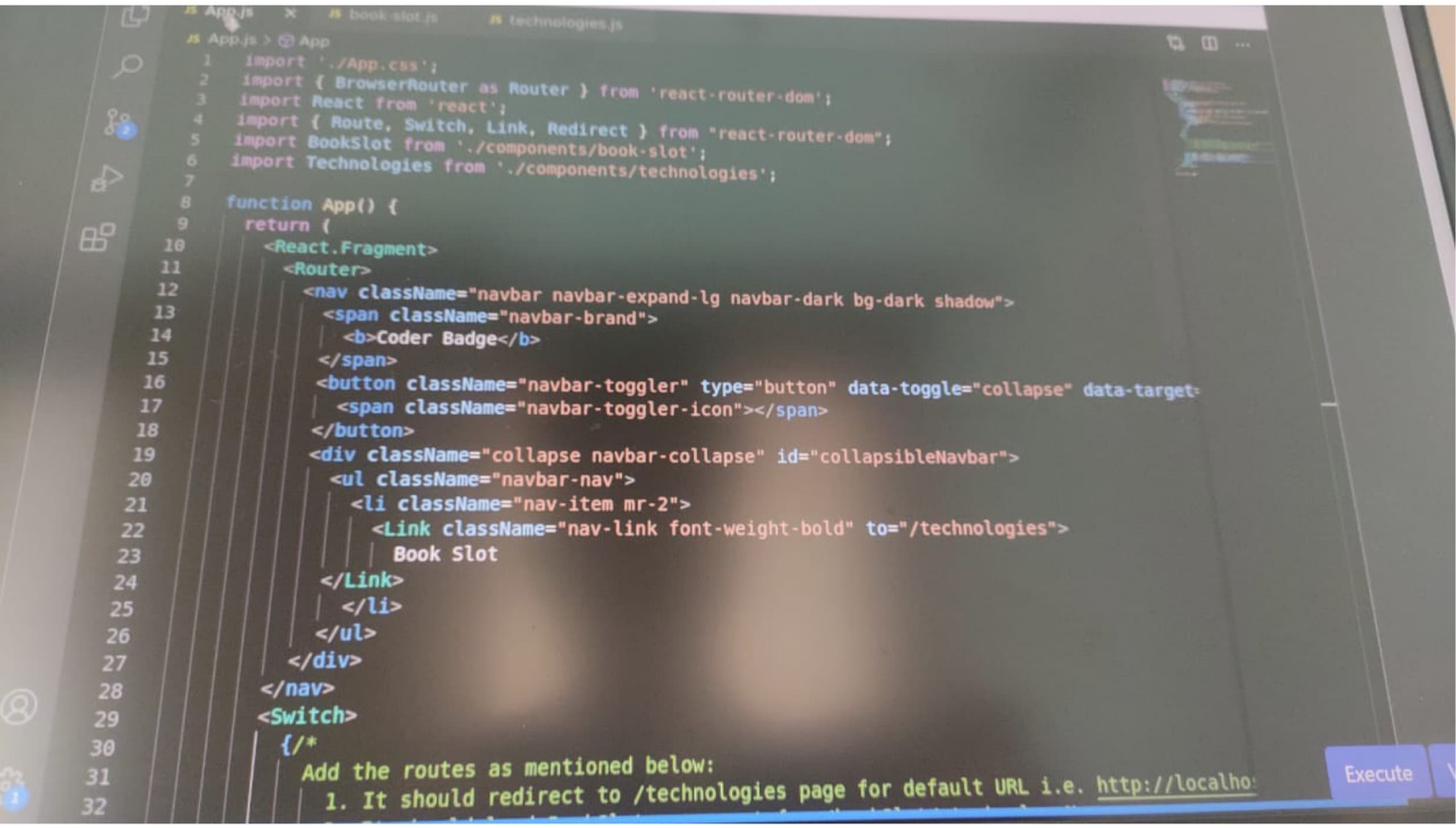

App.js
book-slot.js
technologies.js
App.js > App
import './App.css';
import { BrowserRouter as Router } from 'react-router-dom';
import React from 'react';
import { Route, Switch, Link, Redirect } from "react-router-dom";
import BookSlot from './components/book-slot';
import Technologies from './components/technologies';
function App() {
return (
<React.Fragment>
<Router>
<nav className="navbar navbar-expand-lg navbar-dark bg-dark shadow">

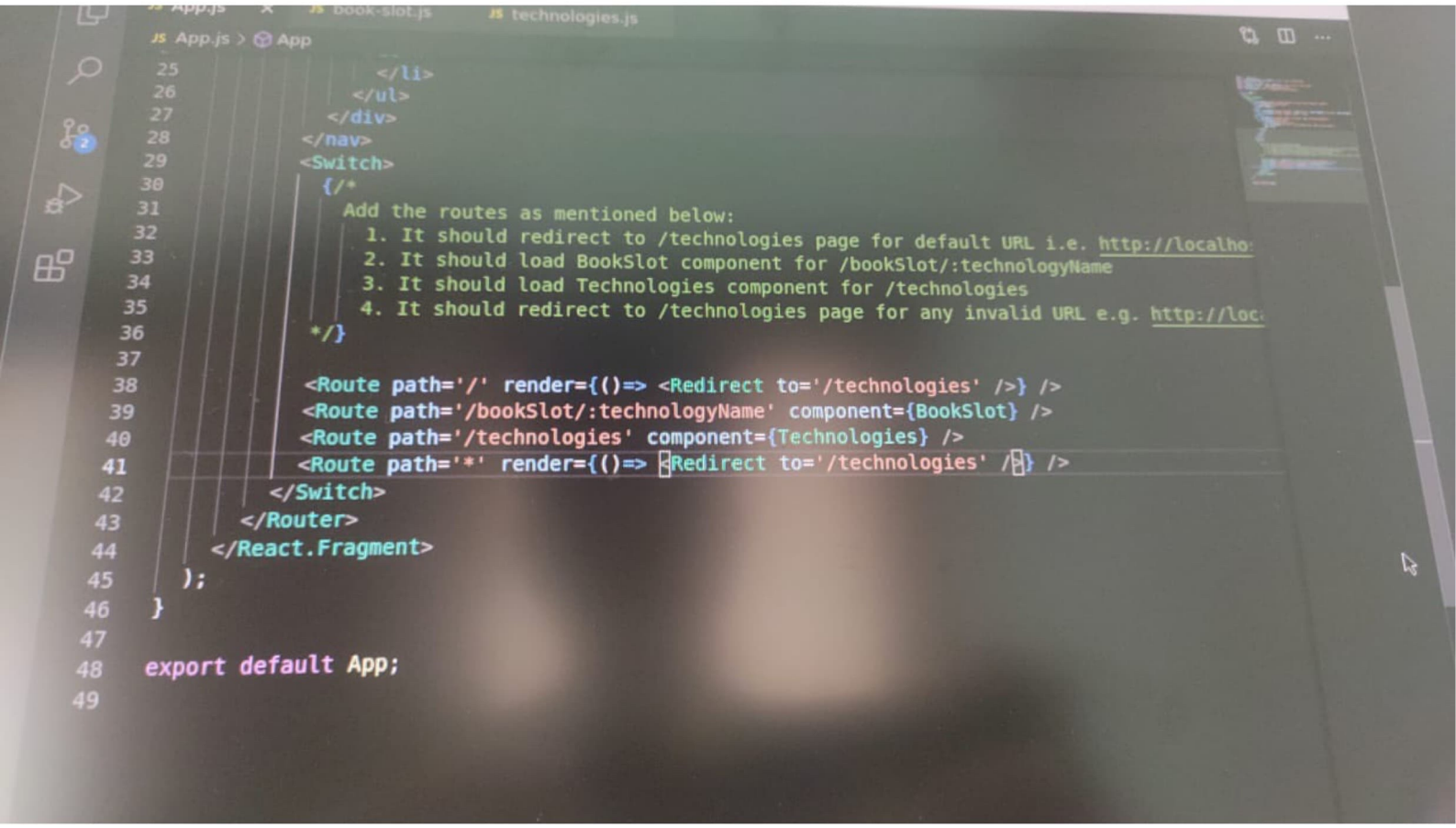
Coder Badge

<button className="navbar-toggler" type="button" data-toggle="collapse" data-target=

