

Frontend > src > JS App.js > App

```
1 import React from 'react';
2 import { Link, Redirect, Route, Switch } from 'react-router-dom';
3 import { BrowserRouter as Router } from 'react-router-dom';
4 import './App.css';
5 import Locations from './components/Locations';
6 import FarmVisit from './components/FarmVisit';
7
8 function App() {
9   return (
10     <Router>
11     <div>
12       <nav className="navbar navbar-expand-lg navbar-light bg-warning">
13         <Link className="navbar-brand" to="/">Infy Green</Link>
14       </nav>
15       <Switch>
16         <Route exact path="/" component={Locations}/>
17         <Route path="/farmVisit/:locationName" component={FarmVisit}/>
18         <Route path="/locations" component={Locations}/>
19         <Route path="*" render={()=><Redirect to="/locations"/>}/>
20
21         {/*
22          Add the routes as mentioned below:
23          1. It should redirect to /locations page for default URL i.e. http://localhost:
24          2. It should load FarmVisit component for /farmVisit/:locationName
25          3. It should load Locations component for /locations
26          4. It should redirect to /locations page for any invalid URL e.g. http://localh
27          */}
28       </Switch>
29     </div>
30   </Router>
31 )
32 }
```



JS App.js

JS Locations.js x

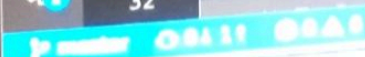
JS FarmVisit.js



Frontend > src > components > JS Locations.js > Locations > render

```
1 import React, { Component } from 'react'
2 import axios from 'axios';
3 import { Redirect } from 'react-router-dom';
4
5 export default class Locations extends Component {
6
7   constructor(props) {
8     super(props);
9     this.state = {
10       locationsArray: [],
11       errorMessage: "",
12       redirectStatus: false,
13       selectedLocationName: null
14     }
15   }
16
17   /* Implemented - No Changes Required */
18   getAllLocations = async () => {
19     try {
20       let locationsData = await axios.get("http://localhost:4000/locations");
21       this.setState({ locationsArray: locationsData.data });
22     } catch (err) {
23       if (err.response) this.setState({ errorMessage: err.response.data.message })
24       else this.setState({ errorMessage: 'Network Error' })
25     }
26   }
27
28   /* Implemented - No Changes Required */
29   componentDidMount() {
30     this.getAllLocations();
31   }
32 }
```

