

```
File Edit Selection View Go Run ... dummy-folder
JS App.js JS index.js index.html JS BookPT.js JS ShowBooking.js package.json JS Exercises.js
src > JS App.js > ...
9 function App() {
10   return (
11     <>
12       <BrowserRouter>
13         <Navbar />
14         <Routes>
15           {/*
16             Add the routes as mentioned below:
17             1. It should redirect to Exercises component for default URL i.e. http://localhost:3000/.
18             2. It should load BookPT component for /book_pt/:exerciseType.
19             3. It should load ShowBooking component for /show_booking/:exerciseType.
20             4. It should redirect to Exercises component for any wrong URL e.g. http://localhost:3000/wrongpage.
21           */}
22           <Route path="/" element={<Exercises/>} />
23           <Route path="/book_pt/:exerciseType" element={<BookPT/>} />
24           <Route path="/show_booking/:exerciseType" element={<ShowBooking/>} />
25           <Route path="*" element={<Exercises/>} />
26         </Routes>
27       </BrowserRouter>
28     </>
29   );
30 }
31 export default App;
```

BookPT.js Starts now

src &gt; component &gt; JS BookPT.js &gt; BookPT &gt; validateField

```
1  import React, { useState } from "react";
2  import { useParams } from "react-router-dom";
3  import axios from "axios";
4
5  export default function BookPT() {
6    const params = useParams();
7
8    const [formData, setFormData] = useState({
9      exerciseType: params.exerciseType,
10     name: "",
11     contactNo: "",
12     startDate: "",
13   });
14
15   const [formErrorMessage, setFormErrorMessage] = useState({
16     name: "",
17     contactNo: "",
18     startDate: "",
19   });
20
21   const [formDataValid, setFormDataValid] = useState({
22     name: false, //value given to 'name' field is invalid, if false
23     contactNo: false, //value given to 'contactNo field is invalid, if false
24     startDate: false, //value given to 'startDate field is invalid, if false
25     buttonActive: false, //the form button is disabled if set to false
26   });
27
28   const [successMessage, setSuccessMessage] = useState("");
29   const [errorMessage, setErrorMessage] = useState("");
```

I

src &gt; component &gt; JS BookPT.js &gt; BookPT

```
5  export default function BookPT() {
26  });
27
28  const [successMessage, setSuccessMessage] = useState("");
29  const [errorMessage, setErrorMessage] = useState("");
30
31  /* Use the messages from the below messages object
32  to display all kinds of success, error and validation messages instead of hardcoding. */
33  const messages = {
34    REQUIRED_VALIDATION: "Field is required",
35    NAME_VALIDATION: "Must be at least 3 characters long",
36    DATE_VALIDATION: "You cannot select today's date or a date in the past. Please choose a future date to begin your package.",
37    CONTACT_VALIDATION: "Cannot start with a number below 6 and must be 10 digits",
38    TRAINER_BOOKED_SUCCESS: "Personal trainer booked successfully",
39    TRAINER_BOOKED_ERROR: "Something went wrong",
40  };
41
42  const url = "http://localhost:4000/bookings";
43
44  /* To Be Implemented. */
45  /* This function will be invoked whenever the value provided to any of the input fields changes */
46  const handleChange = (event) => {
47    /*
48     1. use the name and value property from the event object.
49     2. set the corresponding formData state property to the received value.
50     3. invoke the validateField by passing event object's name and value as parameter.
51    */
52  }
```



```
42 const url = "http://localhost:4000/bookings";
43
44 /* To Be Implemented. */
45 /* This function will be invoked whenever the value provided to any of the input fields changes */
46 const handleChange = (event) => {
47   /*
48    1. use the name and value property from the event object.
49    2. set the corresponding formData state property to the received value.
50    3. invoke the validateField by passing event object's name and value as parameter.
51   */
52
53   const { name, value } = event.target;
54   setFormData((prevFormData) => ({ ...prevFormData, [name]: value }));
55   validateField(name, value);
56 };
57
58 /* To Be Modified. */
59 /* This method validates the input fields based on the mentioned conditions. */
60 const validateField = (fieldName, fieldValue) => {
61   const newFormDataErrorMessages = { ...formErrorMessage };
62   const newFormDataValid = { ...formDataValid };
63
```

I

Common requirement for all the inputs as follows:

1. If input is valid then
  - a. set the associated error message as empty string.
  - b. set the associated formValid value as true,
2. Otherwise set associated formValid value as false.

