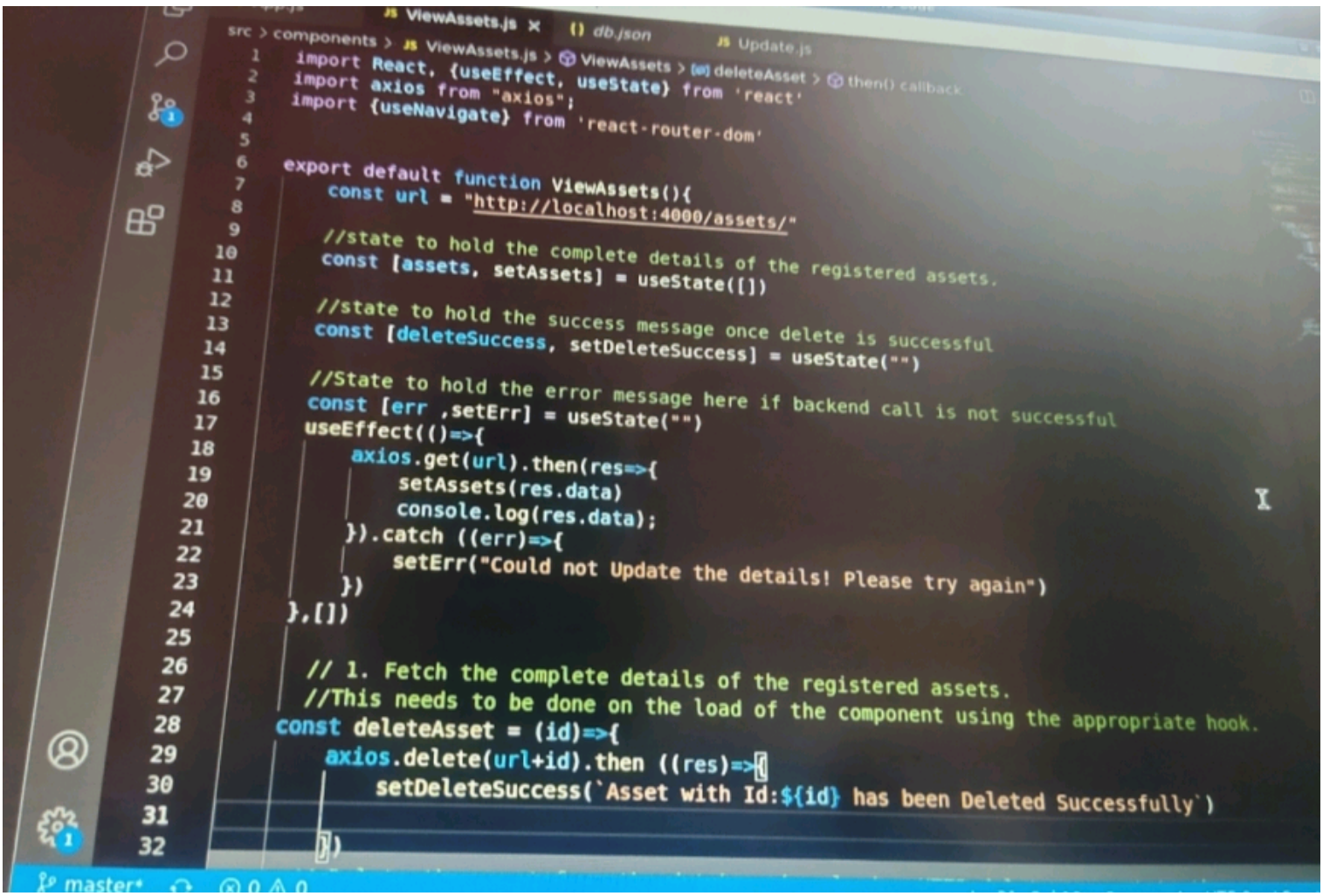
src > App.js > App

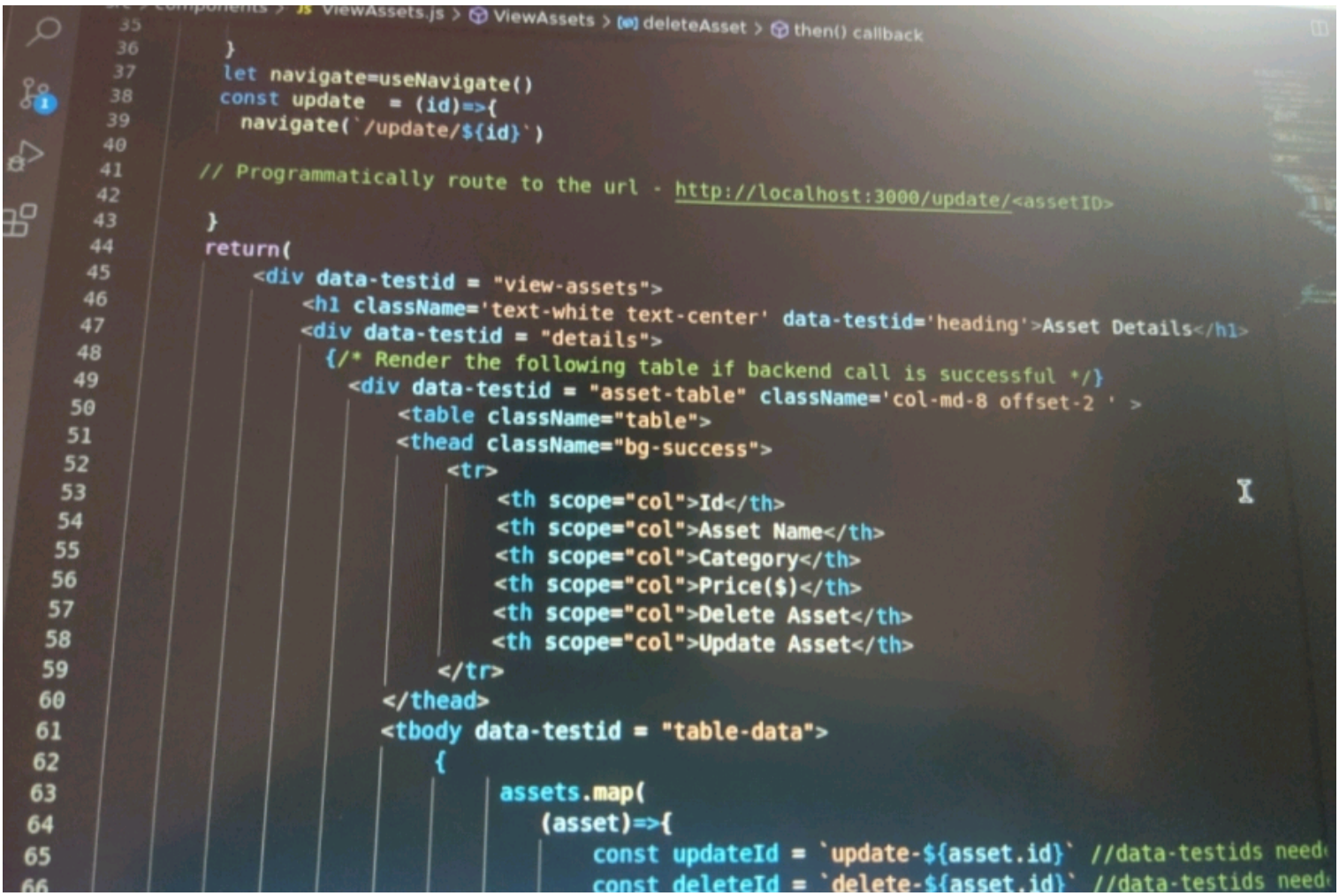
```
import React from 'react'
import {Routes, BrowserRouter, Link, Route} from 'react-router-dom'
// Import components
import Update from './components/Update'
import ViewAssets from './components/ViewAssets'

export default function App(){
    return(
      <BrowserRouter>
        <div className={'bg'}>
        <nav className="navbar navbar-expand-lg navbar-light  bg-custom">
          <span className="navbar-brand">Quebes</span>
          <ul className="navbar-nav">
            <li className="nav-item" data-testid = "view-asset-link" >
              {/* Provide link to 'view-assets' */}
              <Link to="/view-asset">View Assets</Link>
            </li>
          </ul>
        </nav>
        <Routes>
          <Route path="/" element={<ViewAssets/>}/>
          <Route path="/view-asset" element={<ViewAssets/>}/>
          <Route path="/update/:id" element={<Update/>}/>

         {/* Implement routing */}
        </Routes>


      </div>
      </BrowserRouter>
    )
}
```

