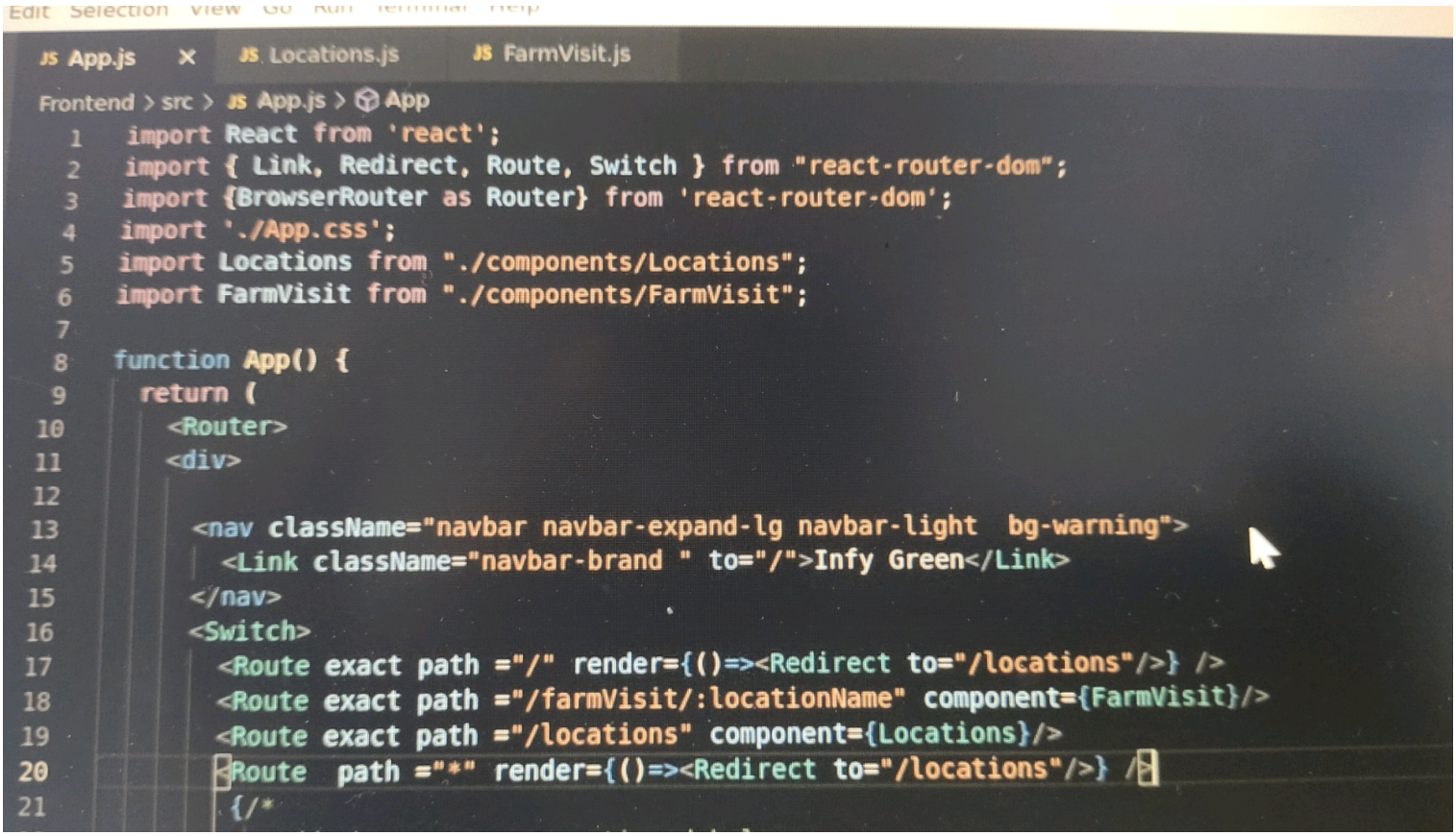
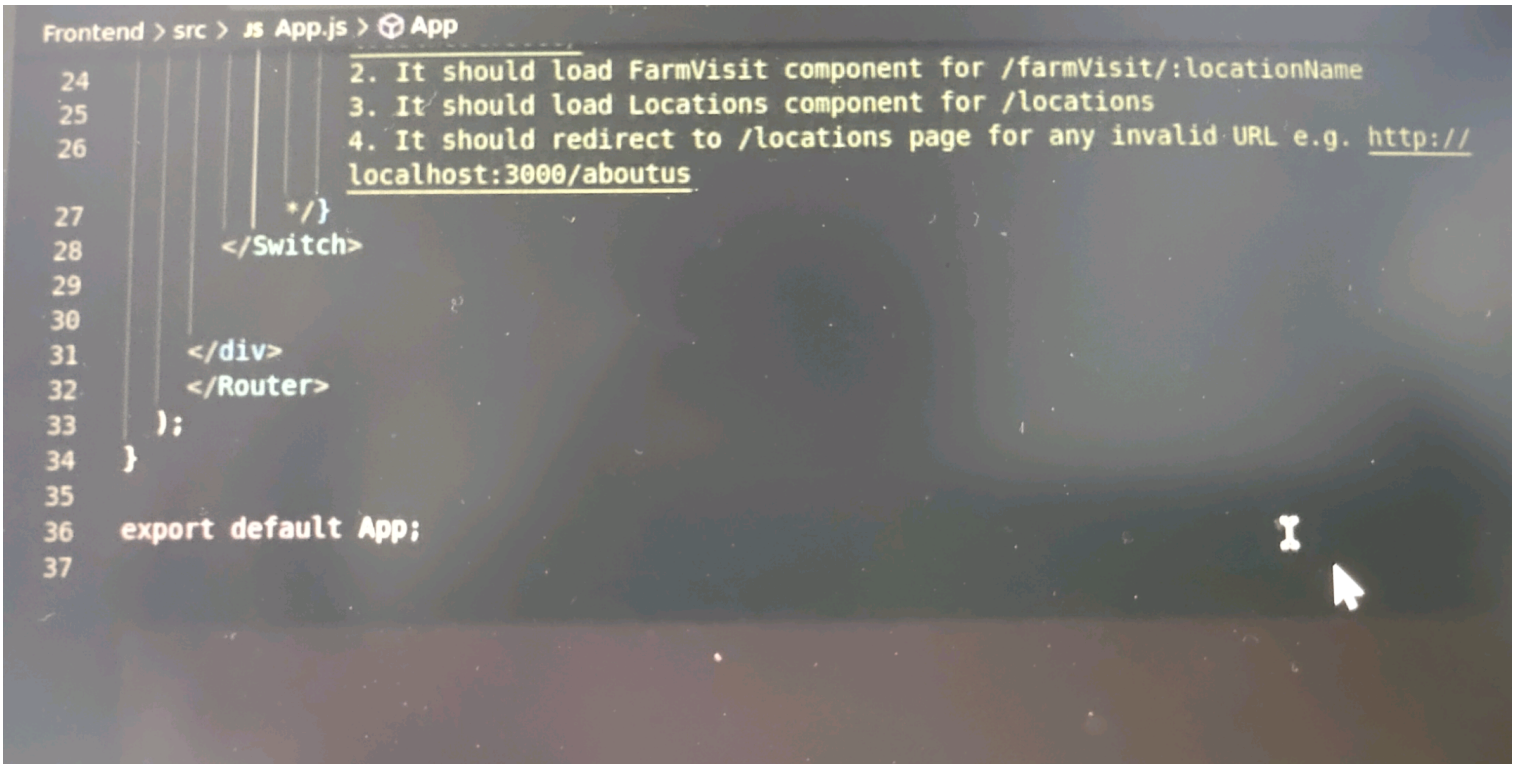
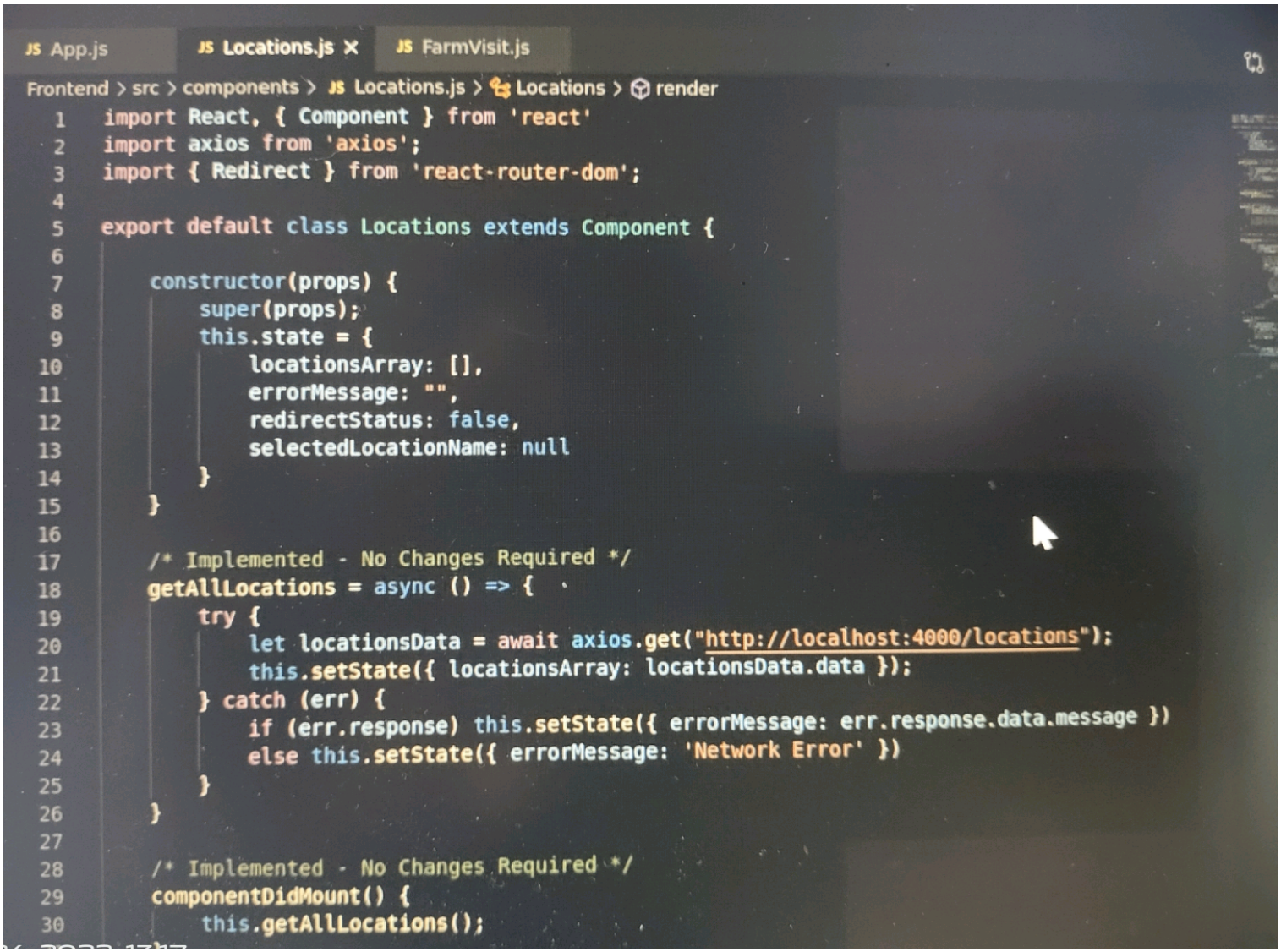
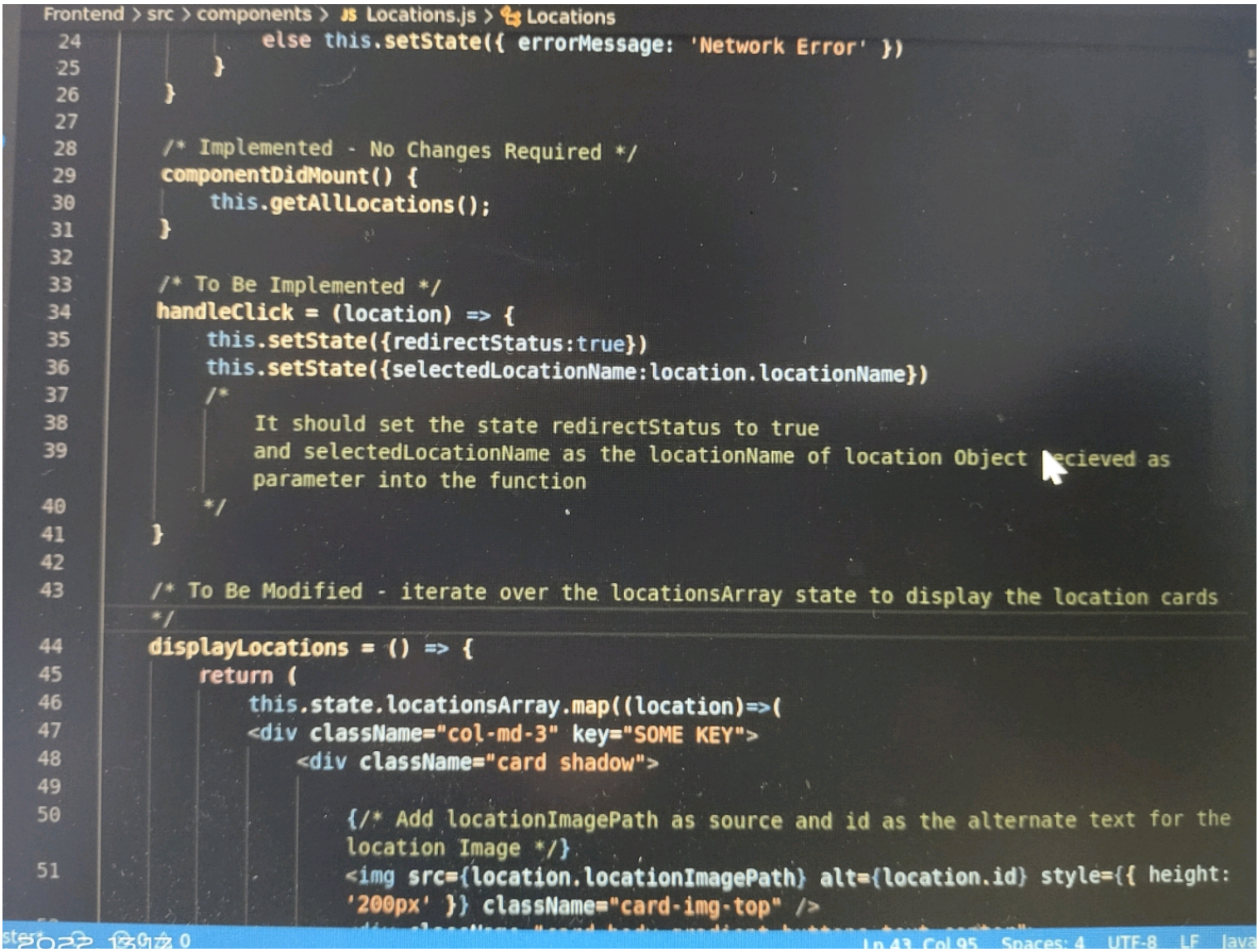
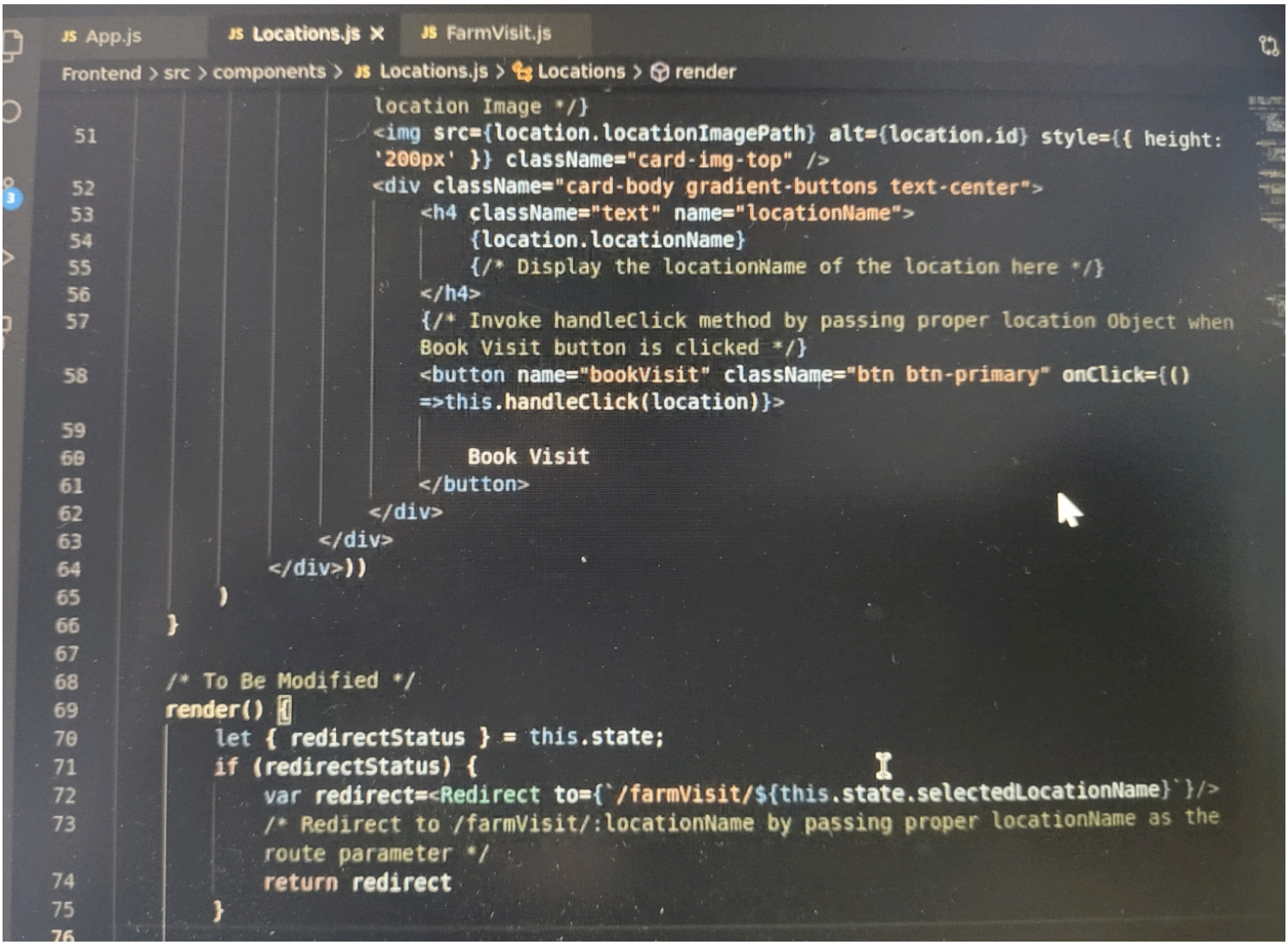

Edit Selection View Go Run Terminal Help
App.js
Locations.js
FarmVisit.js
Frontend > src > App.js > App
1 import React from 'react';
2 import { Link, Redirect, Route, Switch } from "react-router-dom";
3 import {BrowserRouter as Router} from 'react-router-dom';