\*/

switch (fieldName) {

case "name":

const nameRegex = /^[A-Za-z][A-Za-z ]{3,29}\$/;

/\*

1. If the value of name is empty,
  - set the associated error message as REQUIRED\_VALIDATION.
2. If the value contains less than 3 alphabets or greater than 29 alphabets or if the value contains other than alphabet characters,
  - set the associated error message as NAME\_VALIDATION.

\*/

if (fieldValue === "") {

newFormDataErrorMessages.name = messages.REQUIRED\_VALIDATION;

newFormDataValid.name = false;

} else if (!nameRegex.test(fieldValue)) {

newFormDataErrorMessages.name = messages.NAME\_VALIDATION;

newFormDataValid.name = false;

} else {

newFormDataErrorMessages.name = "";

newFormDataValid.name = true;

}

break;

```
case "startDate":
```

```
let today = new Date().setUTCHours(0, 0, 0, 0);
```

```
let checkIn = new Date(fieldValue).setUTCHours(0, 0, 0, 0);
```

```
/*
```

```
1. If startDate field is empty it should set the error message as REQUIRED_VALIDATION.
```

```
2. If startDate field has past date or today's date,  
it should set the error message as DATE_VALIDATION.
```

```
*/
```

```
if (fieldValue === "") {
```

```
newFormDataErrorMessages.startDate = messages.REQUIRED_VALIDATION;
```

```
newFormDataValid.startDate = false;
```

```
} else if (checkIn <= today) {
```

```
newFormDataErrorMessages.startDate = messages.DATE_VALIDATION;
```

```
newFormDataValid.startDate = false;
```

```
} else {
```

```
newFormDataErrorMessages.startDate = "";
```

```
newFormDataValid.startDate = true;
```

```
}
```

```
break;
```



```
112
113 case "contactNo":
114     /*
115     1. If contactNo field is empty it should set the error message as REQUIRED_VALIDATION.
116     2. If contactNo field has fieldValues starting from a number less than 6 and
117         the number of digits is less than ten,
118         it should set the error message as CONTACT_VALIDATION.
119         Hint: Use Regex pattern.
120     */
121     const contactNoRegex = /^[6-9][0-9]{9}$/;
122     if (fieldValue === "") {
123         newFormDataErrorMessages.contactNo = messages.REQUIRED_VALIDATION;
124         newFormDataValid.contactNo = false;
125     } else if (!contactNoRegex.test(fieldValue)) {
126         newFormDataErrorMessages.contactNo = messages.CONTACT_VALIDATION;
127         newFormDataValid.contactNo = false;
128     } else {
129         newFormDataErrorMessages.contactNo = "";
130         newFormDataValid.contactNo = true;
131     }
132     break;
133
134 default:
135     break;
136 }
```