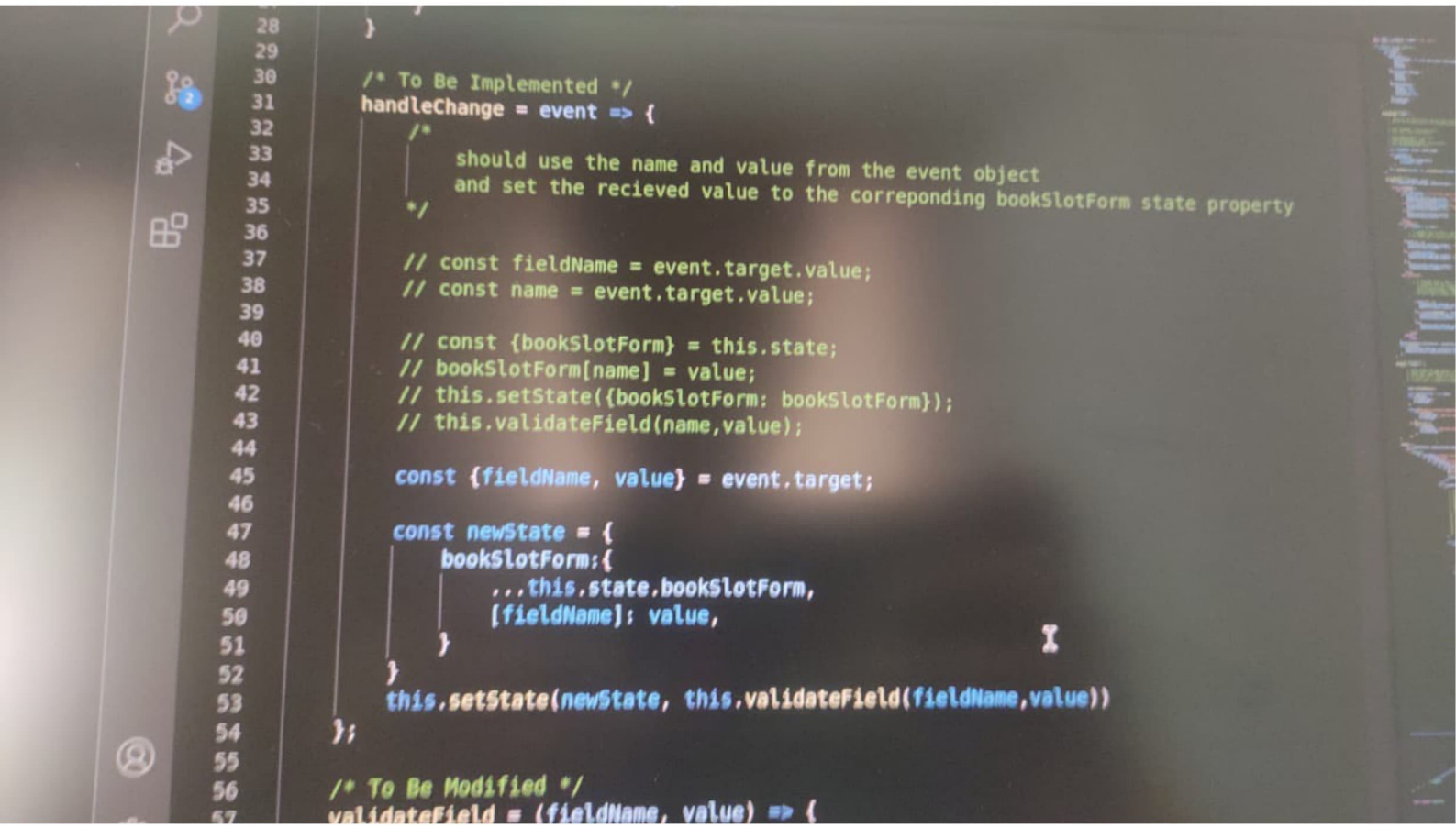
</button>

<ul className="navbar-nav">
<li className="nav-item mr-2">
<Link className="nav-link font-weight-bold" to="/technologies">
Book Slot
</Link>

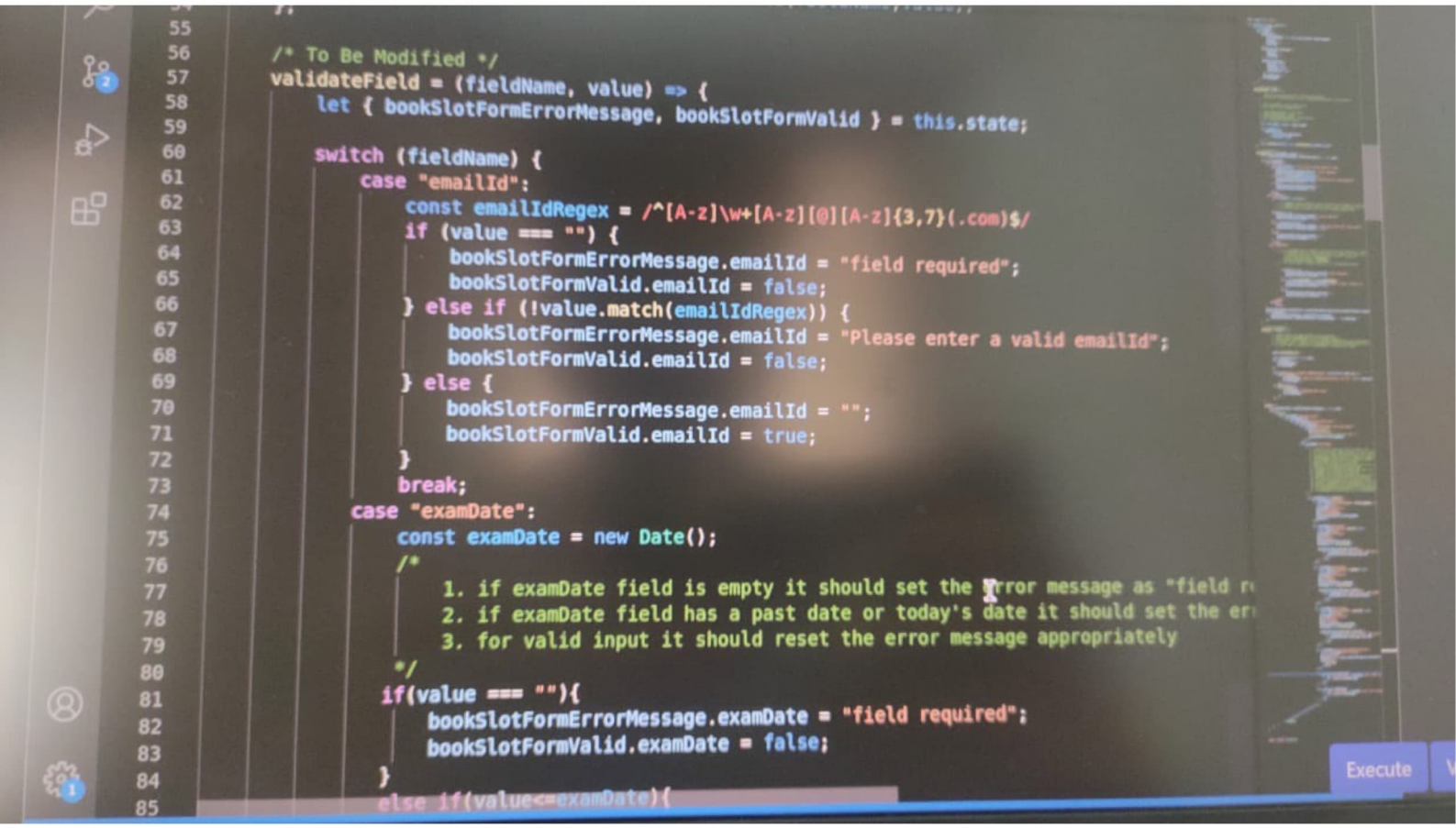
</nav>
<Switch>
{/*
Add the routes as mentioned below:
1. It should redirect to /technologies page for default URL i.e. http://localho
Execute


App.js
book-slot.js
technologies.js
App.js > App

</nav>
<Switch>
{/*
Add the routes as mentioned below:
1. It should redirect to /technologies page for default URL i.e. http://localho
2. It should load BookSlot component for /bookSlot/:technologyName
3. It should load Technologies component for /technologies
4. It should redirect to /technologies page for any invalid URL e.g. http://loc
*/}
<Route path='/' render={()=> <Redirect to='/technologies' />} />
<Route path='/bookSlot/:technologyName' component={BookSlot} />
<Route path='/technologies' component={Technologies} />
<Route path='*' render={()=> <Redirect to='/technologies' />} />
</Switch>
</Router>
</React.Fragment>
);
}
export default App;

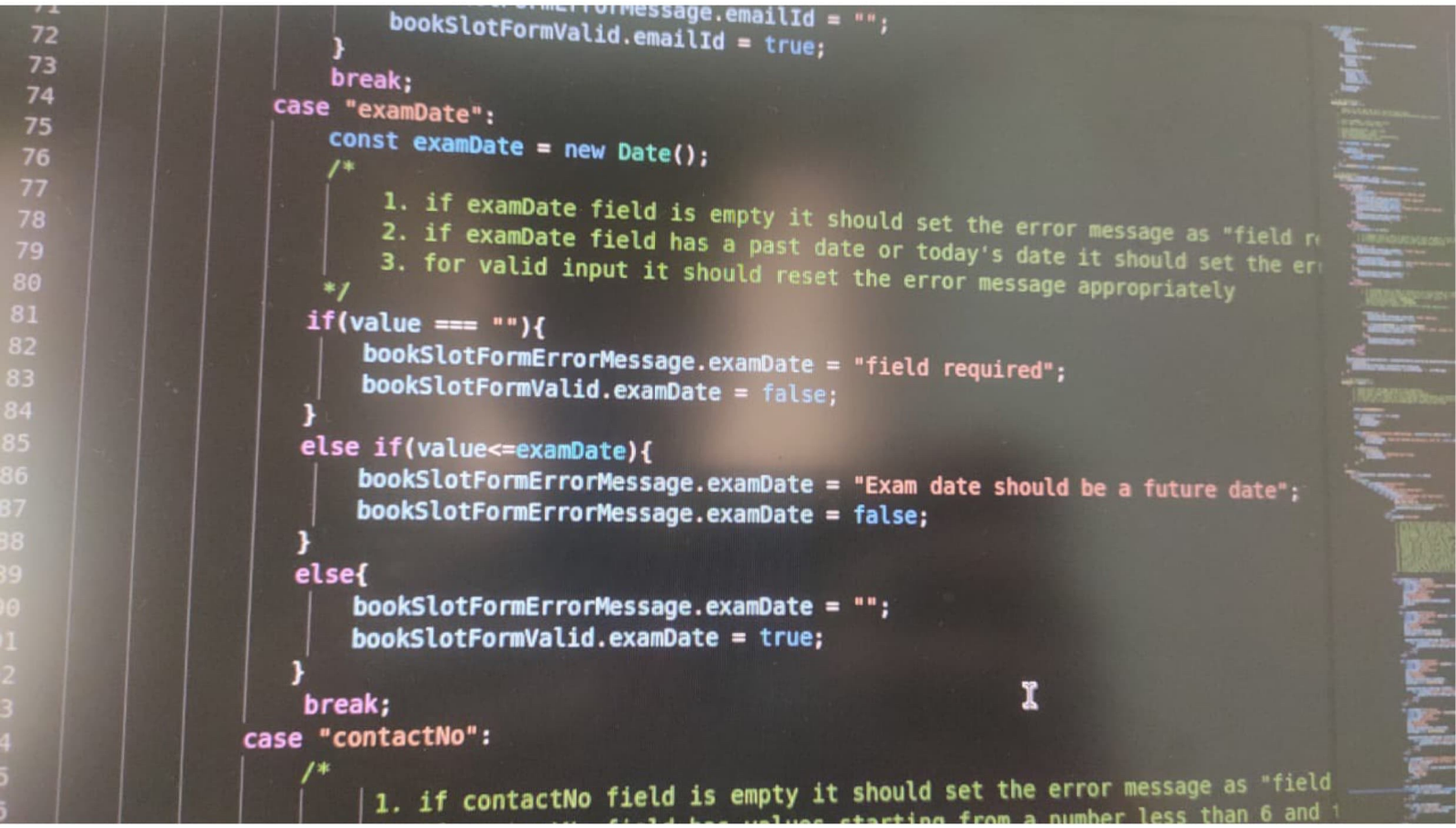


```
28      }
29
30      /* To Be Implemented */
31      handleChange = event => {
32          /*
33              should use the name and value from the event object
34              and set the recieved value to the correponding bookSlotForm state property
35          */
36
37          // const fieldName = event.target.value;
38          // const name = event.target.value;
39
40          // const {bookSlotForm} = this.state;
41          // bookSlotForm[name] = value;
42          // this.setState({bookSlotForm: bookSlotForm});
43          // this.validateField(name,value);
44
45          const {fieldName, value} = event.target;
46
47          const newState = {
48              bookSlotForm:{
49                  ...this.state.bookSlotForm,
50                  [fieldName]: value,
51              }
52          }
53          this.setState(newState, this.validateField(fieldName,value))
54      };
55
56      /* To Be Modified */
57      validateField = (fieldName, value) => {
```