Ln 24, Col 55 SyntaxError: 4 UTS-8 LF JavaScript 09:0



Mozilla Firefox




```
JS App.js JS Locations.js X JS FarmVisit.js
Frontend > src > components > JS Locations.js > Locations > render
25     }
26   }
27
28   /* Implemented - No Changes Required */
29   componentDidMount() {
30     this.getAllLocations();
31   }
32
33   /* To Be Implemented */
34   handleClick = (location) => {
35     this.setState({redirectStatus:true})
36     this.setState({selectedLocationName:location.locationName})
37   }
38   /*
39    It should set the state redirectStatus to true
40    and selectedLocationName as the locationName of location Object recieved as param
41  */
42
43   /* To Be Modified - iterate over the locationsArray state to display the location cards */
44   displayLocations = () => {
45     return (
46       this.state.locationsArray.map((location)=>{
47         <div className="col-md-3" key="SOME KEY">
48           <div className="card shadow">
49             <img src={location.locationImagePath} alt={location.id}
50               style={{ height: '200px' }} className="card-img-top" />
51             /* Add locationImagePath as source and id as the alternate text for the
52             /* <img style={{ height: '200px' }} className="card-img-top" /> */
53             <div className="card-body gradient-buttons text-center">
54               <h4 className="text" name="locationName">
55                 {location.locationName}
56                 /* Display the locationName of the location here */
57             </h4>
58           </div>
59         </div>
60       )
61     )
62   }
63 }
64
65 export default Locations;
```

```
JS App.js JS Locations.js X JS FarmVisit.js
Frontend > src > components > JS Locations.js > Locations > render
38      It should set the state redirectStatus to true
39      and selectedLocationName as the locationName of location Object recieved as param
40      */
41  }
42
43  /* To Be Modified - iterate over the locationsArray state to display the location cards */
44  displayLocations = () => {
45    return (
46      this.state.locationsArray.map((location)=>{
47        <div className="col-md-3" key="SOME KEY">
48          <div className="card shadow">
49            <img src={location.locationImagePath} alt={location.id}
50              style={{ height: '200px' }} className="card-img-top" />
51            /* Add locationImagePath as source and id as the alternate text for the
52              /* <img style={{ height: '200px' }} className="card-img-top" /> */
53            <div className="card-body gradient-buttons text-center">
54              <h4 className="text" name="locationName">
55                {location.locationName}
56                /* Display the locationName of the location here */
57              </h4>
58              /* Invoke handleClick method by passing proper location Object when I
59              <button name="bookVisit" className="btn btn-primary"
60                onClick={()=> this.handleClick(location)}>
61
62                Book Visit
63              </button>
64            </div>
65          </div>
66        </div>
67      )
68    )
69  }
```



```
69  
70 }  
71  
72 /* To Be Modified */  
73 render() {  
74   let { redirectStatus } = this.state;  
75   if (redirectStatus) {  
76     var redirect=<Redirect to={"/farmVisit/"+this.state.selectedLocationName}/>  
77     /* Redirect to /farmVisit/:locationName by passing proper locationName as the route */  
78     return redirect  
79   }  
80  
81   return (  
82     <div className="container-fluid mt-5">  
83       {this.state.locationsArray&&this.state.locationsArray.length>0  
84         ?this.displayLocations():this.state.errorMessage}  
85       {  
86         /*  
87          if locationsArray has location Objects - invoke the displayLocations  
88          else render the errorMessage  
89         */  
90       }  
91     </div>  
92   )  
93 }  
94  
95
```



JS App.js

JS Locations.js

JS FarmVisit.js x

```
Frontend > src > components > JS FarmVisit.js > FarmVisit > handleChange
1 import React, { Component } from "react";
2 import axios from "axios";
3
4 const url = "http://localhost:4000/bookings";
5
6 class FarmVisit extends Component {
7   constructor(props) {
8     super(props);
9     this.state = {
10       bookVisitForm: {
11         city: this.props.match.params.locationName,
12         visitDate: "",
13         phoneNumber: "",
14         customerName: ""
15       },
16       bookVisitFormErrorMessage: {
17         city: "",
18         visitDate: "",
19         phoneNumber: "",
20         customerName: ""
21       },
22       bookVisitFormValid: {
23         city: false,
24         visitDate: false,
25         customerName: false,
26         phoneNumber: false,
27         buttonActive: false
28       },
29       successMessage: "",
30       errorMessage: ""
31     };
32   }
```

Ln 43, Col 10 | Spaces: 2 | UTF-8 | LF | JavaScript

Mozilla Firefox



Frontend > src > components > JS FarmVisit.js > FarmVisit > handleChange

```
27     buttonActive: false
28   },
29   successMessage: "",
30   errorMessage: ""
31 }
32 }
33
34 /* To Be Implemented */
35 handleChange = (event) => {
36   const name= event.target.name
37   const value= event.target.value;
38   const newState={
39     bookVisitForm:{
40       ...this.state.bookVisitForm,[name]:value
41     }
42   }
43   this.setState(newState);
44   this.validateField(name,value);
45
46
47   /*
48    | should use the name and value from the event object
49    | and set the recieved value to the corresponding bookVisitForm state property
50    */
51 }
52
53 /* To Be Modified */
54 validateField = (fieldName, value) => {
55   let bookVisitFormErrorMessage = this.state.bookVisitFormErrorMessage;
56   let bookVisitFormValid = this.state.bookVisitFormValid;
57
58   switch (fieldName) {
59     case "city":
```