I



```
137
138  /*
139  |   Set isActive to true when the associated formValid property of all three input fields are true.
140  */
141
142  if (newFormDataValid.name && newFormDataValid.contactNo && newFormDataValid.startDate) {
143      newFormDataValid.buttonActive = true;
144  } else {
145      newFormDataValid.buttonActive = false;
146  }
147  setFormErrorMessage(newFormDataErrorMessages);
148  setFormDataValid(newFormDataValid);
149  };
150
```

```
151  /* To Be Implemented. */
152  /* This function will save the booking form data in the backend. */
153  const bookPT = async (event) => {
154      /*
155       1. Should prevent default behaviour on form submission.
156       2. Should make a POST request to http://localhost:4000/bookings with formData
157       as the request body.
158       3. In case of success, it should set successMessage as TRAINER_BOOKED_SUCCESS.
159       4. In case of error, it should set errorMessage as TRAINER_BOOKED_ERROR.
160      */
161      event.preventDefault();
162      try {
163          const response = await axios.post(url, formData);
164          setSuccessMessage(messages.TRAINER_BOOKED_SUCCESS);
165          setErrorMessage("");
166      } catch (error) {
167          setErrorMessage(messages.TRAINER_BOOKED_ERROR);
168          setSuccessMessage("");
169      }
170
171
172
173  }
```

```
172  /* To be modified. */
173  /* The below code designs registration form. */
174  return (
175      <>
176      <div className="container-fluid">
177          <div className="row">
178              <div className="col-md-6 offset-md-3">
179                  <div
180                      className="card mt-5"
181                      style={{ backgroundColor: "rgba(255, 255, 255, 0.75)" }}
182                  >
183                      <div className="card-header text text-center">
184                          <h3 id="card-title" data-testid="form-title">
185                              Book Your Trainer for: {params.exerciseType}
186                              /* Display selected Exercise Type */
187                          </h3>
188                      </div>
189                      <div className="card-body">
190                          {/*
```



```
191 The common requirements for each input field for the following form are mentioned below:
192 1. On form submission, bookPT method should be invoked.
193 2. Bind all form fields to their corresponding properties in the formData state.
194 3. Give all form fields a name attribute that matches their formData state property.
195 4. Set proper type attributes and bootstrap classes for all form fields.
196 5. Invoke the handleChange() method on every input field value change.
197 6. contactNo field should be of type number.
198 7. startDate field should be of type date.
199 8. The "Book PT" button should have bootstrap classes "btn btn-block btn-dark mt-3".
200 9. Disable the button until the buttonActive state property is true.
201 10. Show error message and success message on form submission as appropriate, but not both at the same time.
202 */}
```

```
203 <form data-testid="bookingForm" onSubmit={bookPT}>
204   <div className="form-group" id="exerciseType">
205     <label
206       htmlFor="exerciseType"
207       data-testid="labelExerciseType"
208     >
209       Personal Trainer For