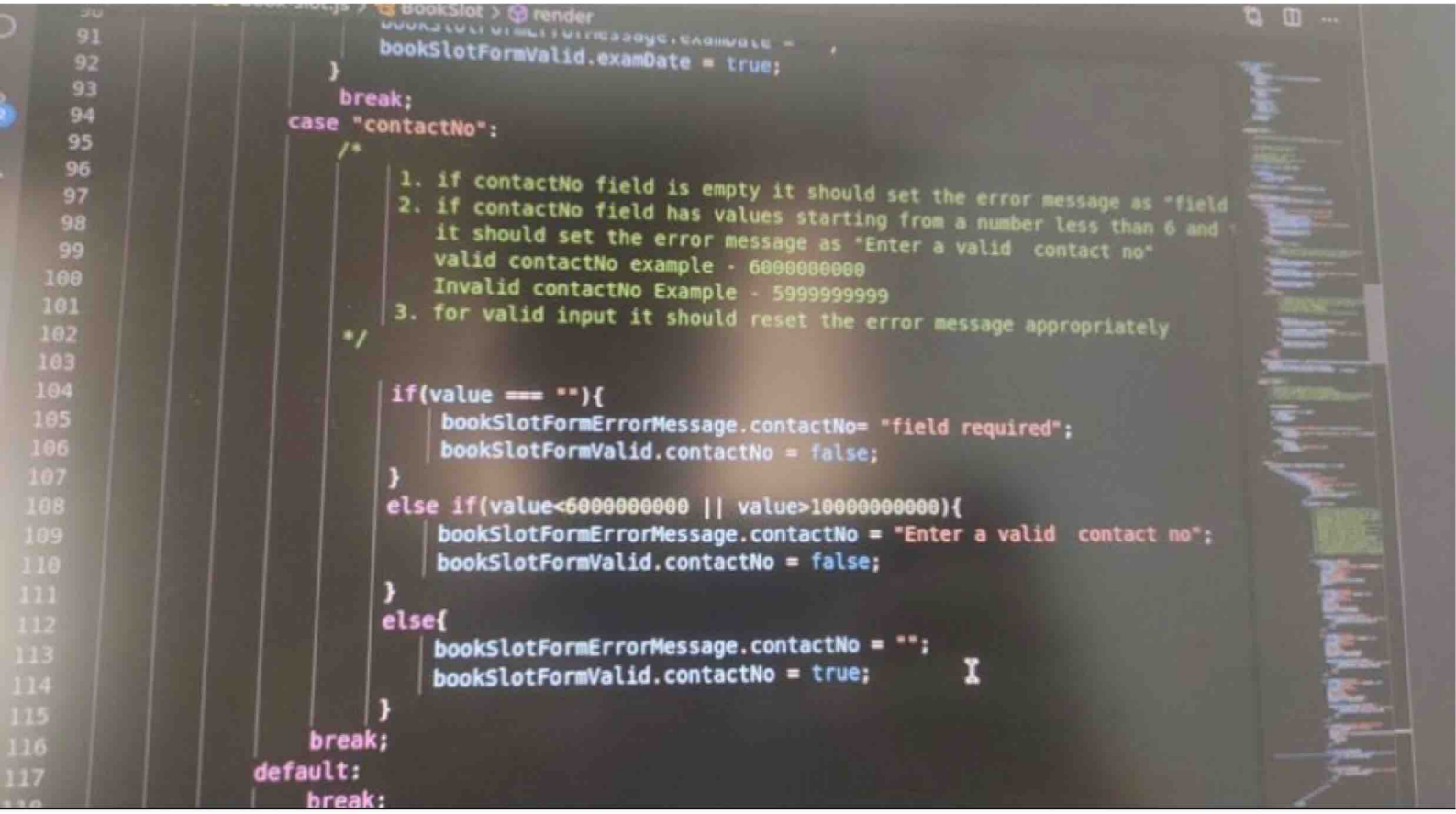


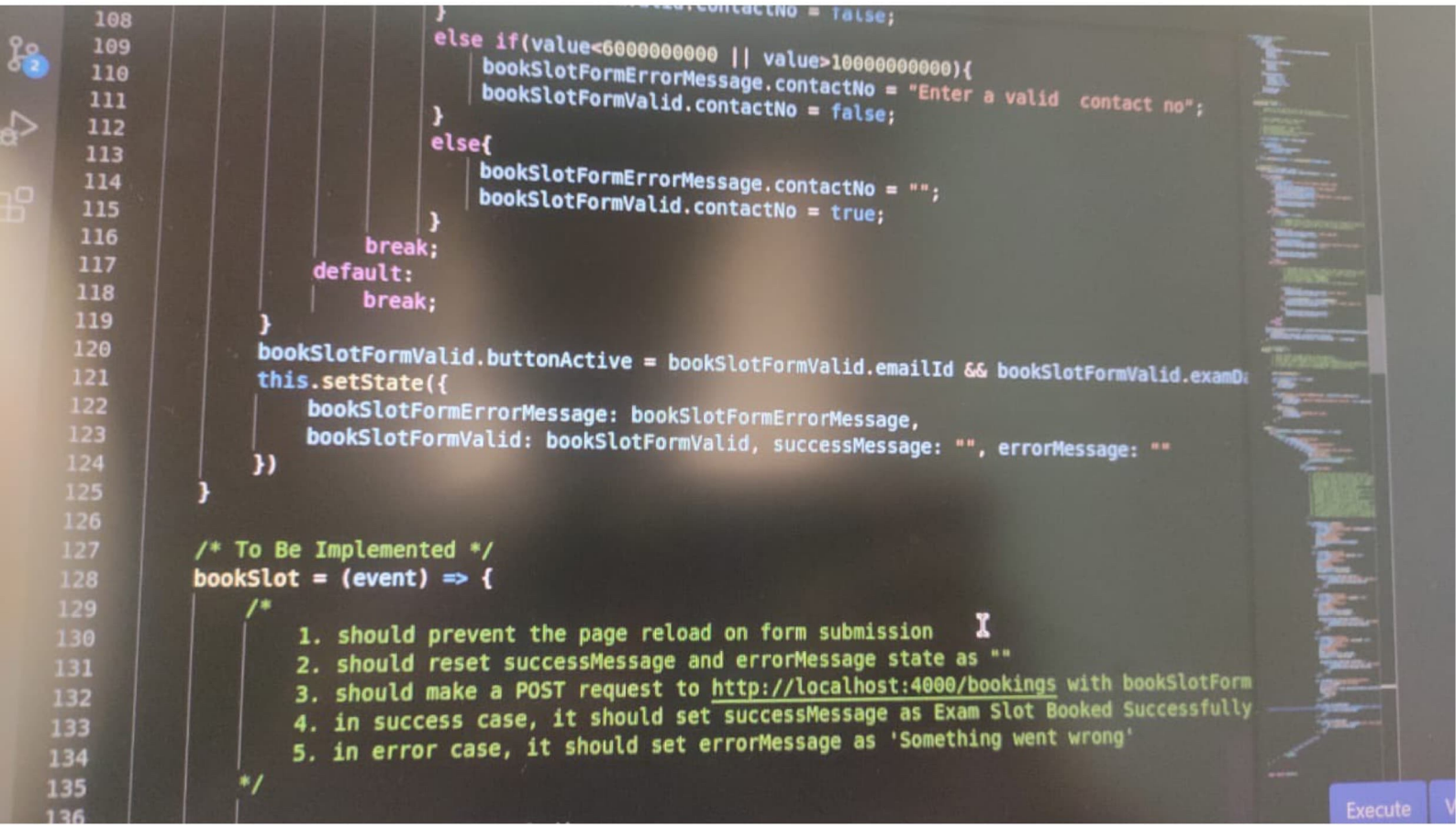
```
55
56      /* To Be Modified */
57      validateField = (fieldName, value) => {
58          let { bookSlotFormErrorMessage, bookSlotFormValid } = this.state;
59
60          switch (fieldName) {
61              case "emailId":
62                  const emailIdRegex = /^[A-z]\w+[A-z][@][A-z]{3,7}(.com)$/
63                  if (value === "") {
64                      bookSlotFormErrorMessage.emailId = "field required";
65                      bookSlotFormValid.emailId = false;
66                  } else if (!value.match(emailIdRegex)) {
67                      bookSlotFormErrorMessage.emailId = "Please enter a valid emailId";
68                      bookSlotFormValid.emailId = false;
69                  } else {
70                      bookSlotFormErrorMessage.emailId = "";
71                      bookSlotFormValid.emailId = true;
72                  }
73                  break;
74              case "examDate":
75                  const examDate = new Date();
76                  /*
77                      1. if examDate field is empty it should set the error message as "field r
78                      2. if examDate field has a past date or today's date it should set the er
79                      3. for valid input it should reset the error message appropriately
80                  */
81                  if(value === ""){
82                      bookSlotFormErrorMessage.examDate = "field required";
83                      bookSlotFormValid.examDate = false;
84                  }
85                  else if(value<=examDate){
```

