

```
JS App.js X book-slot.js B technologies.js
JS App.js > ⚡ App
1 import './App.css';
2 import { BrowserRouter as Router } from 'react-router-dom';
3 import React from 'react';
4 import { Route, Switch, Link, Redirect } from "react-router-dom";
5 import BookSlot from './components/book-slot';
6 import Technologies from './components/technologies';
7
8 function App() {
9   return (
10     <React.Fragment>
11       <Router>
12         <nav className="navbar navbar-expand-lg navbar-dark bg-dark shadow">
13           <span className="navbar-brand">
14             <b>Coder Badge</b>
15           </span>
16           <button className="navbar-toggler" type="button" data-toggle="collapse" data-target="#collapsibleNavbar">
17             <span className="navbar-toggler-icon"></span>
18           </button>
19           <div className="collapse navbar-collapse" id="collapsibleNavbar">
20             <ul className="navbar-nav">
21               <li className="nav-item mr-2">
22                 <Link className="nav-link font-weight-bold" to="/technologies">
23                   Book Slot
24                 </Link>
25               </li>
26             </ul>
27           </div>
28         </nav>
29         <Switch>
30           /*
31             Add the routes as mentioned below:
32             1. It should redirect to /technologies page for default URL i.e. http://localhost:3001
```

Execute

The screenshot shows a code editor with a dark theme. The file being edited is `App.js`. The code contains a `Switch` component with four `Route` components defined. A comment block provides instructions for adding routes:

```
25     </li>
26   </ul>
27 </div>
28 </nav>
29 <Switch>
30   /* Add the routes as mentioned below:
31    1. It should redirect to /technologies page for default URL i.e. http://localhost:3001/
32    2. It should load BookSlot component for /bookSlot/:technologyName
33    3. It should load Technologies component for /technologies
34    4. It should redirect to /technologies page for any invalid URL e.g. http://localhost:3001/someinvalidurl
35   */
36 </Switch>
37 <Router>
38   <React.Fragment>
39     <Route path='/' render={()=> <Redirect to='/technologies' />} />
40     <Route path='/bookSlot/:technologyName' component={BookSlot} />
41     <Route path='/technologies' component={Technologies} />
42     <Route path='*' render={()=> <Redirect to='/technologies' />} />
43   </Switch>
44 </Router>
45 </React.Fragment>
46 }
47
48 export default App;
49 }
```

```
27      }
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 received value to the corresponding 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) =>
```

```
54
55  /* To Be Modified */
56  validateField = (fieldName, value) => {
57    let { bookSlotFormErrorMessage, bookSlotFormValid } = this.state;
58
59    switch (fieldName) {
60      case "emailId":
61        const emailIdRegex = /^[A-z]\w+[A-z]@[A-z]{3,7}(.com)$/
62        if (value === "") {
63          bookSlotFormErrorMessage.emailId = "field required";
64          bookSlotFormValid.emailId = false;
65        } else if (!value.match(emailIdRegex)) {
66          bookSlotFormErrorMessage.emailId = "Please enter a valid emailId";
67          bookSlotFormValid.emailId = false;
68        } else {
69          bookSlotFormErrorMessage.emailId = "";
70          bookSlotFormValid.emailId = true;
71        }
72        break;
73      case "examDate":
74        const examDate = new Date();
75        /*
76          1. if examDate field is empty it should set the error message as "field required"
77          2. if examDate field has a past date or today's date it should set the error message as "invalid date"
78          3. for valid input it should reset the error message appropriately
79        */
80        if(value === ""){
81          bookSlotFormErrorMessage.examDate = "field required";
82          bookSlotFormValid.examDate = false;
83        }
84      else if(value<=examDate){
85        bookSlotFormErrorMessage.examDate = "invalid date";
86        bookSlotFormValid.examDate = false;
87      }
88    }
89  }
90
```