Mozilla Firefox




```
JS App.js JS Locations.js JS FarmVisit.js x
Frontend > src > components > JS FarmVisit.js > FarmVisit > handleChange
51 }
52
53 /* To Be Modified */
54 validateField = (fieldName, value) => {
55   let bookVisitFormErrorMessage = this.state.bookVisitFormErrorMessage;
56   let bookVisitFormValid = this.state.bookVisitFormValid;
57
58   switch (fieldName) {
59     case "city":
60       if (value === "") {
61         bookVisitFormErrorMessage.city = "Field Required"
62         bookVisitFormValid.city = false
63       }
64       else {
65         bookVisitFormErrorMessage.city = ""
66         bookVisitFormValid.city = true
67       }
68       break;
69     case "customerName":
70       const nameRegex=/b\w+\b(?:\b\w+\b){2}/
71       if(value===""){
72         bookVisitFormErrorMessage.customerName="Field Required"
73         bookVisitFormValid.customerName=false;
74       }
75       else if(!value.match(nameRegex)){
76         bookVisitFormErrorMessage.customerName="Name should contain two words and each word
77         bookVisitFormValid.customerName=false;
78       }
79       else{
80         bookVisitFormErrorMessage.customerName=""
81         bookVisitFormValid.customerName=true;
82       }

```

```
JS App.js JS Locations.js JS FarmVisit.js x
Frontend > src > components > JS FarmVisit.js > FarmVisit > handleChange
64   else {
65     bookVisitFormErrorMessage.city = ""
66     bookVisitFormValid.city = true
67   }
68   break;
69   case "customerName":
70     const nameRegex=/b\w+\b(?:\b\w+\b){2}/
71     if(value===""){
72       bookVisitFormErrorMessage.customerName="Field Required"
73       bookVisitFormValid.customerName=false;
74     }
75     else if(!value.match(nameRegex)){
76       bookVisitFormErrorMessage.customerName="Name should contain two words and each word
77       bookVisitFormValid.customerName=false;
78     }
79     else{
80       bookVisitFormErrorMessage.customerName=""
81       bookVisitFormValid.customerName=true;
82     }
83
84
85     /*
86     1. if customerName field is empty it should set the error message as "Field Required"
87     2. if customerName field does not contain two words with atleast 3 characters each
88     3. for valid input it should reset the error message appropriately
89     */
90     break;
91
92     case "visitDate":
93       let date=new Date(value);
94       let today=new Date();
95       if(value===""){

```



```
83
84
85  /*
86   1. if customerName field is empty it should set the error message as "Field Required"
87   2. if customerName field does not contain two words with atleast 3 characters each
88   3. for valid input it should reset the error message appropriately
89  */
90  break;
91
92  case "visitDate":
93    let date=new Date(value);
94    let today=new Date();
95    if(value===""){
96      bookVisitFormErrorMessage.visitDate="Field Required"
97      bookVisitFormValid.visitDate=false;
98    }
99    }else if(value<today){
100      bookVisitFormErrorMessage.visitDate="Date of visit should be after todays date"
101      bookVisitFormValid.visitDate=false;
102    }
103    }else{
104      bookVisitFormErrorMessage.visitDate=""
105      bookVisitFormValid.visitDate=true;
106    }
107  /*
108   1. if visitDate field is empty it should set the error message as "Field Required"
109   2. if visitDate field is not a future date, it should set the error message as "Date of visit should be after todays date"
110   3. for valid input it should reset the error message appropriately
111  */
112  break;
113  case "phoneNumber":
114    const numRegex=/[0-9]{1}[0-9]{9}/
```


Frontend > src > components > JS FarmVisit.js > FarmVisit > handleChange

```

104     bookVisitFormErrorMessage.visitDate=""
105     bookVisitFormValid.visitDate=true;
106 }
107 /*
108     1. if visitDate field is empty it should set the error message as "Field Required"
109     2. if visitDate field is not a future date, it should set the error message as "D
110     3. for valid input it should reset the error message appropriately
111 */
112     break;
113 case "phoneNumber":
114     const numRegex=/[6-9]{1}[0-9]{9}/
115     if(value===""){
116         bookVisitFormErrorMessage.phoneNumber="Field Required"
117         bookVisitFormValid.phoneNumber=false;
118     }else if(!value.match(numRegex)){
119         bookVisitFormErrorMessage.phoneNumber="Phone number should be of 10 digits"
120         bookVisitFormValid.phoneNumber=false;
121     }else{
122         bookVisitFormErrorMessage.phoneNumber=""
123         bookVisitFormValid.phoneNumber=true;
124     }
125 /*
126     1. if phoneNumber field is empty it should set the error message as "Field Required"
127     2. if phoneNumber field has less than or greater than 10 digits set the error message
128     3. for valid input it should reset the error message appropriately
129 */
130     break;
131 default:
132     break;
133 }
134 }
135