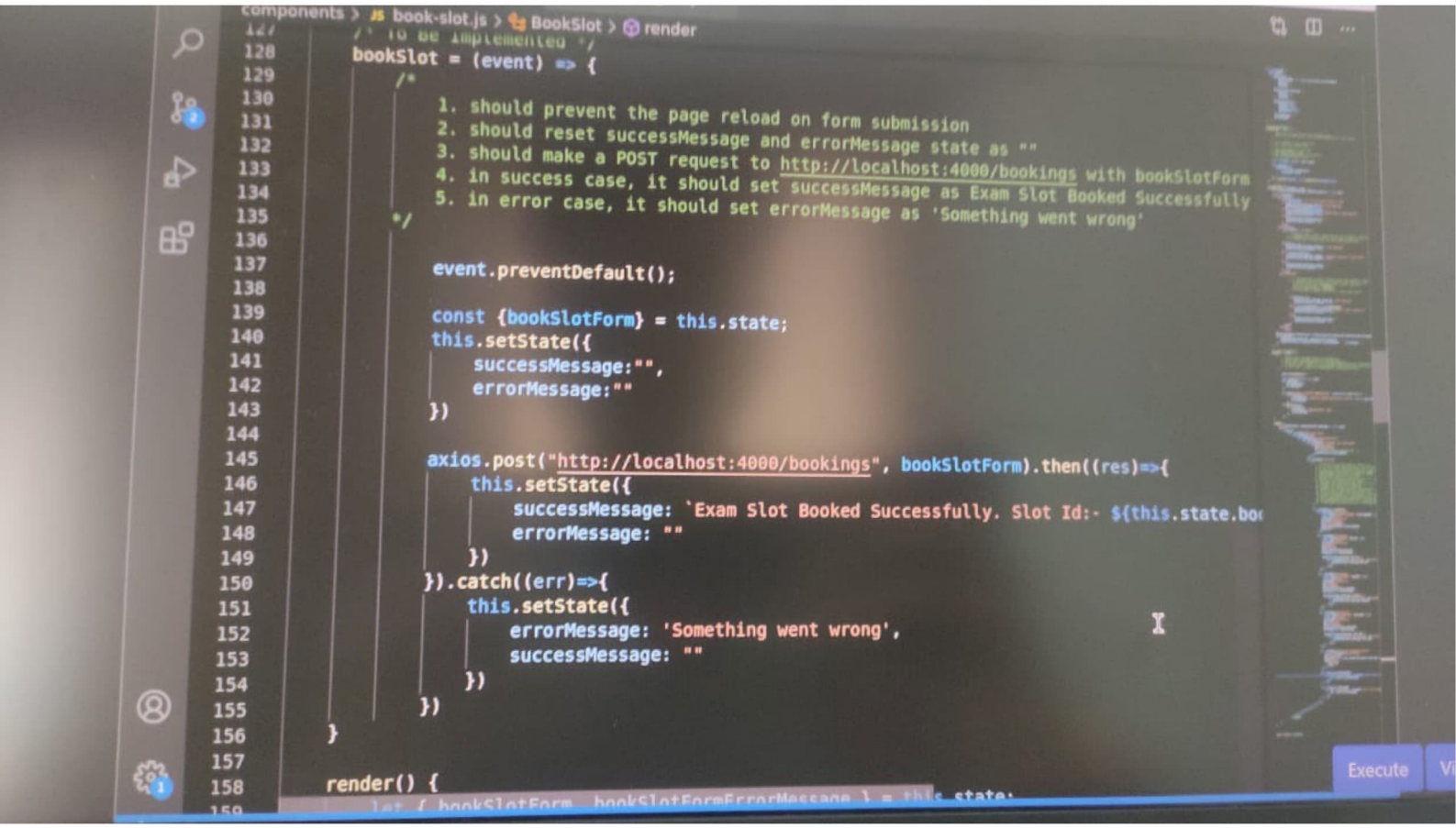
Execute

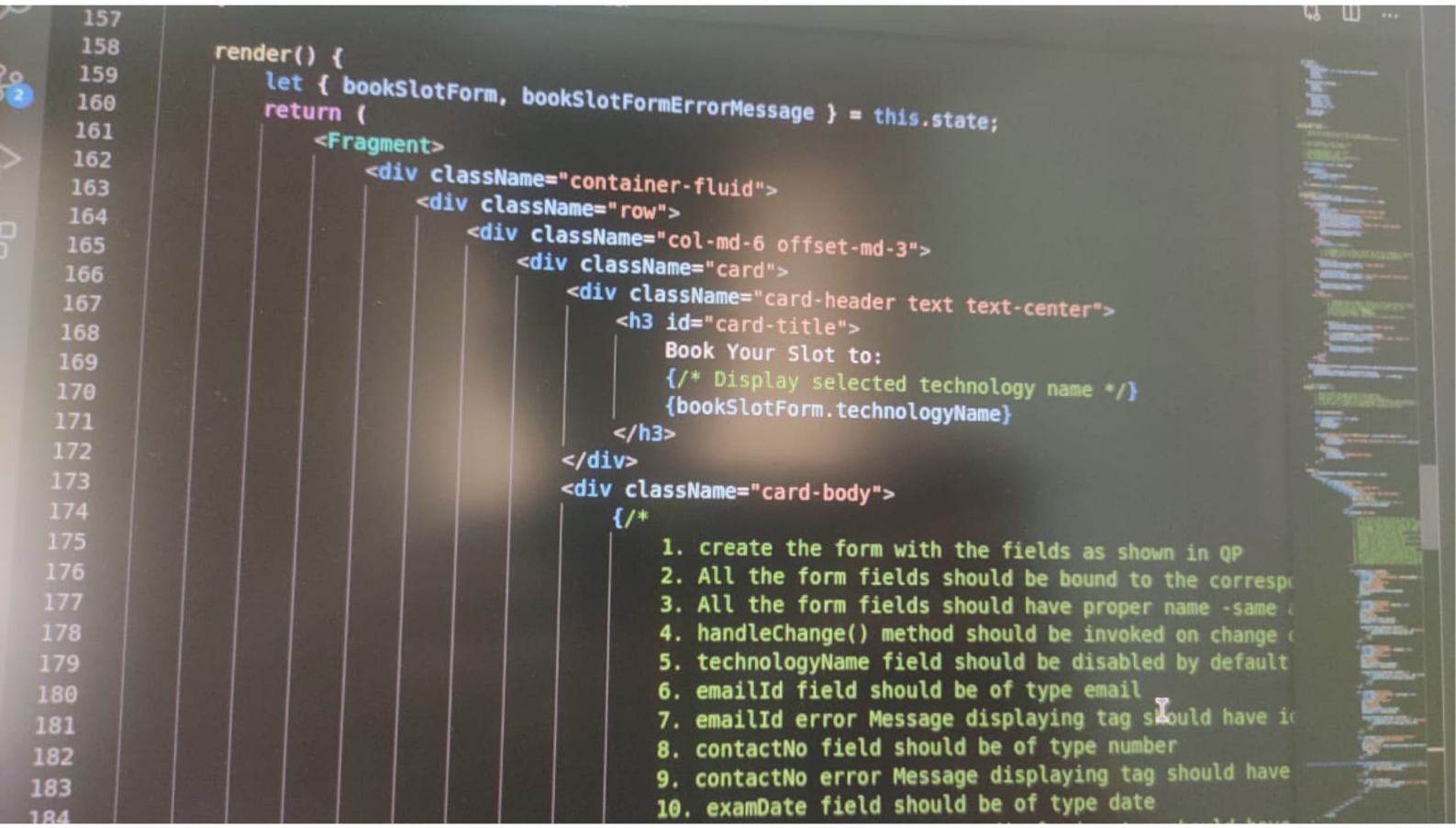


```
            bookSlotFormErrorMessage.emailId = "";
            bookSlotFormValid.emailId = true;
        }
        break;
    case "examDate":
        const examDate = new Date();
        /*
            1. if examDate field is empty it should set the error message as "field r
            2. if examDate field has a past date or today's date it should set the er
            3. for valid input it should reset the error message appropriately
        */
        if(value === ""){
            bookSlotFormErrorMessage.examDate = "field required";
            bookSlotFormValid.examDate = false;
        }
        else if(value<=examDate){
            bookSlotFormErrorMessage.examDate = "Exam date should be a future date";
            bookSlotFormErrorMessage.examDate = false;
        }
        else{
            bookSlotFormErrorMessage.examDate = "";
            bookSlotFormValid.examDate = true;
        }
        break;
    case "contactNo":
        /*
            1. if contactNo field is empty it should set the error message as "field
```