4 import './App.css';
5 import Locations from "./components/Locations";
6 import FarmVisit from "./components/FarmVisit";
7
8 function App() {
9 return (
10 <Router>
11
12
13 <nav className="navbar navbar-expand-lg navbar-light bg-warning">
14 <Link className="navbar-brand " to="/">Infy Green</Link>
15 </nav>
16 <Switch>
17 <Route exact path ="/" render={()=><Redirect to="/locations"/>} />
18 <Route exact path ="/farmVisit/:locationName" component={FarmVisit}/>
19 <Route exact path ="/locations" component={Locations}/>
20 <Route path ="*" render={()=><Redirect to="/locations"/>} />
21 {/*


Frontend > src > JS App.js > App
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://localhost:3000/aboutus
*/}
</Switch>

</Router>
);
}
export default App;


App.js
Locations.js
FarmVisit.js
Frontend > src > components > Locations.js > Locations > render
import React, { Component } from 'react'
import axios from 'axios';
import { Redirect } from 'react-router-dom';
export default class Locations extends Component {
constructor(props) {
super(props);
this.state = {
locationsArray: [],
errorMessage: "",
redirectStatus: false,
selectedLocationName: null
}
}
/* Implemented - No Changes Required */
getAllLocations = async () => {
try {
let locationsData = await axios.get("http://localhost:4000/locations");
this.setState({ locationsArray: locationsData.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.getAllLocations();