Execute

```
71
72         bookSlotFormValid.emailId = "";
73     }
74     break;
75 case "examDate":
76     const examDate = new Date();
77     /*
78      1. if examDate field is empty it should set the error message as "field re-
79      2. if examDate field has a past date or today's date it should set the er-
80      3. for valid input it should reset the error message appropriately
81     */
82     if(value === ""){
83         bookSlotFormErrorMessage.examDate = "field required";
84         bookSlotFormValid.examDate = false;
85     }
86     else if(value<=examDate){
87         bookSlotFormErrorMessage.examDate = "Exam date should be a future date";
88         bookSlotFormErrorMessage.examDate = false;
89     }
90     else{
91         bookSlotFormErrorMessage.examDate = "";
92         bookSlotFormValid.examDate = true;
93     }
94     break;
95 case "contactNo":
96     /*
97      1. if contactNo field is empty it should set the error message as "field re-
98      2. if contactNo field has a value less than 10 digits it should set the er-
99      3. if contactNo field has values starting from a number less than 6 and 1
100     */
101
```

```
90     }
91     bookSlotFormValid.examDate = true;
92   }
93   break;
94 case "contactNo":
95 /*
96 1. if contactNo field is empty it should set the error message as "field
97 2. if contactNo field has values starting from a number less than 6 and
98 it should set the error message as "Enter a valid contact no"
99 valid contactNo example - 6000000000
100 Invalid contactNo Example - 5999999999
101 3. for valid input it should reset the error message appropriately
102 */
103
104 if(value == ""){
105   bookSlotFormErrorMessage.contactNo= "field required";
106   bookSlotFormValid.contactNo = false;
107 }
108 else if(value<6000000000 || value>10000000000){
109   bookSlotFormErrorMessage.contactNo = "Enter a valid contact no";
110   bookSlotFormValid.contactNo = false;
111 }
112 else{
113   bookSlotFormErrorMessage.contactNo = "";
114   bookSlotFormValid.contactNo = true; I
115 }
116 break;
117 default:
118   break;
```

```
108     if(bookSlotFormValid.contactNo == false);
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.examDate != null;
122 this.setState({
123     bookSlotFormErrorMessage: bookSlotFormErrorMessage,
124     bookSlotFormValid: bookSlotFormValid, successMessage: "", errorMessage: ""
125 })
126
127 /* To Be Implemented */
128 bookSlot = (event) => {
129     /*
130         1. should prevent the page reload on form submission
131         2. should reset successMessage and errorMessage state as ""
132         3. should make a POST request to http://localhost:4000/bookings with bookSlotForm
133         4. in success case, it should set successMessage as Exam Slot Booked Successfully
134         5. in error case, it should set errorMessage as 'Something went wrong'
135     */
136 }
```

Execute

```
components > js book-slot.js > BookSlot > render
127     /* To be implemented */
128     bookSlot = (event) => {
129         /*
130          1. should prevent the page reload on form submission
131          2. should reset successMessage and errorMessage state as ""
132          3. should make a POST request to http://localhost:4000/bookings with bookSlotForm
133          4. in success case, it should set successMessage as Exam Slot Booked Successfully
134          5. in error case, it should set errorMessage as 'Something went wrong'
135      */
136
137     event.preventDefault();
138
139     const {bookSlotForm} = this.state;
140     this.setState({
141         successMessage:"",
142         errorMessage:""
143     })
144
145     axios.post("http://localhost:4000/bookings", bookSlotForm).then((res)=>{
146         this.setState({
147             successMessage: `Exam Slot Booked Successfully. Slot Id:- ${this.state.bookSlotForm.slotId}`,
148             errorMessage: ""
149         })
150     }).catch((err)=>{
151         this.setState({
152             errorMessage: 'Something went wrong',
153             successMessage: ""
154         })
155     })
156
157
158     render() {
159         let {bookSlotForm, bookSlotFormErrorMessage} = this.state;
```

```
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 corresponding state
177                                         3. All the form fields should have proper name - same as the state
178                                         4. handleChange() method should be invoked on change event
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 id="emailIdError"
182                                         8. contactNo field should be of type number
183                                         9. contactNo error Message displaying tag should have id="contactNoError"
184                                         10. examDate field should be of type date

```

```
    211
  212  value={bookSlotForm.emailId}
  213  onChange={this.handleChange} />
  214
  215  {bookSlotFormErrorMessage.emailId ? (
  216    <div className="alert alert-danger" id="emailIdError">
  217      |   {bookSlotFormErrorMessage.emailId}
  218    </div>
  219  ) : ("")}
  220  </div>
  221
  222  <div className="form-group">
  223    <label htmlFor='examDate'>examDate</label>
  224    <input type='date'
  225      className='form-control'
  226      id='examDate'
  227      name='examDate'
  228      value={bookSlotForm.examDate}
  229      onChange={this.handleChange} />
  230
  231  {bookSlotFormErrorMessage.examDate ? (
  232    <div className="alert alert-danger" id="examDateError">
  233      |   {bookSlotFormErrorMessage.examDate}
  234    </div>
  235  ) : ("")}

  236  <div className="form-group">
  237    <label htmlFor='contactNo'>contactNo</label>
  238    <input type='number'
  239      className='form-control'
```