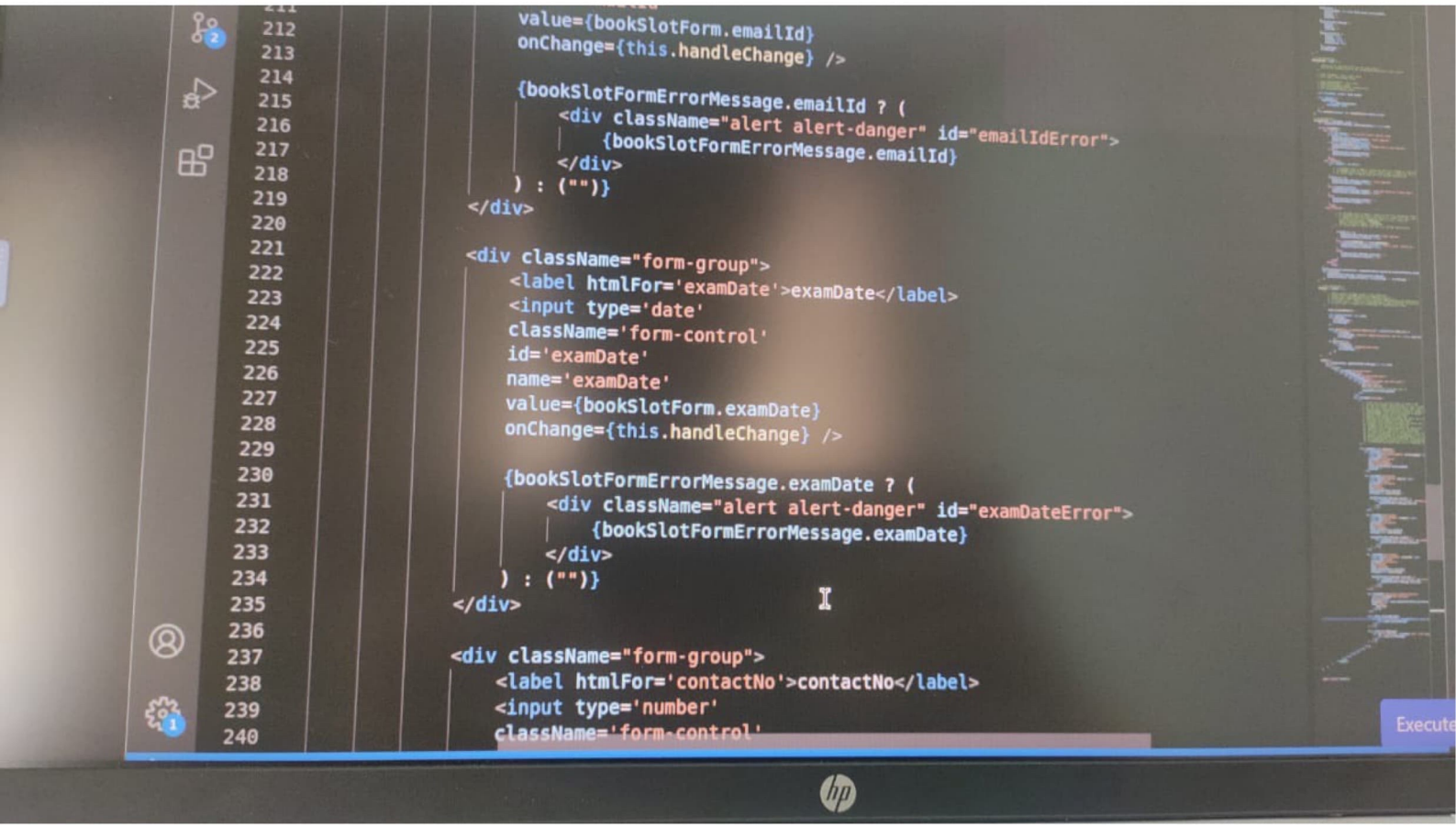

BookSlot > render
bookSlotFormValid.examDate = true;
}
break;
case "contactNo":
/*
1. if contactNo field is empty it should set the error message as "field
2. if contactNo field has values starting from a number less than 6 and
it should set the error message as "Enter a valid contact no"
valid contactNo example - 6000000000
Invalid contactNo Example - 5999999999
3. for valid input it should reset the error message appropriately
*/
if(value === ""){
bookSlotFormErrorMessage.contactNo= "field required";
bookSlotFormValid.contactNo = false;
}
else if(value<6000000000 || value>10000000000){
bookSlotFormErrorMessage.contactNo = "Enter a valid contact no";
bookSlotFormValid.contactNo = false;
}
else{
bookSlotFormErrorMessage.contactNo = "";
bookSlotFormValid.contactNo = true;
}
break;
default:
break;


108 }
109 else if(value<6000000000 || value>10000000000){
110 bookSlotFormErrorMessage.contactNo = "Enter a valid contact no";
111 bookSlotFormValid.contactNo = false;
112 }
113 else{
114 bookSlotFormErrorMessage.contactNo = "";
115 bookSlotFormValid.contactNo = true;
116 }
117 break;
118 default:
119 break;
120 }
121 bookSlotFormValid.buttonActive = bookSlotFormValid.emailId && bookSlotFormValid.examD
122 this.setState({
123 bookSlotFormErrorMessage: bookSlotFormErrorMessage,
124 bookSlotFormValid: bookSlotFormValid, successMessage: "", errorMessage: ""
125 })
126 }
127
128 /* To Be Implemented */
129 bookSlot = (event) => {
130 /*
131 1. should prevent the page reload on form submission
132 2. should reset successMessage and errorMessage state as ""
133 3. should make a POST request to http://localhost:4000/bookings with bookSlotForm
134 4. in success case, it should set successMessage as Exam Slot Booked Successfully
135 5. in error case, it should set errorMessage as 'Something went wrong'
136 */
Execute


components > book-slot.js > BookSlot > render
/* To be implemented */
bookSlot = (event) => {
/*
1. should prevent the page reload on form submission
2. should reset successMessage and errorMessage state as ""
3. should make a POST request to http://localhost:4000/bookings with bookSlotForm
4. in success case, it should set successMessage as Exam Slot Booked Successfully
5. in error case, it should set errorMessage as 'Something went wrong'
*/
event.preventDefault();
const {bookSlotForm} = this.state;
this.setState({
successMessage:"",
errorMessage:""
})
axios.post("http://localhost:4000/bookings", bookSlotForm).then((res)=>{
this.setState({
successMessage: `Exam Slot Booked Successfully. Slot Id:- ${this.state.bo
errorMessage: ""
})
}).catch((err)=>{
this.setState({
errorMessage: 'Something went wrong',
successMessage: ""
})
})
}
render() {
Execute



```
157
158     render() {
159         let { bookSlotForm, bookSlotFormErrorMessage } = this.state;
160         return (
161             <Fragment>
162                 <div className="container-fluid">
163                     <div className="row">
164                         <div className="col-md-6 offset-md-3">
165                             <div className="card">
166                                 <div className="card-header text text-center">
167                                     <h3 id="card-title">
168                                         Book Your Slot to:
169                                         {/* Display selected technology name */}
170                                         {bookSlotForm.technologyName}
171                                     </h3>
172                                 </div>
173                                 <div className="card-body">
174                                     {/*
175                                         1. create the form with the fields as shown in QP
176                                         2. All the form fields should be bound to the corresp
177                                         3. All the form fields should have proper name -same
178                                         4. handleChange() method should be invoked on change
179                                         5. technologyName field should be disabled by default
180                                         6. emailId field should be of type email
181                                         7. emailId error Message displaying tag should have i
182                                         8. contactNo field should be of type number
183                                         9. contactNo error Message displaying tag should have
184                                         10. examDate field should be of type date
```


value={bookSlotForm.emailId}
onChange={this.handleChange} />
{bookSlotFormErrorMessage.emailId ? (

{bookSlotFormErrorMessage.emailId}

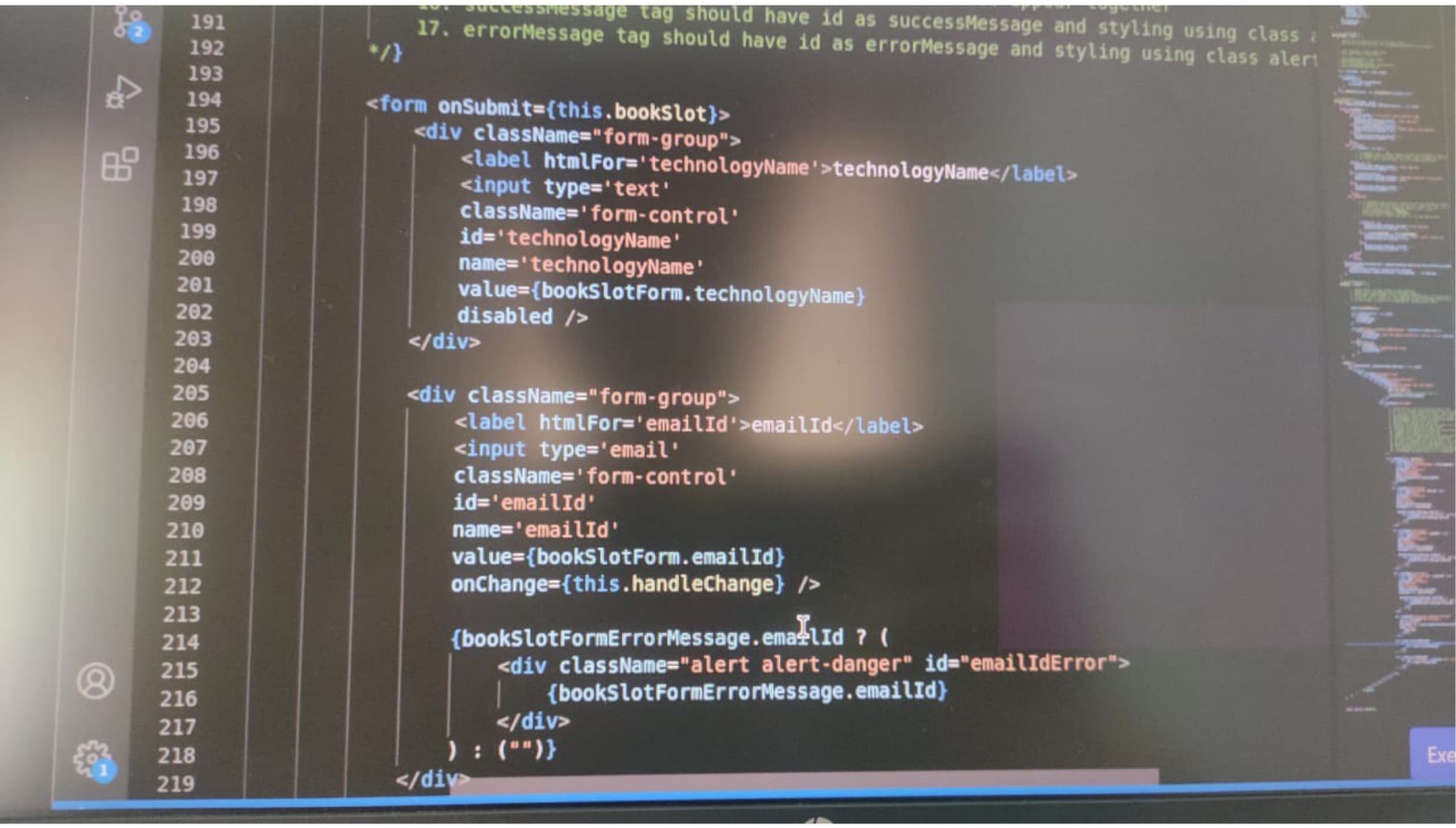
) : ("")}

<label htmlFor='examDate'>examDate</label>
<input type='date'
className='form-control'
id='examDate'
name='examDate'
value={bookSlotForm.examDate}
onChange={this.handleChange} />
{bookSlotFormErrorMessage.examDate ? (

{bookSlotFormErrorMessage.examDate}

) : ("")}

<label htmlFor='contactNo'>contactNo</label>
<input type='number'
className='form-control'
Execute