```

Ln 43, Col 10 Spaces: 2 UTF-8 LF JavaScript

09:01

```

167     2. should reset successMessage and errorMessage state as ""
168     3. should make a POST request to http://localhost:4000/bookings with bookVisitForm as the
169     4. in success case, it should set successMessage as Your Visit is successfully booked f
170     5. in error case, it should set errorMessage as 'Something went wrong'
171 */
172 }
173 /* To Be Modified */
174 render() {
175     return (
176         <div className="row">
177             <div className="col-md-6 offset-md-3">
178                 <br />
179                 <div className="card">
180                     <div className="card-header text-center bg-success">
181                         <h3><label>Farm visit</label></h3>
182                     </div>
183                     <div className="card-body">
184                         <form onSubmit={this.submitVisit}>
185                             <div className="form-group">
186                                 <label htmlFor="city">
187                                     City
188                                 </label>
189                                 <input type="text"
190                                     name="city"
191                                     className="form-control"
192                                     id="city"
193                                     value={this.state.bookVisitForm.city}
194                                     onChange={this.handleChange}
195                                     disabled/>
196                                 <span id="cityError"
197                                     className="alert alert-danger">
198

```

Ln 43, Col 10 Spaces: 2 UTF-8 LF JavaScript

09:01

Execute



JS App.js

JS Locations.js

JS FarmVisit.js x



Frontend > src > components > JS FarmVisit.js > FarmVisit > handleChange

```
138     this.setState({
139       bookVisitFormErrorMessage: bookVisitFormErrorMessage,
140       bookVisitFormValid: bookVisitFormValid,
141       successMessage: "", errorMessage: ""
142     });
143   }
144 }
145
146 /* To Be Implemented */
147 submitVisit = (event) => {
148   event.preventDefault()
149   this.state.successMessage=""
150   this.state.errorMessage=""
151   var data={
152     city:this.props.match.params.locationName,
153     visitDate:this.state.bookVisitForm.visitDate,
154     phoneNumber:this.state.bookVisitForm.phoneNumber,
155     customerName:this.state.bookVisitForm.customerName,
156   }
157   axios.post("http://localhost:4000/bookings",data)
158   .then(()=>{
159     this.setState({successMessage:"Your Visit is successfully booked for <<response visitDa
160   }).catch(()=>{
161     this.setState({errorMessage:"Something went wrong"})
162   })
163
164
165   /*
166   1. should prevent the page reload on form submission
167   2. should reset successMessage and errorMessage state as ""
168   3. should make a POST request to http://localhost:4000/bookings with bookVisitForm as t
169   4. in success case, it should set successMessage as Your Visit is successfully booked f
```

Ln 43, Col 18 Syntax: 2 UFF-8 LF JavaScript

09:02 Execu

Mozilla Firefox



Frontend > src > components > JS FarmVisit.js > FarmVisit > handleChange

```

185 <form onSubmit={this.submitVisit}>
186 <div className="form-group">
187   <label htmlFor="city">
188     City
189   </label>
190   <input type="text"
191     name="city"
192     className="form-control"
193     id="city"
194     value={this.state.bookVisitForm.city}
195     onChange={this.handleChange}
196     disabled/>
197   <span id="cityError"
198     className="alert alert-danger">
199     {this.state.bookVisitForm.errorMessage.city}
200   </span>
201 </div>
202 <div className="form-group">
203   <label htmlFor="customerName">
204     Customer Name
205   </label>
206   <input type="text"
207     name="customerName"
208     className="form-control"
209     id="customerName"
210     value={this.state.bookVisitForm.customerName}
211     onChange={this.handleChange}
212   />
213   <span id="customerNameError"
214     className="text text-danger">
215     {this.state.bookVisitForm.errorMessage.customerName}
216   </span>
217 </div>