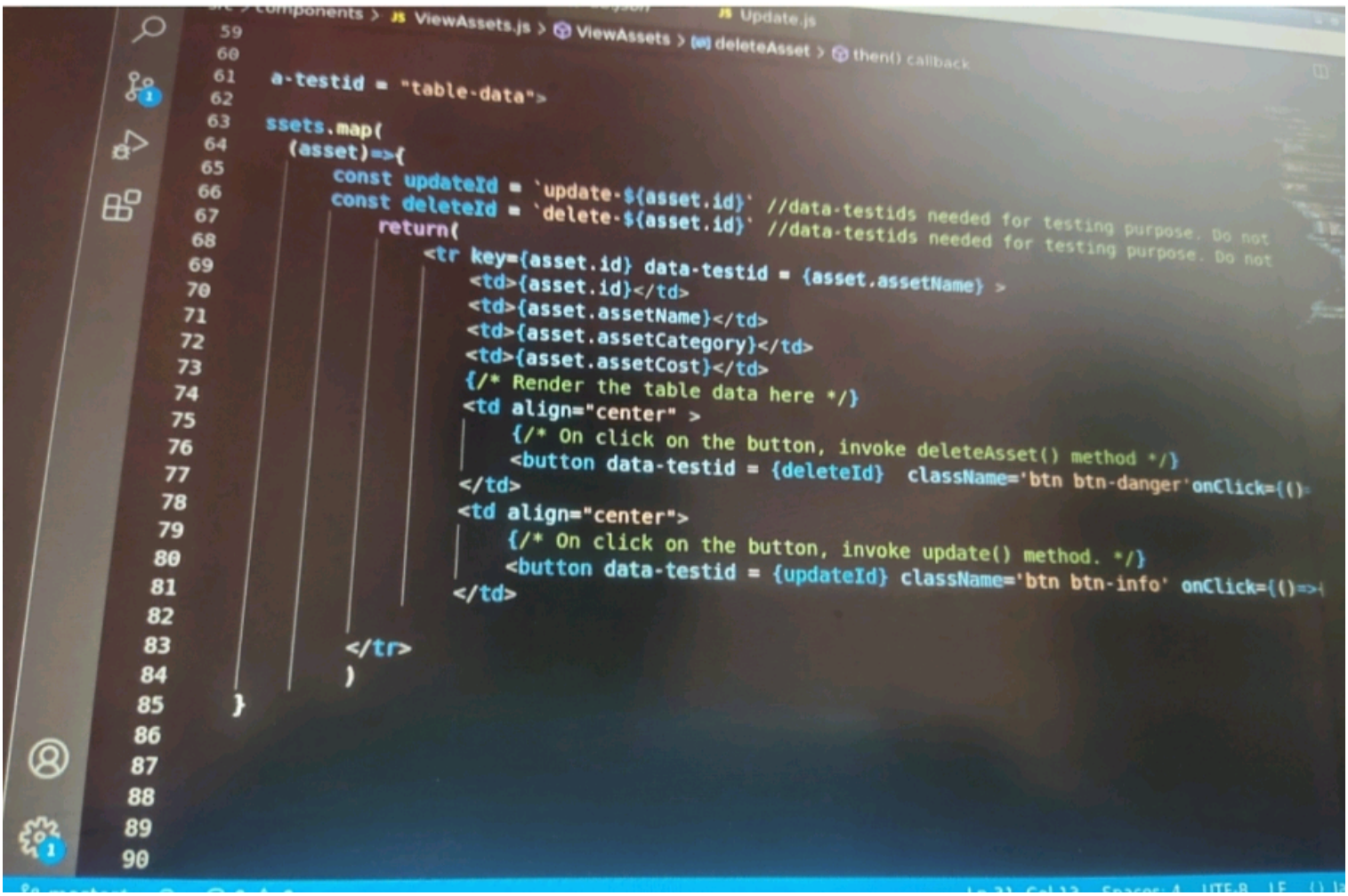

ViewAssets.js
db.json
Update.js
src > components > ViewAssets.js > ViewAssets > deleteAsset > then() callback
import React, {useEffect, useState} from 'react'
import axios from "axios";
import {useNavigate} from 'react-router-dom'
export default function ViewAssets(){
const url = "http://localhost:4000/assets/"
//state to hold the complete details of the registered assets.
const [assets, setAssets] = useState([])
//state to hold the success message once delete is successful
const [deleteSuccess, setDeleteSuccess] = useState("")
//State to hold the error message here if backend call is not successful
const [err ,setErr] = useState("")
useEffect(()=>{
axios.get(url).then(res=>{
setAssets(res.data)
console.log(res.data);
}).catch ((err)=>{
setErr("Could not Update the details! Please try again")
})
},[])
// 1. Fetch the complete details of the registered assets.
//This needs to be done on the load of the component using the appropriate hook.
const deleteAsset = (id)=>{
axios.delete(url+id).then ((res)=>{
setDeleteSuccess(`Asset with Id:${id} has been Deleted Successfully`)
})
master*