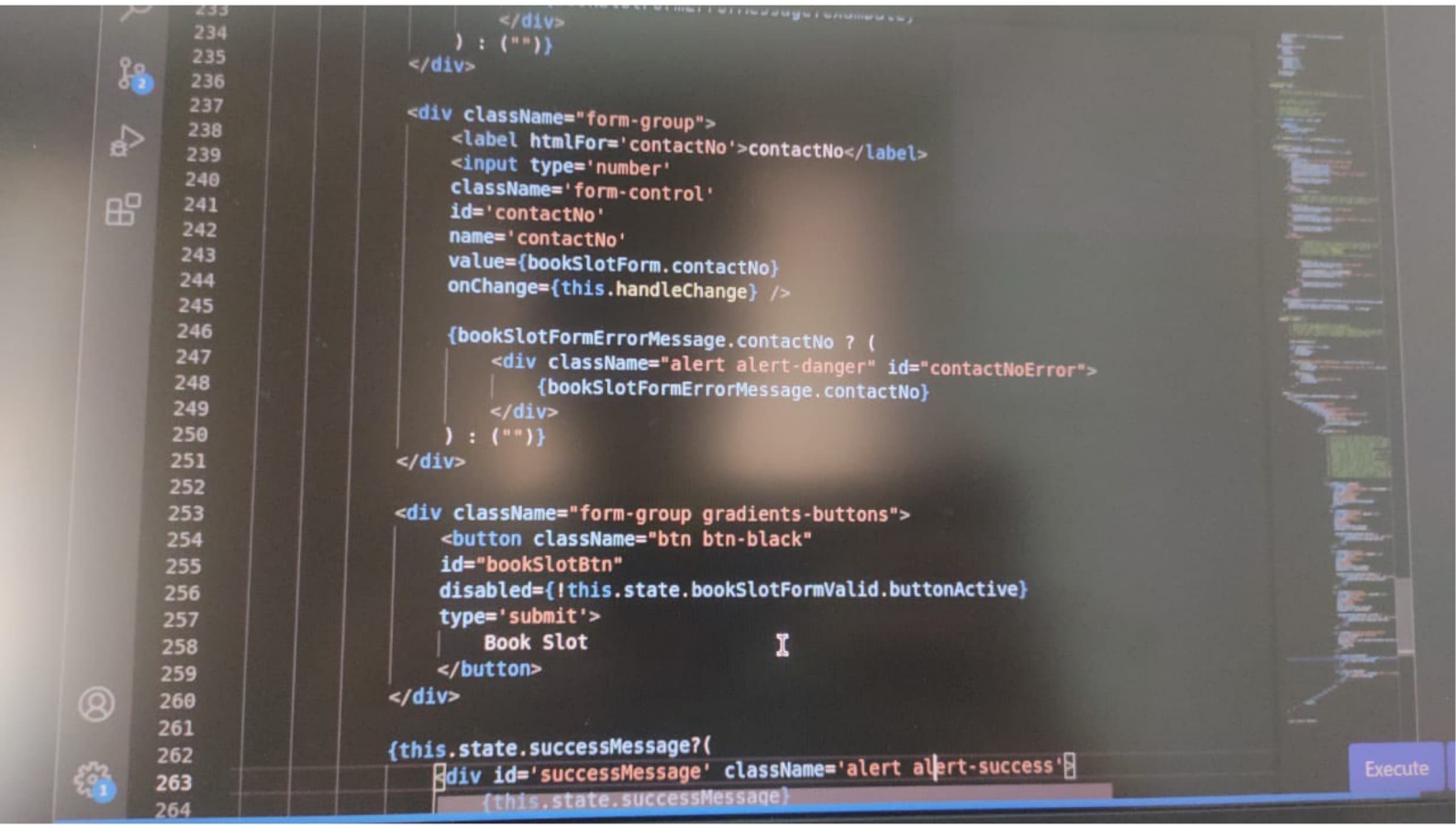

17. successMessage tag should have id as successMessage and styling using class
17. errorMessage tag should have id as errorMessage and styling using class alert
*/}
<form onSubmit={this.bookSlot}>

<label htmlFor='technologyName'>technologyName</label>
<input type='text'
className='form-control'
id='technologyName'
name='technologyName'
value={bookSlotForm.technologyName}
disabled />

<label htmlFor='emailId'>emailId</label>
<input type='email'
className='form-control'
id='emailId'
name='emailId'
value={bookSlotForm.emailId}
onChange={this.handleChange} />
{bookSlotFormErrorMessage.emailId ? (

{bookSlotFormErrorMessage.emailId}

) : ("")}


) : ("")}

<label htmlFor='contactNo'>contactNo</label>
<input type='number'
className='form-control'
id='contactNo'
name='contactNo'
value={bookSlotForm.contactNo}
onChange={this.handleChange} />
{bookSlotFormErrorMessage.contactNo ? (

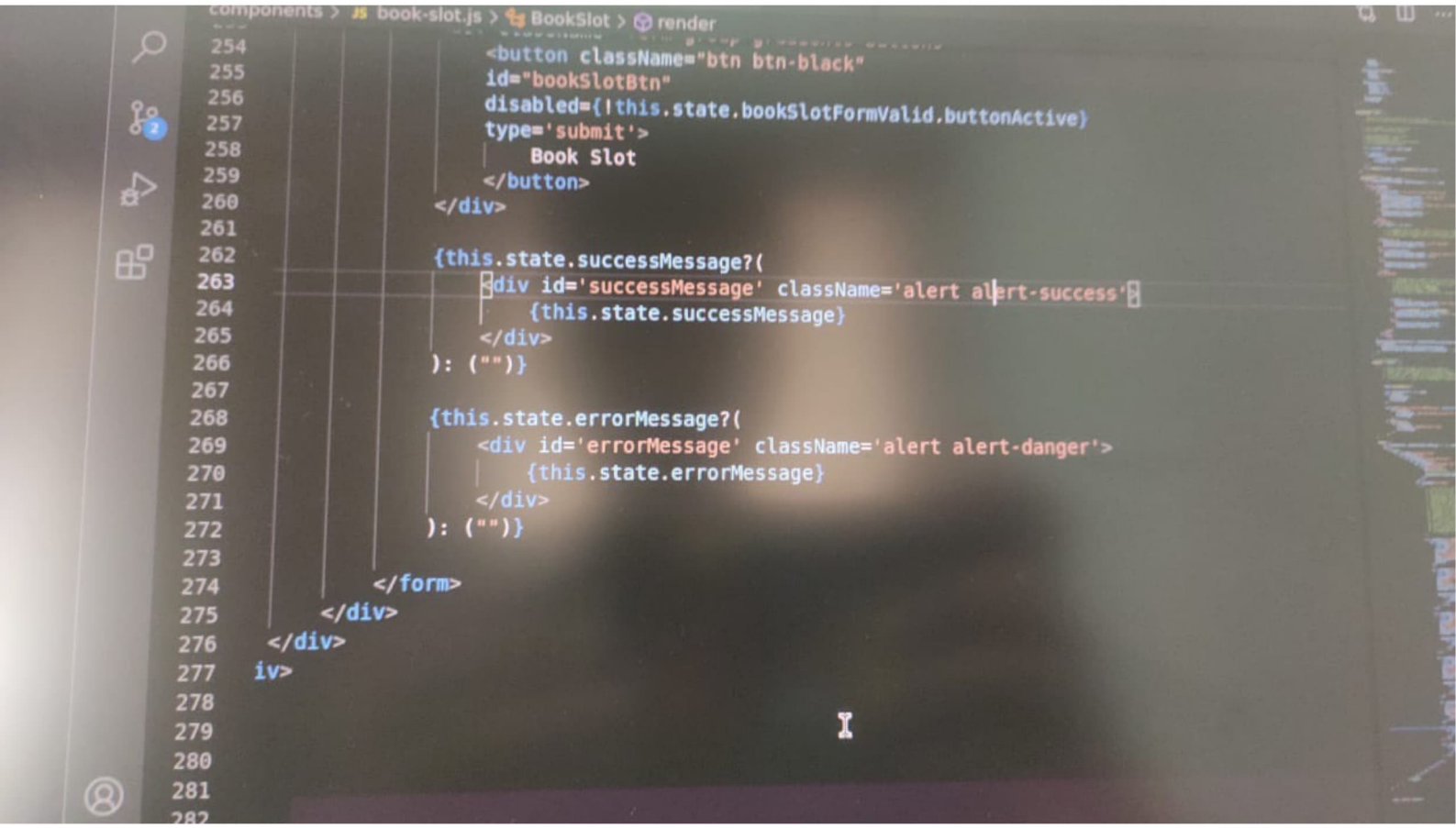
{bookSlotFormErrorMessage.contactNo}

) : ("")}

<button className="btn btn-black"
id="bookSlotBtn"
disabled={!this.state.bookSlotFormValid.buttonActive}
type='submit'>
Book Slot
</button>

{this.state.successMessage?(

{this.state.successMessage}
Execute


components > book-slot.js > BookSlot > render
<button className="btn btn-black"
id="bookSlotBtn"
disabled={!this.state.bookSlotFormValid.buttonActive}
type='submit'>
Book Slot
</button>

