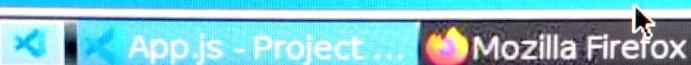


```
    1. It should redirect to /locations page for default URL i.e. http://l
    2. It should load FarmVisit component for /farmVisit/:locationName
    3. It should load Locations component for /locations
    4. It should redirect to /locations page for any invalid URL e.g. http
 */
<Route exact path="/" component={Locations}/>
<Route exact path="/farmVisit/:locationName" component={FarmVisit}/>
<Route exact path="/locations" component={Locations}/>
<Redirect to="/" />
</Switch>
</Router>
</>
</div>
);

```

Ln 25, Col 11 Spaces: 2 UTF-8



locations.js JSON JS App.js

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

```
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   }
30
31   handleFormSubmit(event) {
32     event.preventDefault();
33
34     const { city, visitDate, phoneNumber, customerName } = this.state.bookVisitForm;
35
36     if (city === "" || visitDate === "" || phoneNumber === "" || customerName === "") {
37       this.setState({
38         bookVisitFormErrorMessage: {
39           city: true,
40           visitDate: true,
41           phoneNumber: true,
42           customerName: true
43         }
44       });
45     } else {
46       axios
47         .post(url, {
48           locationName: city,
49           visitDate: visitDate,
50           phoneNumber: phoneNumber,
51           customerName: customerName
52         })
53         .then(response => {
54           console.log(response);
55         })
56         .catch(error => {
57           console.error(error);
58         });
59     }
60   }
61
62   render() {
63     const { bookVisitFormValid } = this.state;
64
65     return (
66       <div>
67         <h1>Farm Visit</h1>
68         <form>
69           <input type="text" value={bookVisitForm.city} />
70           <input type="date" value={bookVisitForm.visitDate} />
71           <input type="text" value={bookVisitForm.phoneNumber} />
72           <input type="text" value={bookVisitForm.customerName} />
73           <button type="submit" disabled={!bookVisitFormValid}>Book Visit</button>
74         </form>
75       </div>
76     );
77   }
78 }
```

```
JS FarmVisit.js X JS Locations.js (1) db.json JS App.js
Frontend > src > components > JS FarmVisit.js > FarmVisit > handleChange
35     handleChange = (event) => {
36         /*
37          * should use the name and value from the event object
38          * and set the received value to the corresponding bookVisitForm state property
39         */
40         const {name,value} = event.target
41         this.setState({
42             ...this.state.bookVisitForm
43             [name] = value
44         });
45         this.validateField(name,value)
46     }
47
48     /* To Be Modified */
49     validateField = (fieldName, value) => {
50         let bookVisitFormErrorMessage = this.state.bookVisitFormErrorMessage;
51         let bookVisitFormValid = this.state.bookVisitFormValid;
52
53         switch (fieldName) {
54             case "city":
55                 if (value === "") {
56                     bookVisitFormErrorMessage.city = "Field Required"
57                     bookVisitFormValid.city = false
58                 }
59                 else {
60                     bookVisitFormErrorMessage.city = ""
61                 }
62         }
63     }
64 }
```

```
Frontend > src > components > js FarmVisit.js >  FarmVisit >  handleChange
59     else {
60       bookVisitFormErrorMessage.city = ""
61       bookVisitFormValid.city = true
62     }
63   break;
64 case "customerName":
65 /*
66 1. if customerName field is empty it should set the error message as "Field Required"
67 2. if customerName field does not contain two words with atleast 3 characters each
68 3. for valid input it should reset the error message appropriately
69 */
70   const customerNameRegex = /^[A-Z][a-z]{3,}\s[A-Z][a-z]{3,}$/
71   if (value === "") {
72     bookVisitFormErrorMessage.customerName = "Field Required"
73     bookVisitFormValid.customerName = false
74   }
75   else if (!value.match(customerNameRegex)){
76     bookVisitFormErrorMessage.customerName = "Name should contain two words and each"
77     bookVisitFormValid.customerName = false
78   }
79   else {
80     bookVisitFormErrorMessage.customerName = ""
81     bookVisitFormValid.customerName = true
82   }
83   break;
84
85
```

```
Frontend>src>components>js FarmVisit.js > FarmVisit > handleChange
85 case "visitDate":  
86 /*  
87 1. if visitDate field is empty it should set the error  
88 2. if visitDate field is not a future date, it should set  
89 3. for valid input it should reset the error message app  
90 */  
91 const visitDate = new Date(value);  
92 const todayDate = new Date();  
93 if (value === "") {  
94 bookVisitFormErrorMessage.visitDate = "Field Required"  
95 bookVisitFormValid.visitDate = false  
96 }  
97 else if (visitDate < todayDate){  
98 bookVisitFormErrorMessage.visitDate = "Date of visit should be  
99 bookVisitFormValid.visitDate = false  
100 }  
101 else {  
102 bookVisitFormErrorMessage.visitDate = ""  
103 bookVisitFormValid.visitDate = true  
104 }  
105 break;  
106 case "phoneNumber":  
107 /*  
108 1. if phoneNumber field is empty it should set the error message as "Field  
109 2. if phoneNumber field has less than or greater than 10 digits set the error  
110 3. for valid input it should reset the error message app  
111 */
```