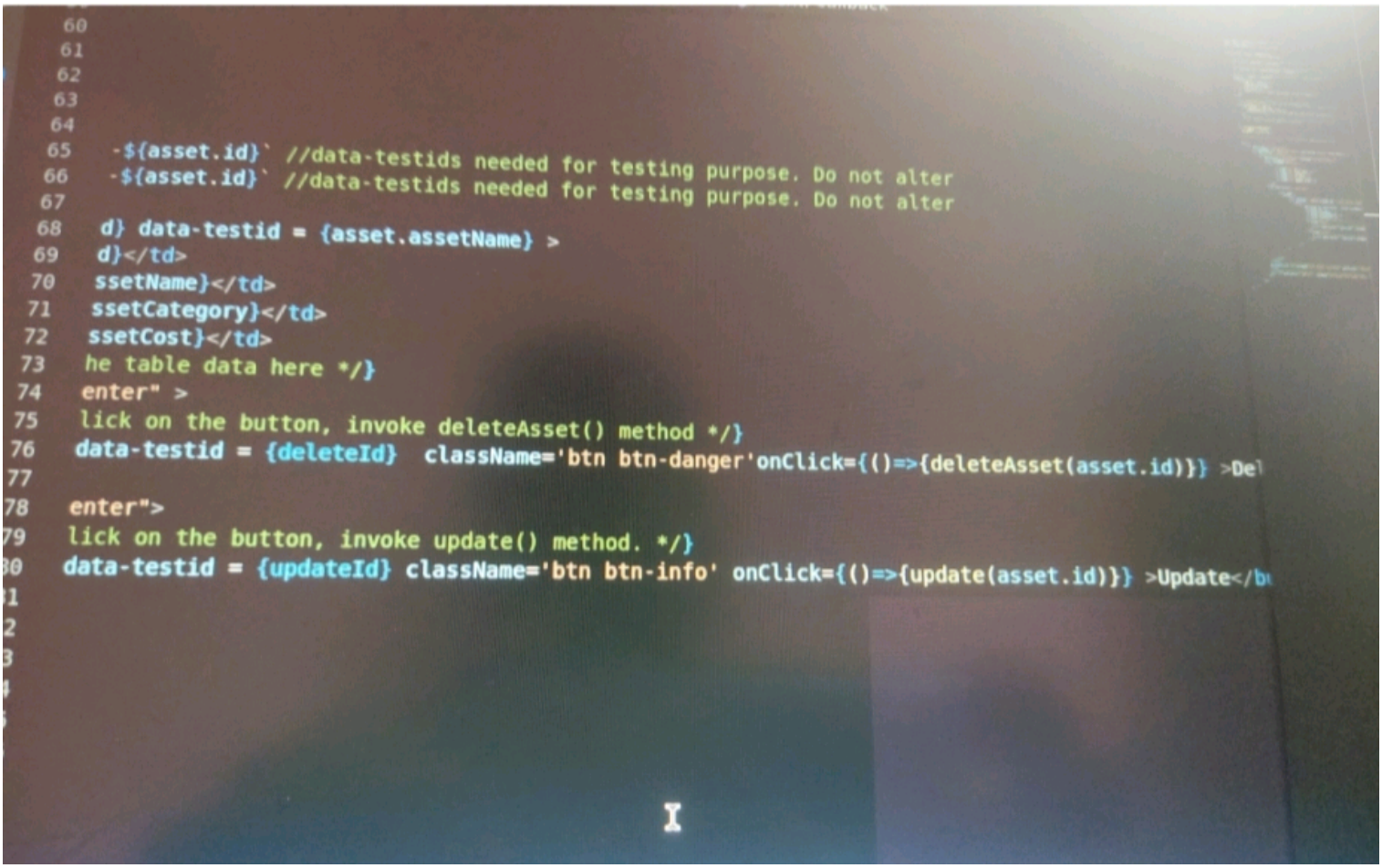

ViewAssets.js > ViewAssets > deleteAsset > then() callback
}
let navigate=useNavigate()
const update = (id)=>{
navigate(`/update/${id}`)
// Programmatically route to the url - http://localhost:3000/update/<assetID>
}
return(

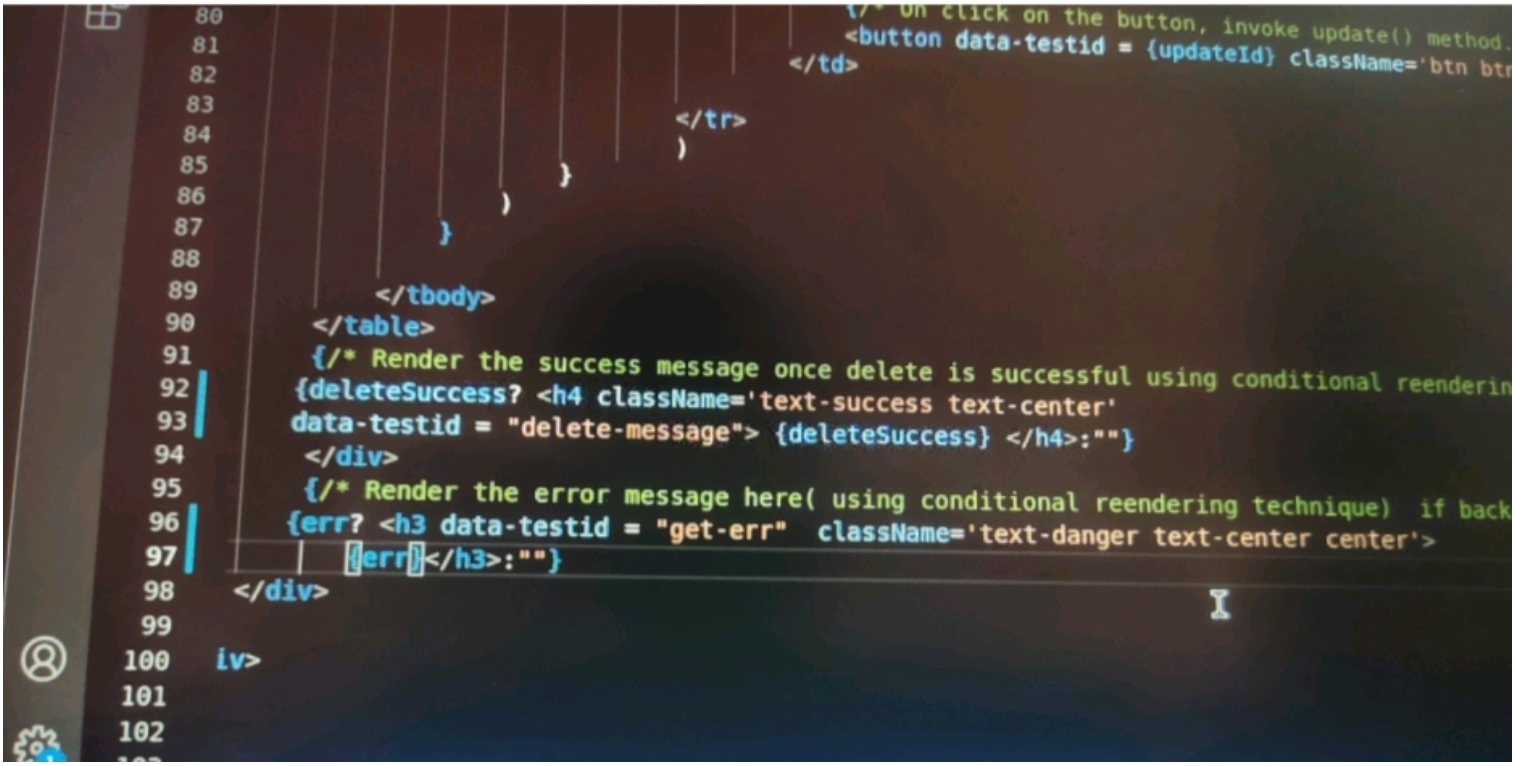
<h1 className='text-white text-center' data-testid='heading'>Asset Details</h1>