{this.state.successMessage?(

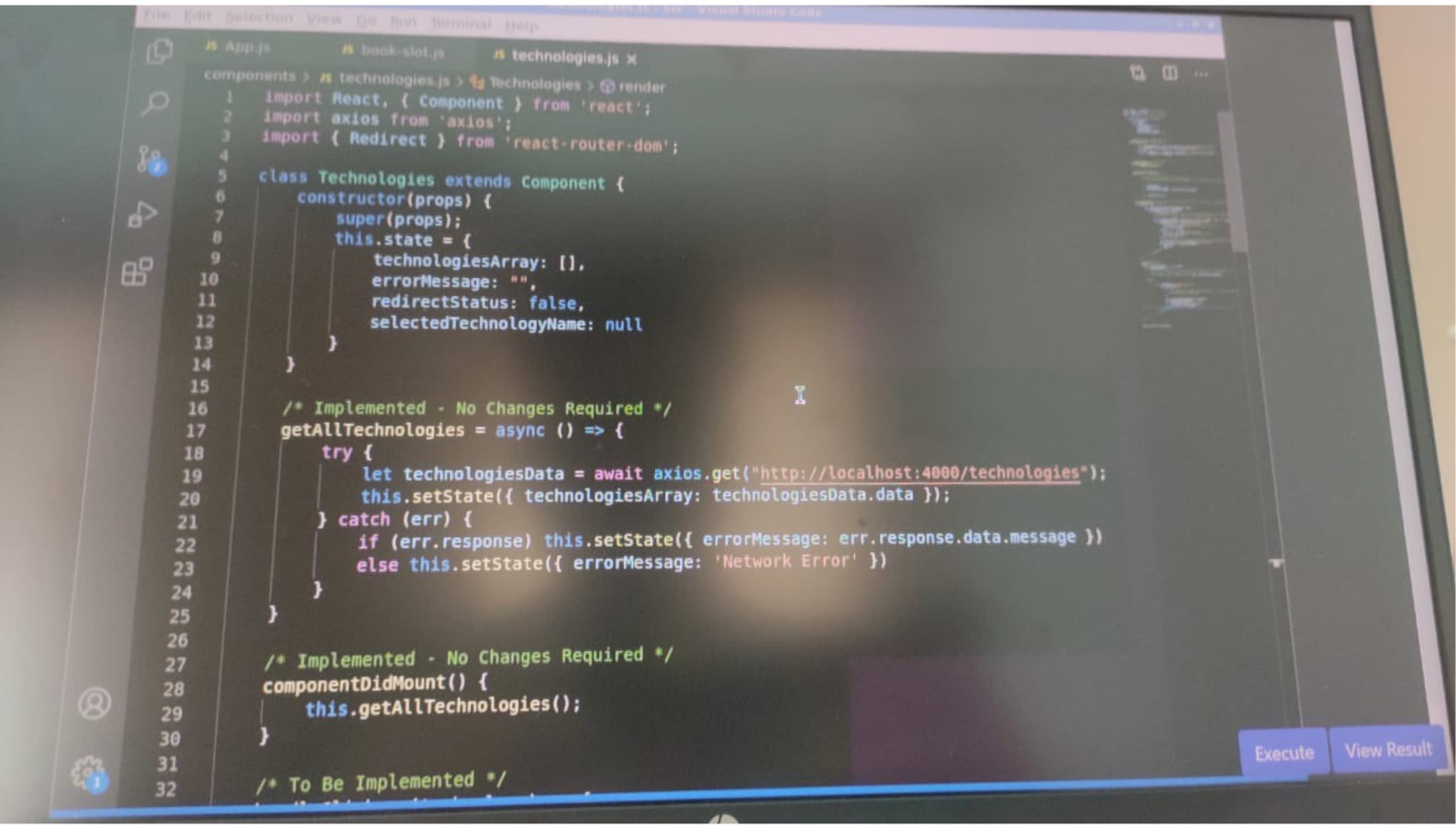
{this.state.successMessage}

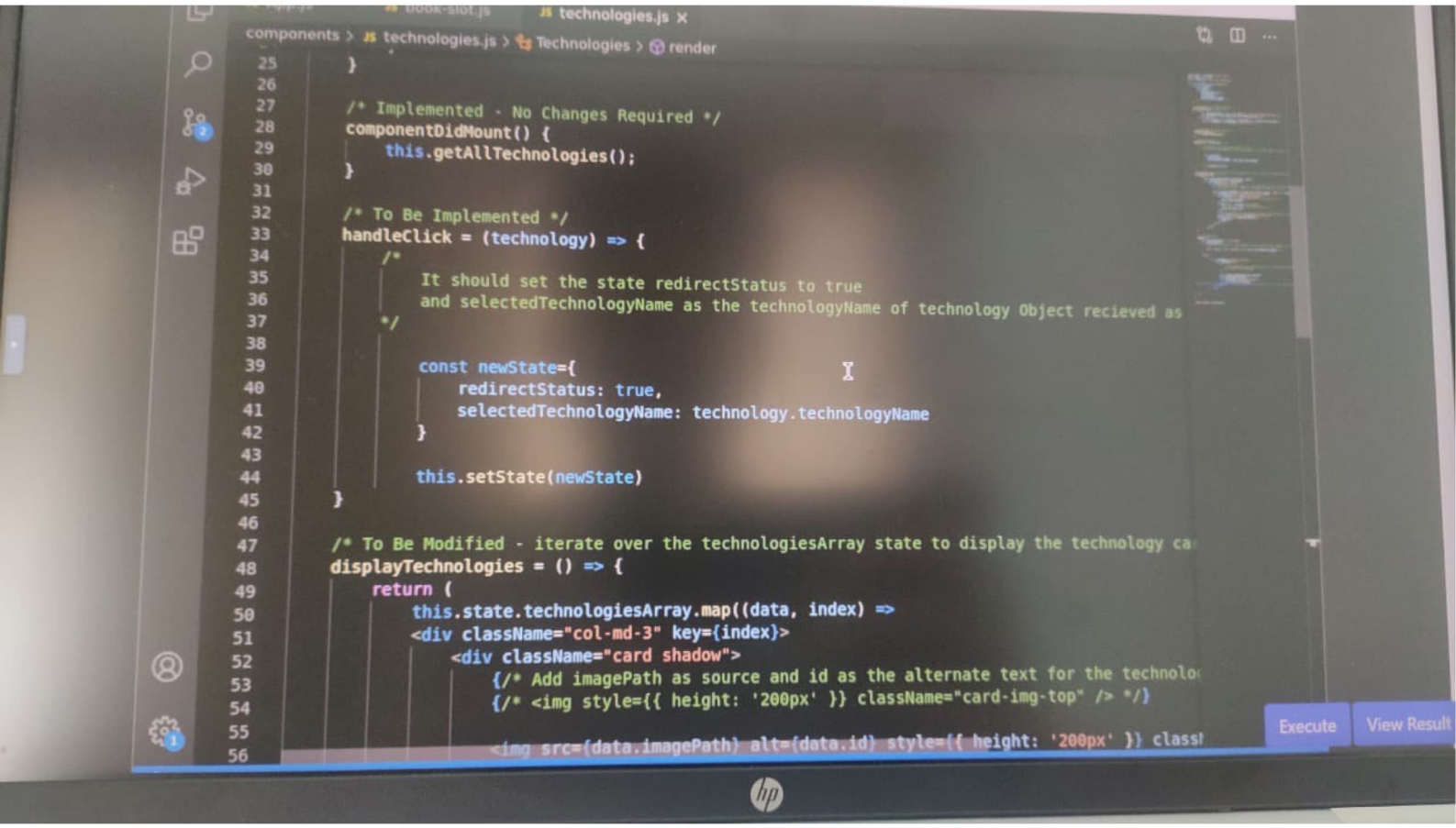
): ("")}
{this.state.errorMessage?(

{this.state.errorMessage}

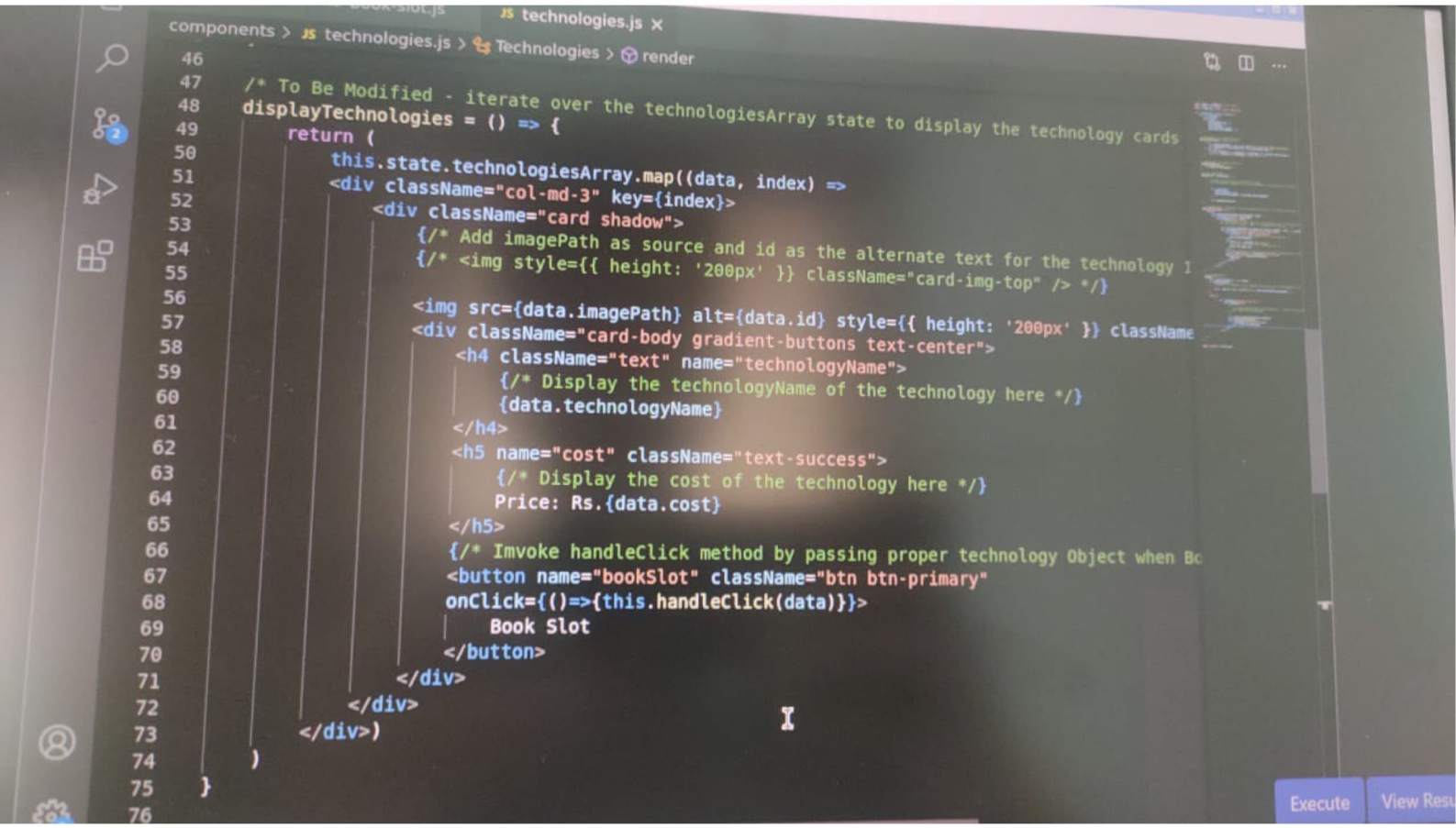
): ("")}
</form>

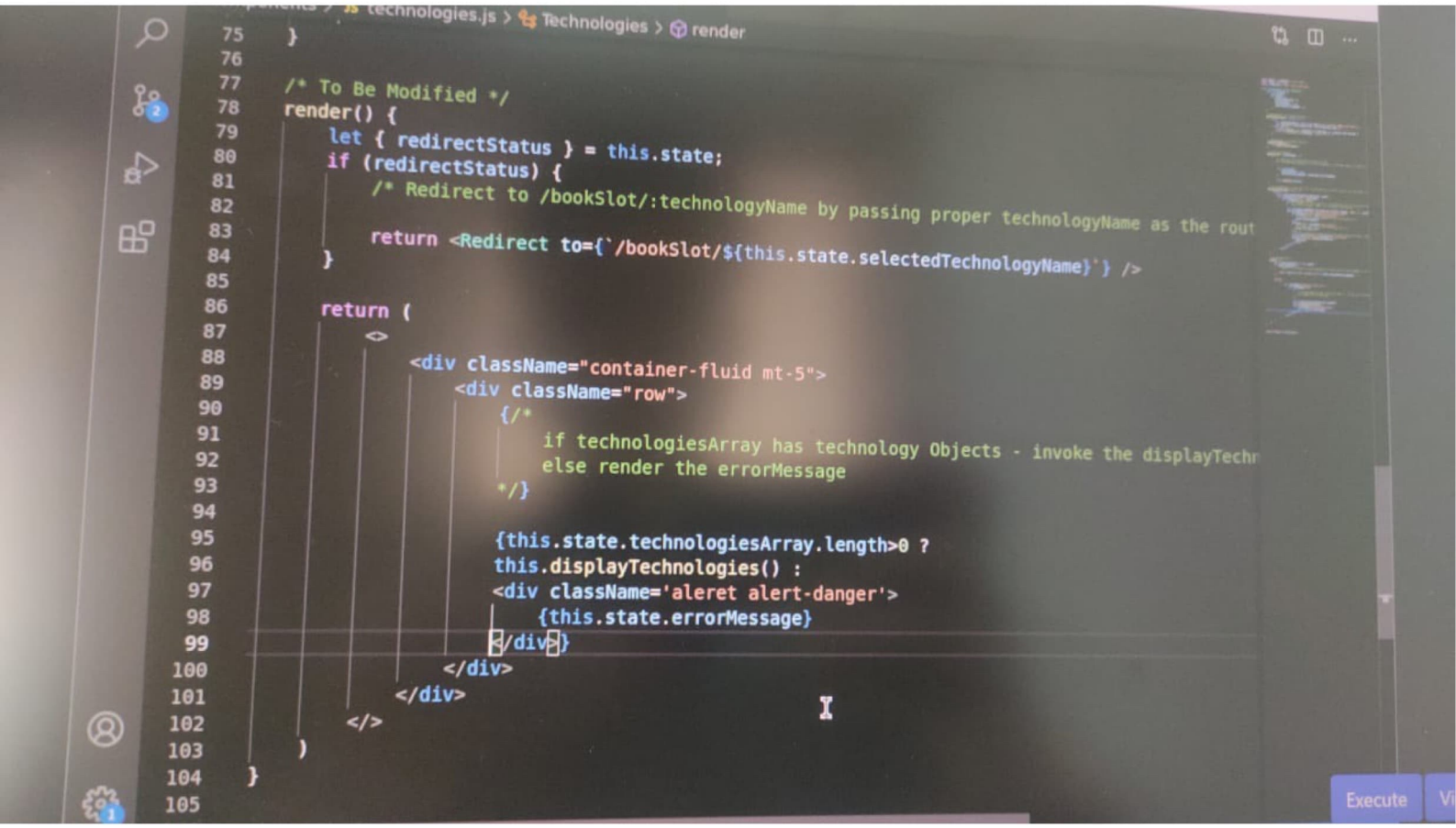
iv>


File Edit Selection View Go Run Terminal Help
App.js
book-slot.js
technologies.js
components > technologies.js > Technologies > render
import React, { Component } from 'react';
import axios from 'axios';
import { Redirect } from 'react-router-dom';
class Technologies extends Component {
constructor(props) {
super(props);
this.state = {
technologiesArray: [],
errorMessage: "",
redirectStatus: false,
selectedTechnologyName: null
}
}
/* Implemented - No Changes Required */
getAllTechnologies = async () => {
try {
let technologiesData = await axios.get("http://localhost:4000/technologies");
this.setState({ technologiesArray: technologiesData.data });
} catch (err) {
if (err.response) this.setState({ errorMessage: err.response.data.message })
else this.setState({ errorMessage: 'Network Error' })
}
}
/* Implemented - No Changes Required */
componentDidMount() {
this.getAllTechnologies();
}
/* To Be Implemented */
Execute
View Result


components > technologies.js > Technologies > render
}
/* Implemented - No Changes Required */
componentDidMount() {
this.getAllTechnologies();
}
/* To Be Implemented */
handleClick = (technology) => {
/*
It should set the state redirectStatus to true
and selectedTechnologyName as the technologyName of technology Object recieved as
*/
const newState={
redirectStatus: true,
selectedTechnologyName: technology.technologyName
}
this.setState(newState)
}
/* To Be Modified - iterate over the technologiesArray state to display the technology ca
displayTechnologies = () => {
return (
this.state.technologiesArray.map((data, index) =>

{/* Add imagePath as source and id as the alternate text for the technolo
{/* */}
<img src={data.imagePath} alt={data.id} style={{ height: '200px' }} class