Execute



```
191      16. successMessage tag should have id as successMessage and styling using class alert-success together
192      17. errorMessage tag should have id as errorMessage and styling using class alert-danger together
193  */
194
195  <form onSubmit={this.bookSlot}>
196    <div className="form-group">
197      <label htmlFor='technologyName'>technologyName</label>
198      <input type='text'
199        className='form-control'
200        id='technologyName'
201        name='technologyName'
202        value={bookSlotForm.technologyName}
203        disabled />
204    </div>
205
206    <div className="form-group">
207      <label htmlFor='emailId'>emailId</label>
208      <input type='email'
209        className='form-control'
210        id='emailId'
211        name='emailId'
212        value={bookSlotForm.emailId}
213        onChange={this.handleChange} />
214
215      {bookSlotFormErrorMessage.emailId ? (
216        <div className="alert alert-danger" id="emailIdError">
217          | {bookSlotFormErrorMessage.emailId}
218        </div>
219      ) : ("")}>
220    </div>
```

```
233           </div>
234       ) : ("")
235   </div>

236
237   <div className="form-group">
238     <label htmlFor='contactNo'>contactNo</label>
239     <input type='number'
240       className='form-control'
241       id='contactNo'
242       name='contactNo'
243       value={bookSlotForm.contactNo}
244       onChange={this.handleChange} />
245
246     {bookSlotFormErrorMessage.contactNo ? (
247       <div className="alert alert-danger" id="contactNoError">
248         {bookSlotFormErrorMessage.contactNo}
249       </div>
250     ) : ("")}
251   </div>

252
253   <div className="form-group gradients-buttons">
254     <button className="btn btn-black"
255       id="bookSlotBtn"
256       disabled={!this.state.bookSlotFormValid.buttonActive}
257       type='submit'>
258       Book Slot
259     </button>
260   </div>

261   {this.state.successMessage?
262     <div id='successMessage' className='alert alert-success'>
263       {this.state.successMessage}
264   
```

Execute

```
components > book-slot.js > BookSlot > render
  254   <button className="btn btn-black"
  255     id="bookSlotBtn"
  256     disabled={!this.state.bookSlotFormValid.buttonActive}
  257     type='submit'>
  258       Book Slot
  259     </button>
  260   </div>
  261
  262   {this.state.successMessage?( 
  263     <div id='successMessage' className='alert alert-success'>
  264       {this.state.successMessage}
  265     </div>
  266   ): ("")}
  267
  268   {this.state.errorMessage?( 
  269     <div id='errorMessage' className='alert alert-danger'>
  270       {this.state.errorMessage}
  271     </div>
  272   ): ("")}
  273
  274   </form>
  275 </div>
  276 </div>
  277 iv>
  278
  279
  280
  281
  282
```

```
File Edit Selection View Go Run Terminal Help
JS App.js JS book-slot.js JS technologies.js X
components > JS technologies.js > Technologies > render
1 import React, { Component } from 'react';
2 import axios from 'axios';
3 import { Redirect } from 'react-router-dom';
4
5 class Technologies extends Component {
6     constructor(props) {
7         super(props);
8         this.state = {
9             technologiesArray: [],
10            errorMessage: '',
11            redirectStatus: false,
12            selectedTechnologyName: null
13        }
14    }
15
16    /* Implemented - No Changes Required */
17    getAllTechnologies = async () => {
18        try {
19            let technologiesData = await axios.get("http://localhost:4000/technologies");
20            this.setState({ technologiesArray: technologiesData.data });
21        } catch (err) {
22            if (err.response) this.setState({ errorMessage: err.response.data.message });
23            else this.setState({ errorMessage: 'Network Error' });
24        }
25    }
26
27    /* Implemented - No Changes Required */
28    componentDidMount() {
29        this.getAllTechnologies();
30    }
31
32    /* To Be Implemented */
```