{/* Render the following table if backend call is successful */}

<table className="table">
<thead className="bg-success">
<tr>
<th scope="col">Id</th>
<th scope="col">Asset Name</th>
<th scope="col">Category</th>
<th scope="col">Price($)</th>
<th scope="col">Delete Asset</th>
<th scope="col">Update Asset</th>
</tr>
</thead>
<tbody data-testid = "table-data">
{
assets.map(
(asset)=>{
const updateId = `update-${asset.id}` //data-testids need
const deleteId = `delete-${asset.id}` //data-testids need

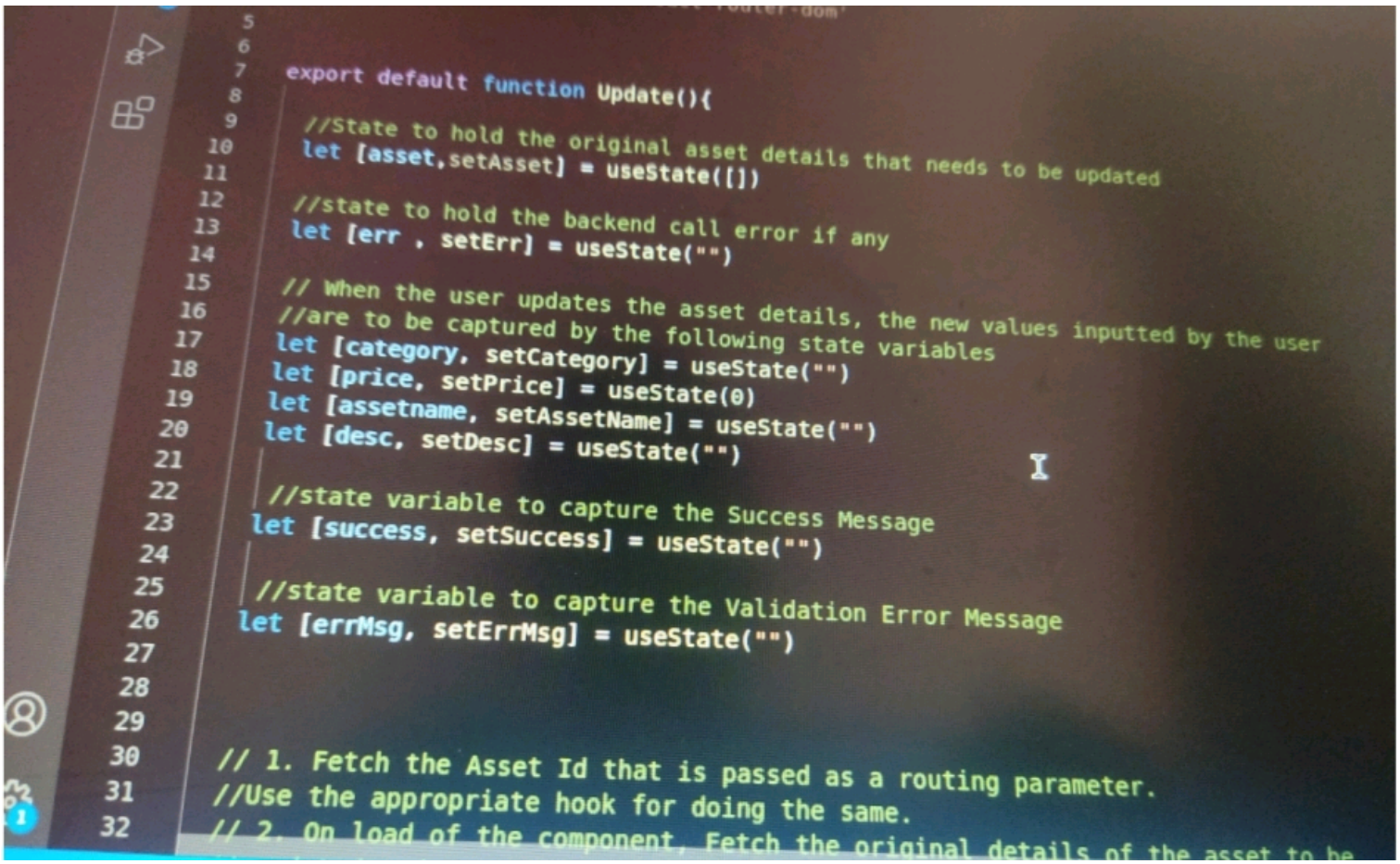

components > ViewAssets.js > ViewAssets > deleteAsset > then() callback
Update.js
a-testid = "table-data">
ssets.map(
(asset)=>{
const updateId = `update-${asset.id}` //data-testids needed for testing purpose. Do not
const deleteId = `delete-${asset.id}` //data-testids needed for testing purpose. Do not
return(
<tr key={asset.id} data-testid = {asset.assetName} >
<td>{asset.id}</td>
<td>{asset.assetName}</td>
<td>{asset.assetCategory}</td>
<td>{asset.assetCost}</td>
{/* Render the table data here */}
<td align="center" >
{/* On click on the button, invoke deleteAsset() method */}
<button data-testid = {deleteId} className='btn btn-danger'onClick={()=
</td>
<td align="center">
{/* On click on the button, invoke update() method. */}
<button data-testid = {updateId} className='btn btn-info' onClick={()=>{
</td>
</tr>
)
}