Execute
View Result

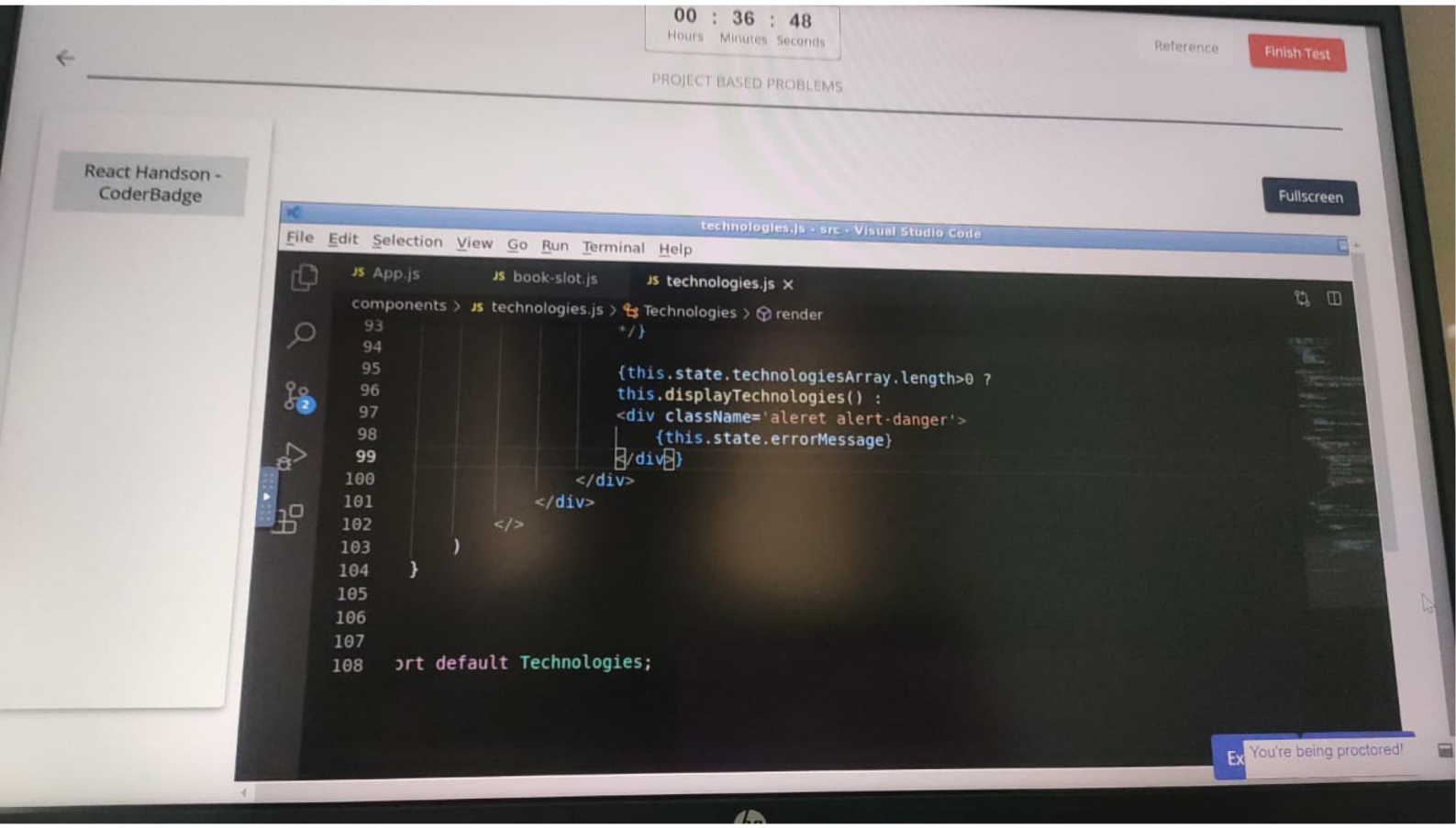

components > technologies.js > Technologies > render
46
47 /* To Be Modified - iterate over the technologiesArray state to display the technology cards
48 displayTechnologies = () => {
49 return (
50 this.state.technologiesArray.map((data, index) =>
51
52
53 {/* Add imagePath as source and id as the alternate text for the technology I
54 {/* */}
55
56 <img src={data.imagePath} alt={data.id} style={{ height: '200px' }} classNam
57
58 <h4 className="text" name="technologyName">
59 {/* Display the technologyName of the technology here */}
60 {data.technologyName}
61 </h4>
62 <h5 name="cost" className="text-success">
63 {/* Display the cost of the technology here */}
64 Price: Rs.{data.cost}
65 </h5>
66 {/* Imvoke handleClick method by passing proper technology Object when Bo
67 <button name="bookSlot" className="btn btn-primary"
68 onClick={()=>{this.handleClick(data)}}>
69 Book Slot
70 </button>
71
72
73)
74)
75 }
76
Execute
View Resu


technologies.js > Technologies > render
}
/* To Be Modified */
render() {
let { redirectStatus } = this.state;
if (redirectStatus) {
/* Redirect to /bookSlot/:technologyName by passing proper technologyName as the rout
return <Redirect to={`/bookSlot/${this.state.selectedTechnologyName}`} />
}
return (
<>

{/*
if technologiesArray has technology Objects - invoke the displayTechn
else render the errorMessage
*/}
{this.state.technologiesArray.length>0 ?
this.displayTechnologies() :

{this.state.errorMessage}
}

</>
)
}
Execute


00 : 36 : 48
Hours Minutes Seconds
Reference
Finish Test
PROJECT BASED PROBLEMS
React Handson - CoderBadge
Fullscreen
technologies.js - src - Visual Studio Code
File Edit Selection View Go Run Terminal Help
App.js
book-slot.js
technologies.js
components > technologies.js > Technologies > render
*/}
{this.state.technologiesArray.length>0 ?
this.displayTechnologies() :

{this.state.errorMessage}
}

</>
)
}
ort default Technologies;
You're being proctored!