210     </label>
211     /* input1 - Do not change.*/
212     <input
213       id="exerciseType"
214       name="exerciseType"
215       className="form-control"
216       value={formData.exerciseType}
217       type="text"
218       disabled
```

```
211      {/* input1 - Do not change.*/}  
212      <input  
213          id="exerciseType"  
214          name="exerciseType"  
215          className="form-control"  
216          value={formData.exerciseType}  
217          type="text"  
218          disabled  
219          data-testid="inputExerciseType"  
220      />  
221 </div>
```

```
222 <div className="form-group" id="name">
223   <label htmlFor="name" data-testid="labelName">
224     Name
225   </label>
226   { /* input2 */ }
227   <input
228     id="name"
229     type="text"
230     data-testid="inputName"
231     name="name"
232     value={formData.name}
233     onChange={handleChange}
234     className="form-control"
235
236   />
237   <div id="nameError" data-testid="errorName" className="badge text-bg-danger">
238     { /* Show name error message and style it with class "badge text-bg-danger". */ }
239     {formErrorMessage.name}
240   </div>
241 </div>
```



```
242 <div className="form-group" id="contactNo">
243   <label htmlFor="contactNo" data-testid="labelContactNo">
244     Contact No
245   </label>
246   { /* input3 */ }
247   <input
248     id="contactNo"
249     data-testid="inputContactNo"
250     type="number"
251     name="contactNo"
252     value={formData.contactNo}
253     onChange={handleChange}
254     className="form-control"
255   />
256   <div id="contactNoError" data-testid="errorContactNo" className="badge text-bg-danger">
257     { /* Show contact number error message and style it with class "badge text-bg-danger". */ }
258     {formErrorMessage.contactNo}
259   </div>
260 </div>
```

```
261 <div className="form-group" id="startDate">
262   <label htmlFor="startDate" data-testid="labelDate">
263     Package Start Date
264   </label>
265   { /* input4 */ }
266   <input id="startDate"
267     data-testid="inputDate"
268     name="startDate"
269     type="date"
270     value={formData.startDate}
271     onChange={handleChange}
272     className="form-control"/>
273   <div id="startDateError" data-testid="errorDate" className="badge text-bg-danger">
274     { /* Show date error message and style it with class "badge text-bg-danger". */ }
275     {formErrorMessage.startDate}
276   </div>
277 </div> I
```

```
277     </div>
278     <div className="form-group" id="btnForm" >
279       { /* button */ }
280       <button id="bookPTBtn" data-testid="bookPTButton"
281         className="btn btn-block btn-dark mt-3"
282         disabled={!formDataValid.buttonActive}>
283         Book PT
284       </button>
285     </div>
286   </form>
287   <div id="successMessage" data-testid="successMessageForm" className="badge text-bg-success">
288     { /* Display success message here and style it with class "badge text-bg-success". */ }
289     {successMessage}
290   </div>
291   <div id="errorMessage" data-testid="errorMessageForm" className="badge text-bg-danger">
292     { /* Display error message here and style it with class "badge text-bg-danger". */ }
293     {errorMessage}
294   </div>
295 </div>
296 </div>
297 </div>
298 </div>
299 </div>
300 </>
301 );
302 }
303
```