-${asset.id}` //data-testids needed for testing purpose. Do not alter
-${asset.id}` //data-testids needed for testing purpose. Do not alter
d} data-testid = {asset.assetName} >
d}</td>
ssetName}</td>
ssetCategory}</td>
ssetCost}</td>
he table data here */}
enter" >
lick on the button, invoke deleteAsset() method */}
data-testid = {deleteId} className='btn btn-danger'onClick={()=>{deleteAsset(asset.id)}} >De
enter">
lick on the button, invoke update() method. */}
data-testid = {updateId} className='btn btn-info' onClick={()=>{update(asset.id)}} >Update</b

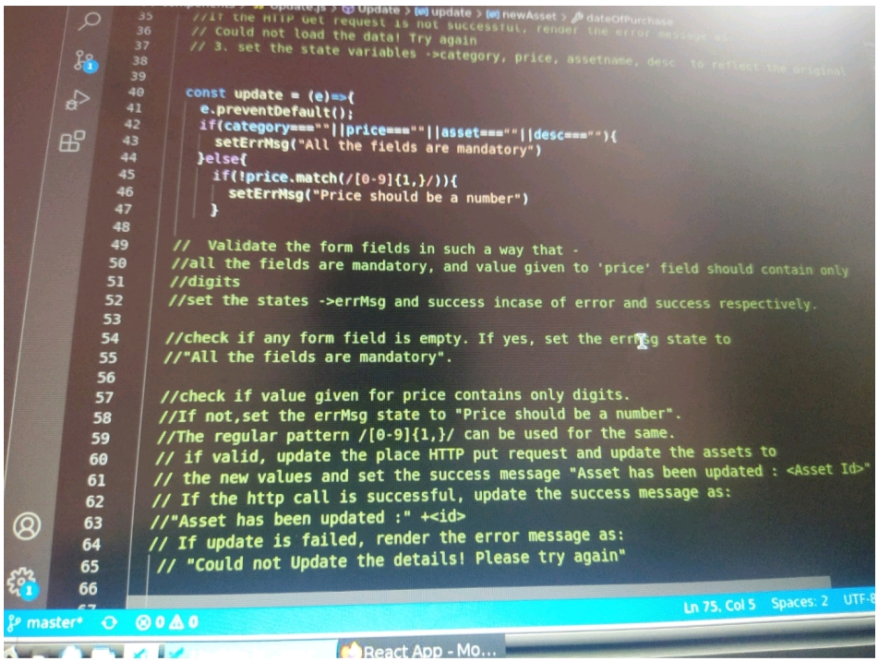

{/* On click on the button, invoke update() method
<button data-testid = {updateId} classNam
</td>
</tr>
)
}
)
}
</tbody>
</table>
{/* Render the success message once delete is successful using conditional reenderin
{deleteSuccess? <h4 className='text-success text-center'
data-testid = "delete-message"> {deleteSuccess} </h4>:""}

{/* Render the error message here(using conditional reendering technique) if back
{err? <h3 data-testid = "get-err" className='text-danger text-center center'>
{err}</h3>:""}