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

```
106 case "phoneNumber":  
107     /*  
108      1. if phoneNumber field is empty it should set the error message as "Field Required"  
109      2. if phoneNumber field has less than or greater than 10 digits set the error message  
110      3. for valid input it should reset the error message appropriately  
111     */  
112     const phoneNumberRegex = /^[0-9]{10}$/;  
113     if (value === "") {  
114         bookVisitFormErrorMessage.phoneNumber = "Field Required";  
115         bookVisitFormValid.phoneNumber = false;  
116     }  
117     else if (!value.match(phoneNumberRegex)){  
118         bookVisitFormErrorMessage.phoneNumber = "Phone number should be of 10 digits";  
119         bookVisitFormValid.phoneNumber = false;  
120     }  
121     else {  
122         bookVisitFormErrorMessage.phoneNumber = "";  
123         bookVisitFormValid.phoneNumber = true;  
124     }  
125     break;  
126     default:  
127     break;  
128 }  
129 bookVisitFormValid.buttonActive = bookVisitFormValid.customerName && bookVisitFormValid.phoneNumber;  
130  
131 this.setState({
```

```
Frontend > src > components > js FarmVisit.js > FarmVisit > handleChange
127      break;
128  }
129  bookVisitFormValid.buttonActive = bookVisitFormValid.customerName && bookVisitFormValid.pl
130
131  this.setState({
132    bookVisitFormErrorMessage: bookVisitFormErrorMessage,
133    bookVisitFormValid: bookVisitFormValid,
134    successMessage: "", errorMessage: ""
135  });
136
137 }
138
139 /* To Be Implemented */
140 submitVisit = (event) => {
141   /*
142     1. should prevent the page reload on form submission
143     2. should reset successMessage and errorMessage state as ""
144     3. should make a POST request to http://localhost:4000/bookings with bookVisitForm as ti
145     4. in success case, it should set successMessage as Your Visit is successfully booked fo
146     5. in error case, it should set errorMessage as 'Something went wrong'
147   */
148   event.preventDefault();
149   this.state.successMessage = ""
150   this.state.errorMessage = ""
151   axios.post("/http://localhost:4000/bookings", this.state.bookVisitForm)
152 }
```

```
139  /* To Be Implemented */
140  submitVisit = (event) => {
141    /*
142      1. should prevent the page reload on form submission
143      2. should reset successMessage and errorMessage state as ""
144      3. should make a POST request to http://localhost:4000/bookings with bookVisitForm as t
145      4. in success case, it should set successMessage as Your Visit is successfully booked fo
146      5. in error case, it should set errorMessage as 'Something went wrong'
147    */
148    event.preventDefault();
149    this.state.successMessage = ""
150    this.state.errorMessage = ""
151    axios.post("/http://localhost:4000/bookings", this.state.bookVisitForm)
152      .then((response)=>{
153        this.setState({
154          successMessage : `Your Visit is successfully booked for ${this.state.visitDate}`
155        })
156      })
157      .catch(()=>{
158        this.setState({
159          errorMessage : "Something went wrong"
160        })
161      })
162    }
163  }
164 }
```

```
165  /* To Be Modified */
166  render(){
167    return (
168      <div className="row">
169        <div className="col-md-6 offset-md-3">
170          <br />
171          <div className="card">
172            <div className="card-header text-center bg-success">
173              <h3><label>Farm visit</label></h3>
174            </div>
175            <div className="card-body">
176              /*
177                1. create the form with the fields as shown in QP
178                2. All the form fields should be bound to the corresponding bookVisitForm state
179                3. All the form fields should have proper name same as bookVisitForm state
180                4. city field should be disabled by default
181                5. customerName field should be of type text
182                6. customerName error message displaying tag should have id as customerNameError
183                7. visitDate field should be of type date
184                8. visitDate error message displaying tag should have id as visitDateError
185                9. phoneNumber field should be of type number
186                10. phoneNumber error message displaying tag should have id as phoneNumberError
187                11. Book Visit button should have id as bookVisitBtn
188                12. It should be disabled until buttonActive state property becomes true
189                13. display errorMessage and successMessage on form submission appropriate
190                both errors
```

```
189      11. It should be disabled until buttonActive state property becomes true
190      13. display errorMessage and successMessage on form submission appropriately
191      14. both errorMessage and successMessage should not appear together
192    */
193    <form onSubmit={this.submitVisit}>
194      <div className="form-group">
195        <input className="form-control"
196          type="text"
197          id="city"
198          name="city"
199          onChange={this.handleChange}
200          value={this.state.bookVisitForm.city}
201          disabled
202        />
203      </div>
204
205      <div className="form-group">
206        <label htmlFor="customerName">customerName</label>
207        <input className="form-control"
208          type="text"
209          id="customerName"
210          name="customerName"
211          onChange={this.handleChange}
212          value={this.state.bookVisitForm.customerName}
213        />
214        <span id="customerNameError" className="error">
```

```
212     onChange={this.handleChange}
213     value={this.state.bookVisitForm.customerName}
214   />
215   <span id="customerNameError" className="text text-danger">
216     {this.state.bookVisitFormErrorMessage.customerName}</span>
217   </div>
218
219   <div className="form-group">
220     <label htmlFor="visitDate">visitDate</label>
221     <input className="form-control"
222       type="date"
223       className="form-control"
224       id="visitDate"
225       name="visitDate"
226       onChange={this.handleChange}
227       value={this.state.bookVisitForm.visitDate}
228     />
229     <span id="visitDateError" className="text text-danger">
230       {this.state.bookVisitFormErrorMessage.visitDate}</span>
231   </div>
232
233   <div className="form-group">
234     <label htmlFor="phoneNumber">phoneNumber</label>
235     <input className="form-control"
236       type="number"
237       id="phoneNumber"
238       name="phoneNumber"
239     />
```

```
<input className="form-control"
type="number"
id="phoneNumber"
name="phoneNumber"
onChange={this.handleChange}
value={this.state.bookVisitForm.phoneNumber}
/>
<span id="phoneNumberError" className="text text-danger">
  {this.state.bookVisitFormErrorMessage.phoneNumber}</span>
</div>

<button id="bookVisitBtn" className="btn btn-success"
disabled={!this.state.bookVisitFormValid.buttonActive}>
  Book Visit
</button>

</form>
{this.state.successMessage ?
(<div id="successMessage" className="text text-success">
  {this.state.successMessage}</div>) : ("")}
{this.state.errorMessage ?
(<div id="errorMessage" className="text text-danger">
  {this.state.errorMessage}</div>) : ("")}
```