Exercises.js starts now

src > component > JS Exercises.js > Exercises

```
1  import React, { useEffect, useState } from "react";
2  import axios from "axios";
3  import { useNavigate } from "react-router-dom";
4
5  export default function Exercises() {
6    const [exercisesArray, setExercisesArray] = useState([]);
7    const [errorMessage, setErrorMessage] = useState("");
8    const navigate = useNavigate();
9
10   const messages = {
11     NETWORK_ERROR: "Network Error",
12   };
13   const url = "http://localhost:4000/exercises";
14   /* To Be Implemented. */
15   /* This method fetches all the exercises from url i.e. http://localhost:4000/exercises. */
16   const getAllExercises = async () => {
17     /*
18      1. Send an AXIOS GET request to the url and handle the success and error cases appropriately.
19      2. In case of success, assign received response's data to exercisesArray state.
20      3. In case of error, set the errorMessage state as NETWORK_ERROR from the messages object.
21     */
22     try {
23       const response = await axios.get(url);
24       setExercisesArray(response.data);
25     } catch (error) {
26       setErrorMessage(messages.NETWORK_ERROR);
27     }
28   };
```

```
30  /* Implemented - No Changes Required. */
31  useEffect(() => {
32    |   getAllExercises();
33  }, []);
34
35  /* To Be Implemented */
36  /* This method navigates to BookPT component. */
37  const handleClickBooking = (exercise) => {
38    /*
39     |   1. Navigate to BookPT component with route params as "exerciseType" of the
40     |   selected exercise.
41     |   Note : Use the navigate method.
42     */
43    navigate(`/book-pt/${exercise.exerciseType}`);
44
45  };
46
47  /* To Be Implemented */
48  /* This method navigates to ShowBooking component. */
49  const handleClickShowBooking = (exercise) => {
50    /*
51     |   1. Navigate to ShowBooking component with route params as "exerciseType" of
52     |   the selected exercise.
53     */
54    navigate(`/show-booking/${exercise.exerciseType}`);
55  }
```



```
57  /* To Be Modified. */
58  /* This method iterates over the exercisesArray to display all the exercises cards. */
59  const displayExercises = () => {
60    let counter = 1000;
61    /*
62     Start iteration over the exercisesArray state below the end of this comment block.
63     return a map function to iterate over the exercisesArray and move the below counter incrementation
64     and return block inside the map function.
65     You are not required to change any attributes where counter is mentioned
66    */
67    return( exercisesArray.map((exercise) => {
68      counter++;
69      return (
70        <div className="col-md{3}" key={counter}>
71          <div className="card shadow">
72            {/*
73             1. Set image source as imagePath.
74             2. Set alternate text as id of exercise.
75            */}
76            <img
77              data-testid={`imageT${counter}`}
78              style={{ height: "200px" }}
79              className="card-img-top"
80              src={exercise.imagePath}
81              alt={exercise.id}
82            />
```

```
83 <div className="card-body gradient-buttons text-center">
```

```
84 <h4
```

```
    className="text"
```

```
    name="exerciseType"
```

```
    data-testid={`exerciseTypeT${counter}`}
```

```
  >
```

```
    {/* Display the exerciseType of the exercise here. */}
```

```
    {exercise.exerciseType}
```

```
  </h4>
```

```
  <h5
```

```
    name="cost"
```

```
    className="text-success"
```

```
    data-testid={`costT${counter}`}
```

```
  >
```

```
    {/* Display the cost of the exercise here. */}
```

```
    Price:{exercise.cost}$/month
```

```
  </h5>
```

```
100      /* Invoke handleClickBooking method by passing proper exercise Object when "Book Trainer" button is clicked. */
101      <button
102          name="bookTrainer"
103          className="btn btn-primary mt-2"
104          style={{ marginRight: "5px" }}
105          data-testid={`bookButtonT${counter}`}
106          onClick={() => handleClickBooking(exercise)}>
107          Book Trainer
108      </button>
109      /* Invoke handleClickShowBooking method by passing proper exercise Object when "Show Booking" button is clicked. */
110      <button
111          name="bookTrainer"
112          className="btn btn-primary mt-2"
113          data-testid={`showButtonT${counter}`}
114          onClick={() => handleClickShowBooking(exercise)}>
115          Show Booking
116      </button>
117  </div>
118 </div>
119 </div>
120 ); //End of return
121 /*
122  Iteration should be ended here.
123 */
124 }));
125 };
```