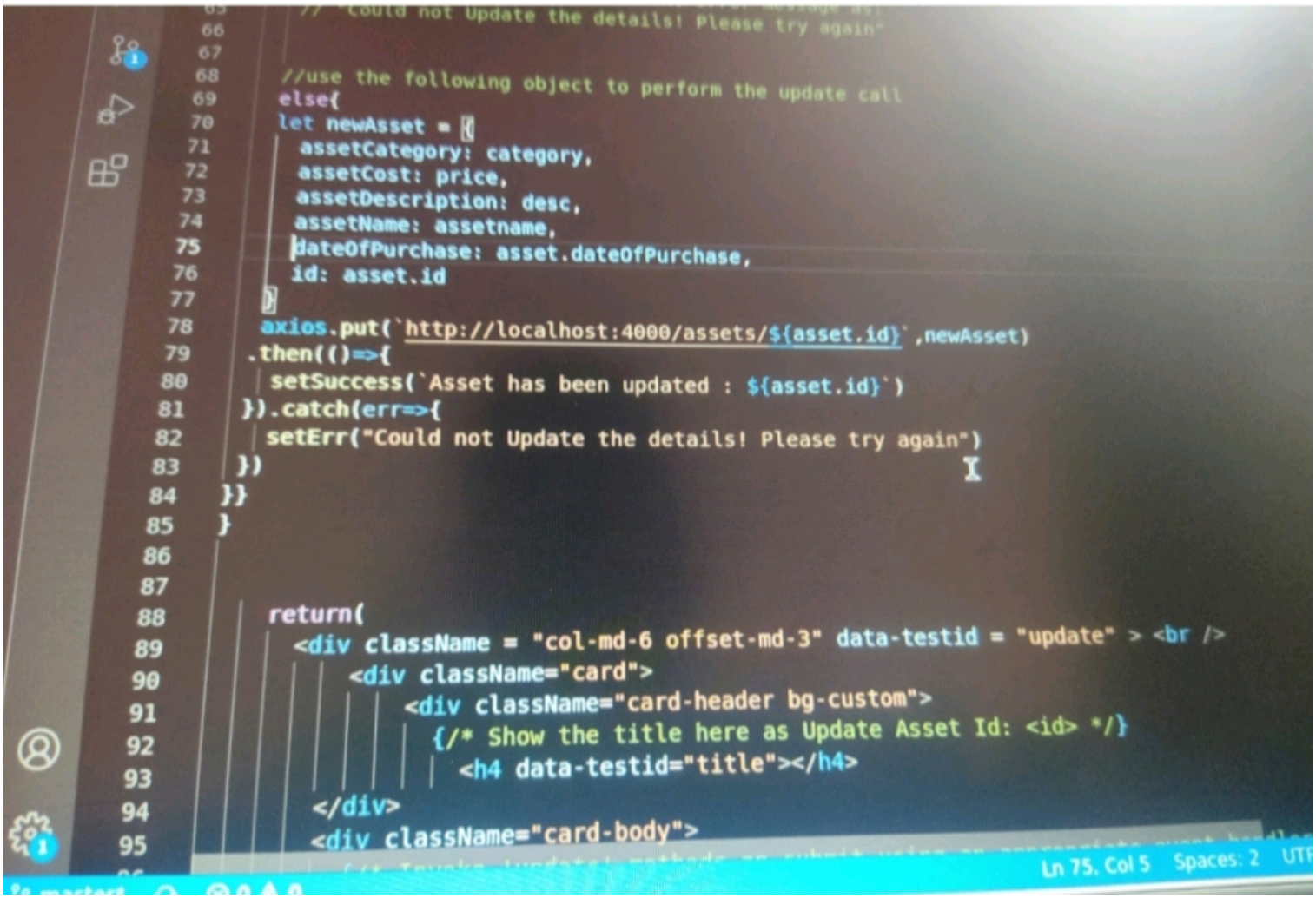
iv>

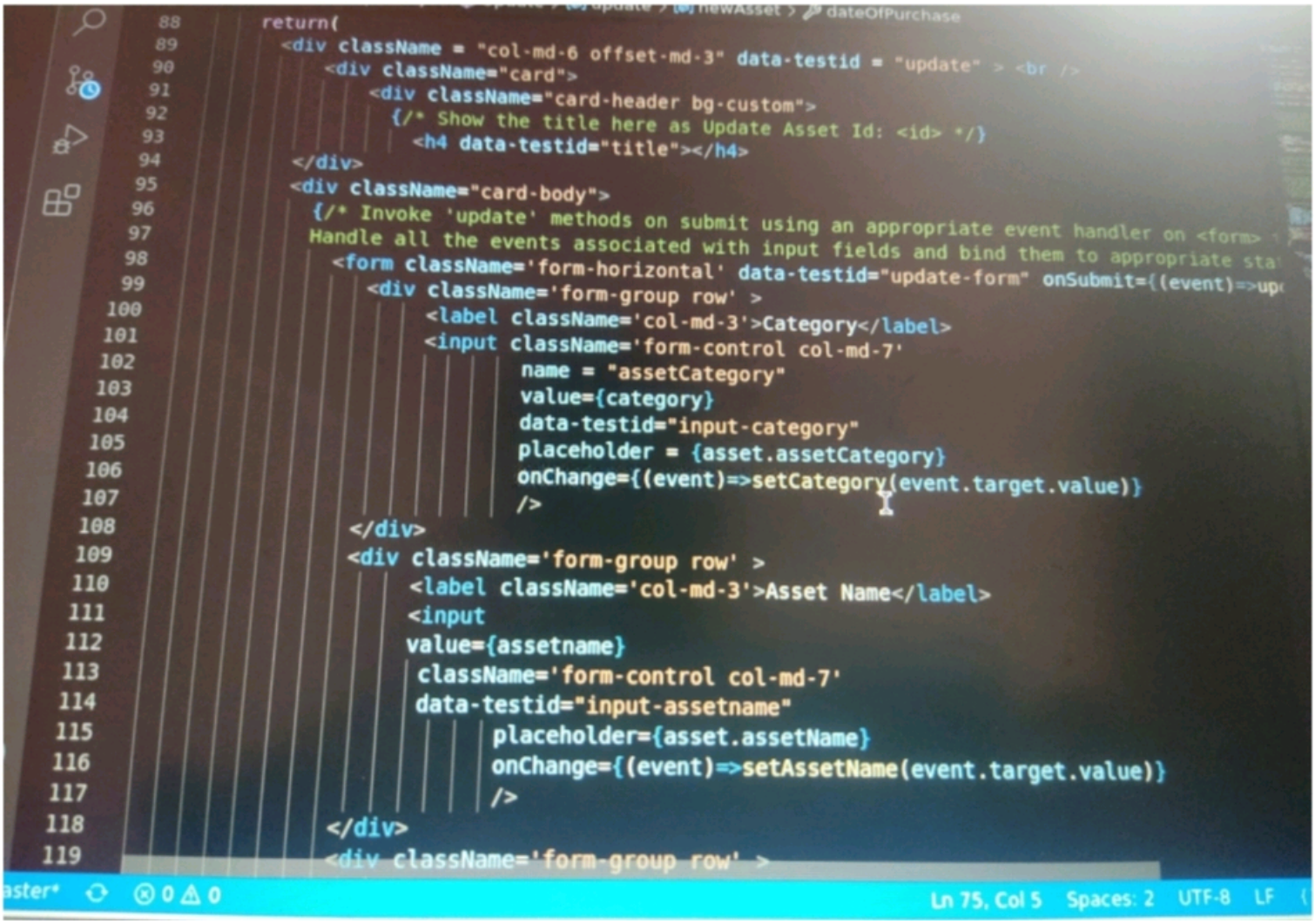


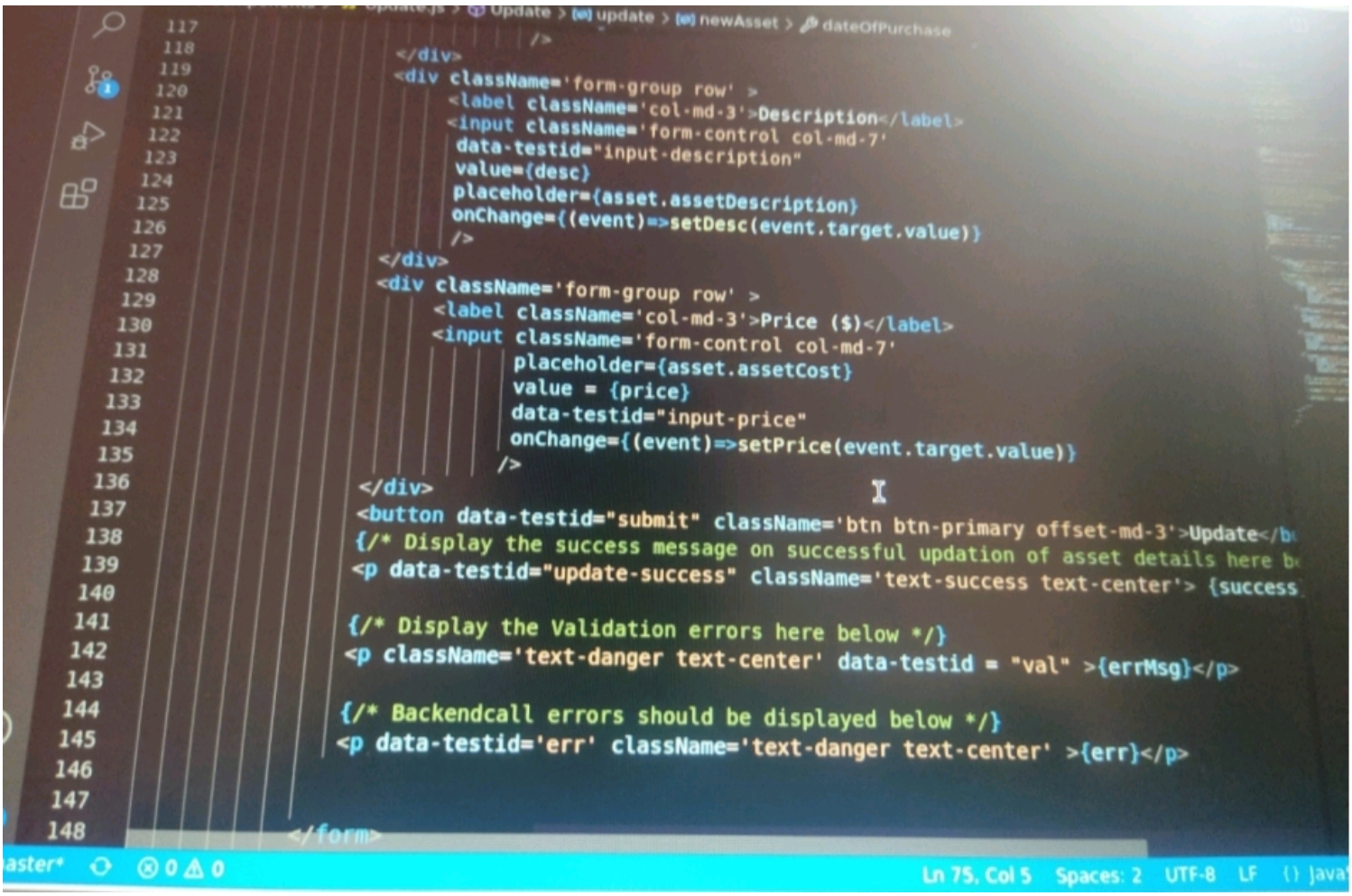
```
5
6
7   export default function Update(){
8
9     //State to hold the original asset details that needs to be updated
10    let [asset,setAsset] = useState([])
11
12    //state to hold the backend call error if any
13    let [err , setErr] = useState("")
14
15    // When the user updates the asset details, the new values inputted by the user
16    //are to be captured by the following state variables
17    let [category, setCategory] = useState("")
18    let [price, setPrice] = useState(0)
19    let [assetname, setAssetName] = useState("")
20    let [desc, setDesc] = useState("")
21
22    //state variable to capture the Success Message
23    let [success, setSuccess] = useState("")
24
25    //state variable to capture the Validation Error Message
26    let [errMsg, setErrMsg] = useState("")
27
28
29
30  // 1. Fetch the Asset Id that is passed as a routing parameter.
31  //Use the appropriate hook for doing the same.
32  // 2. On load of the component, Fetch the original details of the asset to be
```