Execute View Result

```
components > js technologies.js < Technologies > render
25     }
26
27     /* Implemented - No Changes Required */
28     componentDidMount() {
29         this.getAllTechnologies();
30     }
31
32     /* To Be Implemented */
33     handleClick = (technology) => {
34         /*
35             It should set the state redirectStatus to true
36             and selectedTechnologyName as the technologyName of technology Object received as
37             */
38
39         const newState={
40             redirectStatus: true,
41             selectedTechnologyName: technology.technologyName
42         }
43
44         this.setState(newState)
45     }
46
47     /* To Be Modified - iterate over the technologiesArray state to display the technology card */
48     displayTechnologies = () => {
49         return (
50             this.state.technologiesArray.map((data, index) =>
51                 <div className="col-md-3" key={index}>
52                     <div className="card shadow">
53                         {/* Add imagePath as source and id as the alternate text for the technology image */}
54                         {/* <img style={{ height: '200px' }} className="card-img-top" /> */}
55                         <img src={data.imagePath} alt={data.id} style={{ height: '200px' }} className="card-img-top" />
56                     </div>
57                 </div>
58             )
59         )
60     }
61
62     getAllTechnologies = () => {
63         fetch('http://localhost:3001/technologies')
64             .then(response => response.json())
65             .then(data => this.setState({ technologies: data }))
66     }
67
68     componentWillMount() {
69         this.getAllTechnologies();
70     }
71
72     render() {
73         return (
74             <div>
75                 <h1>Available Technologies</h1>
76                 <table border="1">
77                     <thead>
78                         <tr>
79                             <th>Technology Name</th>
80                         <th>Image Path</th>
81                         <th>Action</th>
82                     </tr>
83                 <tbody>
84                     {this.displayTechnologies()}
85                 </tbody>
86             </div>
87         )
88     }
89 }
```

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              <div className="col-md-3" key={index}>
52                  <div className="card shadow">
53                      /* Add imagePath as source and id as the alternate text for the technology I
54                      /* <img style={{ height: '200px' }} className="card-img-top" /> */
55                      <img src={data.imagePath} alt={data.id} style={{ height: '200px' }} className="card-img-top" />
56                      <div className="card-body gradient-buttons text-center">
57                          <h4 className="text" name="technologyName">
58                              /* Display the technologyName of the technology here */
59                              {data.technologyName}
60                          </h4>
61                          <h5 name="cost" className="text-success">
62                              /* Display the cost of the technology here */
63                              Price: Rs.{data.cost}
64                          </h5>
65                          /* Invoke handleClick method by passing proper technology Object when Book Slot button is clicked */
66                          <button name="bookSlot" className="btn btn-primary"
67                              onClick={()=>{this.handleClick(data)}}
68                              > Book Slot
69                          </button>
70                      </div>
71                  </div>
72              </div>
73          )
74      )
75  }
76 }
```

Execute View Results

```
Technologies.js > Technologies > render
75 }
76
77 /* To Be Modified */
78 render() {
79     let { redirectStatus } = this.state;
80     if (redirectStatus) {
81         /* Redirect to /bookSlot/:technologyName by passing proper technologyName as the route
82         return <Redirect to={`/bookSlot/${this.state.selectedTechnologyName}`} />
83     }
84
85     return (
86         <>
87             <div className="container-fluid mt-5">
88                 <div className="row">
89                     {/* 
90                         if technologiesArray has technology Objects - invoke the displayTechnologies()
91                         else render the errorMessage
92                     */}
93
94                     {this.state.technologiesArray.length>0 ?
95                         this.displayTechnologies() :
96                         <div className='alert alert-danger'>
97                             {this.state.errorMessage}
98                         </div>
99                     }
100                </div>
101            </div>
102        </>
103    )
104
105 }
```

Execute

00 : 36 : 48
Hours Minutes Seconds

Reference

Finish Test

PROJECT BASED PROBLEMS

React Handson -
CoderBadge

Fullscreen

File Edit Selection View Go Run Terminal Help

technologies.js - src - Visual Studio Code

components > technologies.js > Technologies > render

```
93     */
94
95     if(this.state.technologiesArray.length>0) {
96         this.displayTechnologies();
97         <div className='aleret alert-danger'>
98             {this.state.errorMessage}
99         </div>
100    </div>
101  </div>
102 }
103 }
104 }
105
106
107
108 export default Technologies;
```

Ex You're being proctored!