```



```
JS App.js JS Locations.js JS FarmVisit.js x
Frontend > src > components > JS FarmVisit.js > FarmVisit > handleChange
209     name="customerName"
210     className="form-control"
211     id="customerName"
212     value={this.state.bookVisitForm.customerName}
213     onChange={this.handleChange}
214   />
215   <span id="customerNameError"
216     className="text text-danger">
217     {this.state.bookVisitFormErrorMessage.customerName}
218   </span>
219
220
221 </div>
222 <div className="form-group">
223   <label htmlFor="visitDate">
224     Visit Date
225   </label>
226   <input type="date"
227     name="visitDate"
228     className="form-control"
229     id="visitDate"
230     value={this.state.bookVisitForm.visitDate}
231     onChange={this.handleChange}
232   />
233   <span id="visitDateError"
234     className="alert alert-danger">
235     {this.state.bookVisitFormErrorMessage.visitDate}
236   </span>
237
238 </div>
239 <div className="form-group">
240
```

```
Frontend > src > components > JS FarmVisit.js > FarmVisit > handleChange
230     value={this.state.bookVisitForm.visitDate}
231     onChange={this.handleChange}
232   />
233   <span id="visitDateError"
234     className="alert alert-danger">
235     {this.state.bookVisitFormErrorMessage.visitDate}
236   </span>
237
238 </div>
239 <div className="form-group">
240   <label htmlFor="phoneNumber">
241     Phone Number
242   </label>
243   <input type="number"
244     name="phoneNumber"
245     className="form-control"
246     id="phoneNumber"
247     value={this.state.bookVisitForm.phoneNumber}
248     onChange={this.handleChange}
249   />
250   <span id="phoneNumberError"
251     className="alert alert-danger">
252     {this.state.bookVisitFormErrorMessage.phoneNumber}
253   </span>
254
255 </div>
256
257 <button type="submit"
258   className="btn btn-success"
259
```



```
259
260
261 <button type="submit"
262       className="btn btn-success"
263       id="bookVisitBtn"
264       disabled={!this.state.bookVisitFormValid.buttonActive}>
265   | Book Visit
266 </button>
```

```
267
268
269
270
271
272
273 </form>
274 {this.state.successMessage?
275   (<span id="successMessage"
276       className="text text-success">
277         {this.state.successMessage}
278       </span>):("")}
279
280 {this.state.errorMessage?
281   (<span id="errorMessage"
282       className="text text-danger">
283         {this.state.errorMessage}
284       </span>):("")}
285
286
```

```
287
288 /*
```

1. create the form with the fields as shown in QP
2. All the form fields should be bound to the corresponding bo

Ln 43, Col 10 Spaces: 2


```

Frontend > src > components
31 }
32 }
33
34 /* To Be Imple
35 handleChange =
36   const name= e
37   const value=
38   const newstat
39   bookVisitFo
40   ...this.s
41 }
42 }
43 this.setState
44 this.validate
45
46
47 /*
48   should up
49   and set
50 */
51 }
52
53 /* To Be Modifi
54 validateField =
55   let bookVisit
56   let bookVisit
57
58   switch (field
59     case "city"
60     if (value
61       bookVis
62       bookVis

```

phoneNumber field			
FarmVisit Component - customerName field	2	0	2
FarmVisit Component - visitDate field	2	0	2
FarmVisit Component - city field	3	0	3
Locations Component - Testing render method	4	0	4
Locations Component - Testing displayLocations method	5	0	5
Locations Component - Testing handleClick Method	2	0	2
Testing Application Routing	3	0	3
Total	50	4	54

Close

master* 0 0 0

FarmVisit.js - Pr...

Mozilla Firefox

Execute