//if the HTTP GET request is not successful, render the error message as
// Could not load the data! Try again
// 3. set the state variables ->category, price, assetname, desc to reflect the original
const update = (e)=>{
e.preventDefault();
if(category===""||price===""||asset===""||desc===""){
setErrMsg("All the fields are mandatory")
}else{
if(!price.match(/[0-9]{1,}/)){
setErrMsg("Price should be a number")
}
// Validate the form fields in such a way that -
//all the fields are mandatory, and value given to 'price' field should contain only
//digits
//set the states ->errMsg and success incase of error and success respectively.
//check if any form field is empty. If yes, set the errMsg state to
//"All the fields are mandatory".
//check if value given for price contains only digits.
//If not,set the errMsg state to "Price should be a number".
//The regular pattern /[0-9]{1,}/ can be used for the same.
// if valid, update the place HTTP put request and update the assets to
// the new values and set the success message "Asset has been updated : <Asset Id>"
// If the http call is successful, update the success message as:
//"Asset has been updated :" +<id>
// If update is failed, render the error message as:
// "Could not Update the details! Please try again"
master*
Ln 75, Col 5
Spaces: 2
React App - Mo...


// "Could not Update the details! Please try again"
//use the following object to perform the update


66 // "Could not Update the details! Please try again"
67
68 //use the following object to perform the update call
69 else{
70 let newAsset = {
71 assetCategory: category,
72 assetCost: price,
73 assetDescription: desc,
74 assetName: assetname,
75 dateOfPurchase: asset.dateOfPurchase,
76 id: asset.id
77 }
78 axios.put(`http://localhost:4000/assets/${asset.id}`,newAsset)
79 .then(()=>{
80 setSuccess(`Asset has been updated : ${asset.id}`)
81 }).catch(err=>{
82 setErr("Could not Update the details! Please try again")
83 })
84 }}
85 }
86
87
88 return(
89
90
91
92 {/* Show the title here as Update Asset Id: <id> */}
93 <h4 data-testid="title"></h4>
94
95
Ln 75, Col 5
Spaces: 2
UTF