Frontend > src > components > 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 }
```

```
26
27
28     /* Implemented - No Changes Required */
29     componentDidMount() {
30         this.getAllLocations();
31     }
32
33     /* To Be Implemented */
34     handleClick = (location) => {
35         /*
36             It should set the state redirectStatus to true
37             and selectedLocationName as the locationName of location Object received as parameter
38         */
39         const newState =({
40             redirectStatus : true,
41             selectedLocationName : location.locationName
42         })
43         this.setState(newState)
44     }
45
46     /* To Be Modified - iterate over the locationsArray state to display the location cards */
47     displayLocations = () => {
48         return (
49             this.state.locationsArray.map((loc,id)=>
50                 <div className="col-md-3" key={"id"}>
51                     <div className="card shadow">
```

```
46  * To Be Modified - iterate over the locationsArray state to display the location cards */
47  isplayLocations = () => {
48    return (
49      this.state.locationsArray.map((loc,id)=>
50        <div className="col-md-3" key={"id"}>
51          <div className="card shadow">
52            <img src={loc.locationImagePath} alt={loc.id}
53              style={{ height: '200px' }} className="card-img-top"></img>
54            /* Add locationImagePath as source and id as the alternate text for the location */
55            /* <img style={{ height: '200px' }} className="card-img-top" /> */
56            <div className="card-body gradient-buttons text-center">
57              <h4 className="text" name="locationName">
58                /* Display the locationName of the location here */
59                {loc.locationName}
60              </h4>
61              /* Invoke handleClick method by passing proper location Object when Book Visit button is clicked */
62              <button name="bookVisit" className="btn btn-primary"
63                onClick={()=>{this.handleClick(loc.location)}}>
64                Book Visit
65              </button>
66            </div>
67          </div>
68        </div>
69      )
70    )
71  * To Be Modified - handle Click event
72 }
```

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