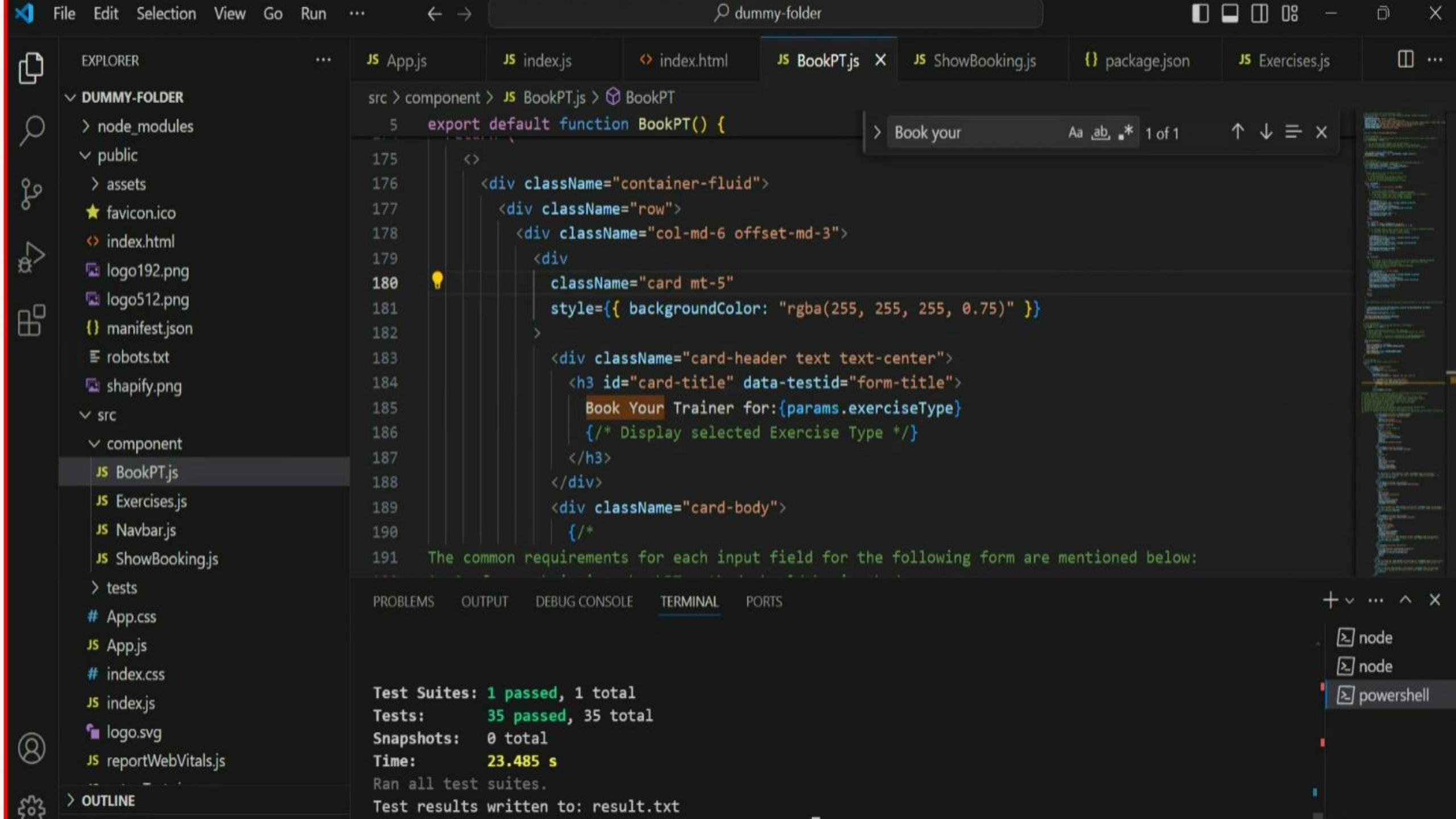
```
126 return (
127   <>
128     <div className="container">
129       <h1 style={{ fontWeight: "bold", color: "black" }}>
130         Make yourself stronger than your excuses...
131       </h1>
132       <p className="lead" style={{ fontWeight: "700", color: "black" }}>
133         Book a Personal Coach on your journey to SHAPIFY yourself
134       </p>
135     </div>
136     <div className="container-fluid mt-5">
137       {errorMessage && (
138         <h2
139           className="text-dark text-center"
140           data-testid="errorMessageExercises"
141         >
142           {/* Render error message here. */}
143           {errorMessage}
144         </h2>
145       )}
146       {exercisesArray.length > 0 &&
147         (
148           <div className="row">
149             {/* Invoke the displayExercises method. */}
150             {displayExercises()}
151           </div>
152         )}
153     </div>
```



```
130     Make yourself stronger than your excuses...
131 </h1>
132 <p className="lead" style={{ fontWeight: "700", color: "black" }}>
133     Book a Personal Coach on your journey to SHAPIFY yourself
134 </p>
135 </div>
136 <div className="container-fluid mt-5">
137     {errorMessage && (
138         <h2
139             className="text-dark text-center"
140             data-testid="errorMessageExercises"
141         >
142             {/* Render error message here. */}
143             {errorMessage}
144         </h2>
145     )}
146     {exercisesArray.length > 0 &&
147         (
148             <div className="row">
149                 {/* Invoke the displayExercises method. */}
150                 {displayExercises()}
151             </div>
152         )}
153     </div>
154 </>
155 );
156 }
```

For 35 test cases....

**In BookPT.js file line no.185 .... Delete the space between[ book  
your trainer for:] and{params.exerciseType}**



END