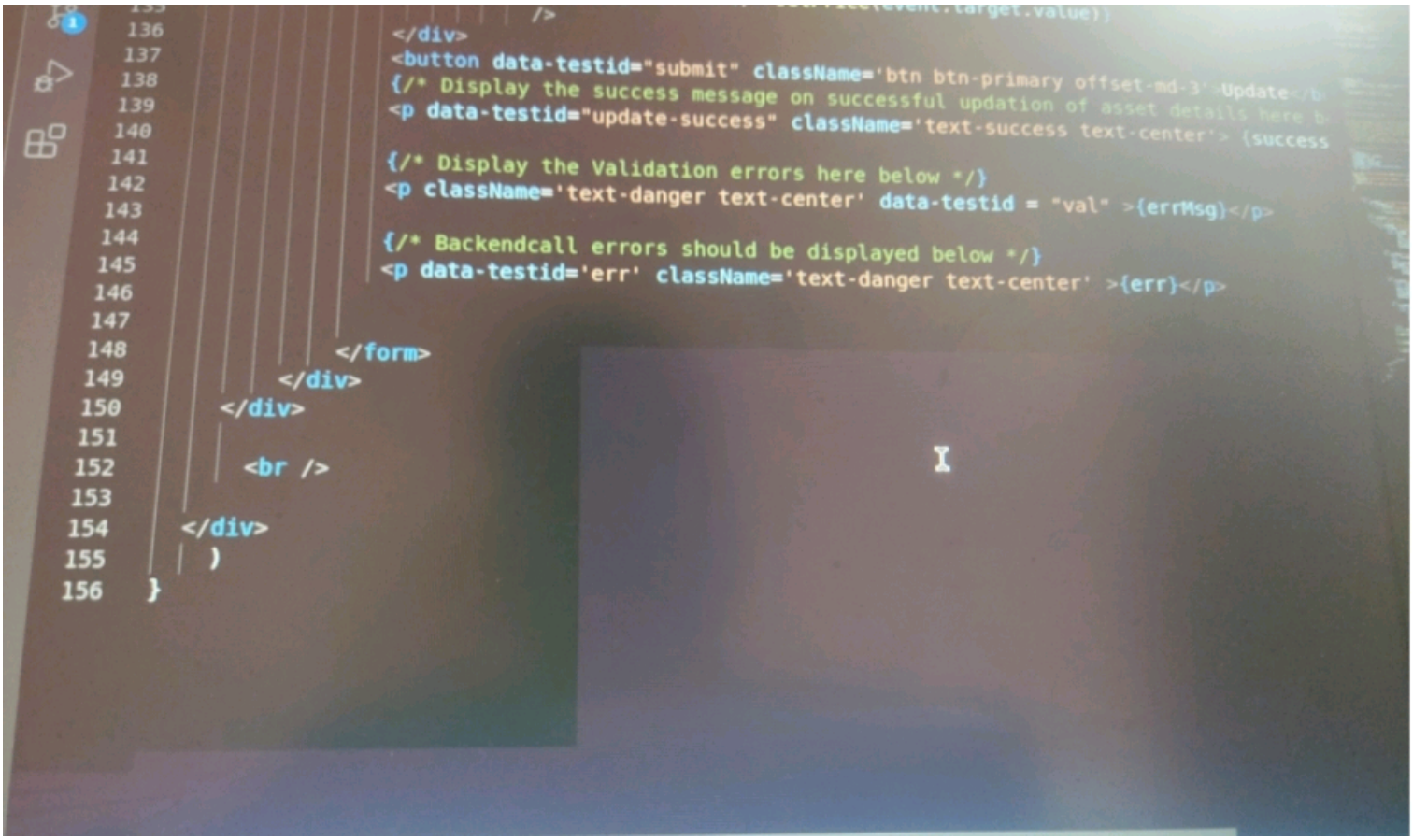

update > newAsset > dateOfPurchase
88 return(
89
90
91
92 {/* Show the title here as Update Asset Id: <id> */}
93 <h4 data-testid="title"></h4>
94
95
96 {/* Invoke 'update' methods on submit using an appropriate event handler on <form>
97 Handle all the events associated with input fields and bind them to appropriate sta
98 <form className='form-horizontal' data-testid="update-form" onSubmit={(event)=>up
99
100 <label className='col-md-3'>Category</label>
101 <input className='form-control col-md-7'
102 name = "assetCategory"
103 value={category}
104 data-testid="input-category"
105 placeholder = {asset.assetCategory}
106 onChange={(event)=>setCategory(event.target.value)}
107 />
108
109
110 <label className='col-md-3'>Asset Name</label>
111 <input
112 value={assetname}
113 className='form-control col-md-7'
114 data-testid="input-assetname"
115 placeholder={asset.assetName}
116 onChange={(event)=>setAssetName(event.target.value)}
117 />
118
119
aster* 0 0
Ln 75, Col 5 Spaces: 2 UTF-8 LF


Update > update > newAsset > dateOfPurchase
/>

<label className='col-md-3'>Description</label>
<input className='form-control col-md-7'
data-testid="input-description"
value={desc}
placeholder={asset.assetDescription}
onChange={(event)=>setDesc(event.target.value)}
/>

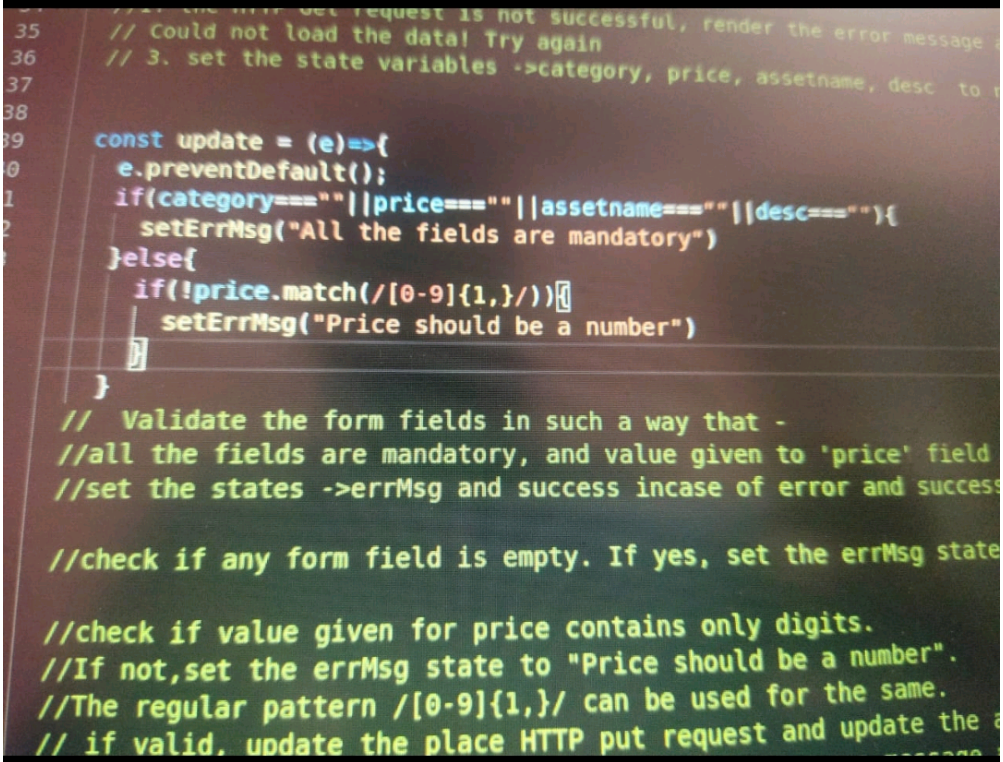
<label className='col-md-3'>Price ($)</label>
<input className='form-control col-md-7'
placeholder={asset.assetCost}
value = {price}
data-testid="input-price"
onChange={(event)=>setPrice(event.target.value)}
/>

<button data-testid="submit" className='btn btn-primary offset-md-3'>Update</b
{/* Display the success message on successful updation of asset details here b
 {success
{/* Display the Validation errors here below */}
{errMsg}
{/* Backendcall errors should be displayed below */}
{err}
</form>
aster*
0 0
Ln 75, Col 5
Spaces: 2
UTF-8
LF
{} Java


/>

<button data-testid="submit" className='btn btn-primary offset-md-3'>Update</b
{/* Display the success message on successful updation of asset details here b
 {success
{/* Display the Validation errors here below */}
{errMsg}
{/* Backendcall errors should be displayed below */}
{err}
</form>

)
}



```
35    // Could not load the data! Try again
36    // 3. set the state variables ->category, price, assetname, desc  to
37
38
39    const update = (e)=>{
0      e.preventDefault();
1      if(category===""||price===""||assetname===""||desc===""){
2        setErrMsg("All the fields are mandatory")
       }else{
         if(!price.match(/[0-9]{1,}/)){
           setErrMsg("Price should be a number")
         }
       }
     //  Validate the form fields in such a way that -
     //all the fields are mandatory, and value given to 'price' field
     //set the states ->errMsg and success incase of error and success

     //check if any form field is empty. If yes, set the errMsg state

     //check if value given for price contains only digits.
     //If not,set the errMsg state to "Price should be a number".
     //The regular pattern /[0-9]{1,}/ can be used for the same.
     // if valid, update the place HTTP put request and update the
```