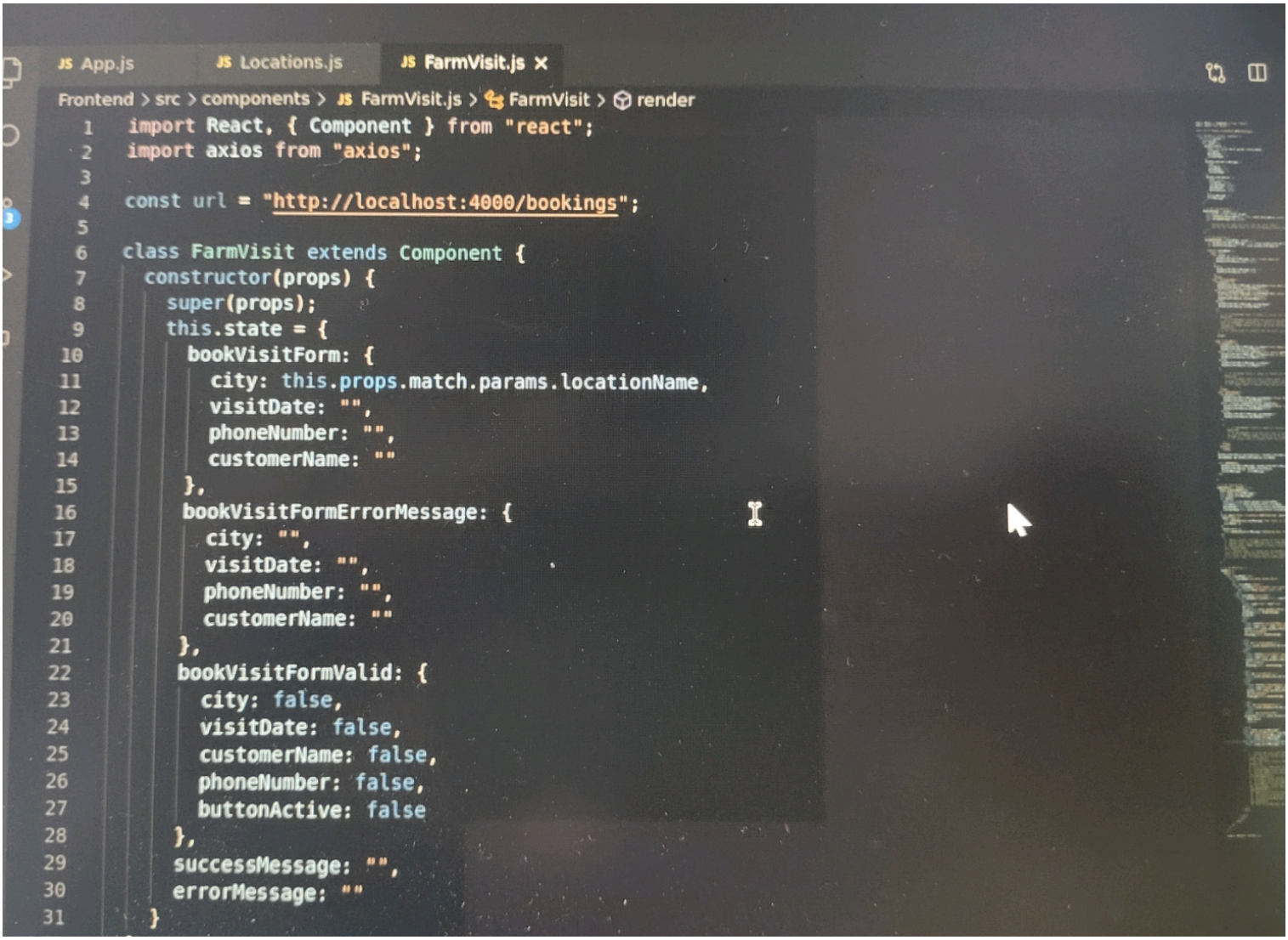

Frontend > src > components > JS Locations.js > Locations
24 else this.setState({ errorMessage: 'Network Error' })
25 }
26 }
27
28 /* Implemented - No Changes Required */
29 componentDidMount() {
30 this.getAllLocations();
31 }
32
33 /* To Be Implemented */
34 handleClick = (location) => {
35 this.setState({redirectStatus:true})
36 this.setState({selectedLocationName:location.locationName})
37 /*
38 It should set the state redirectStatus to true
39 and selectedLocationName as the locationName of location Object recieved as
parameter into the function
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
48
49
50 {/* Add locationImagePath as source and id as the alternate text for the
location Image */}
51 <img src={location.locationImagePath} alt={location.id} style={{ height:
'200px' }} className="card-img-top" />

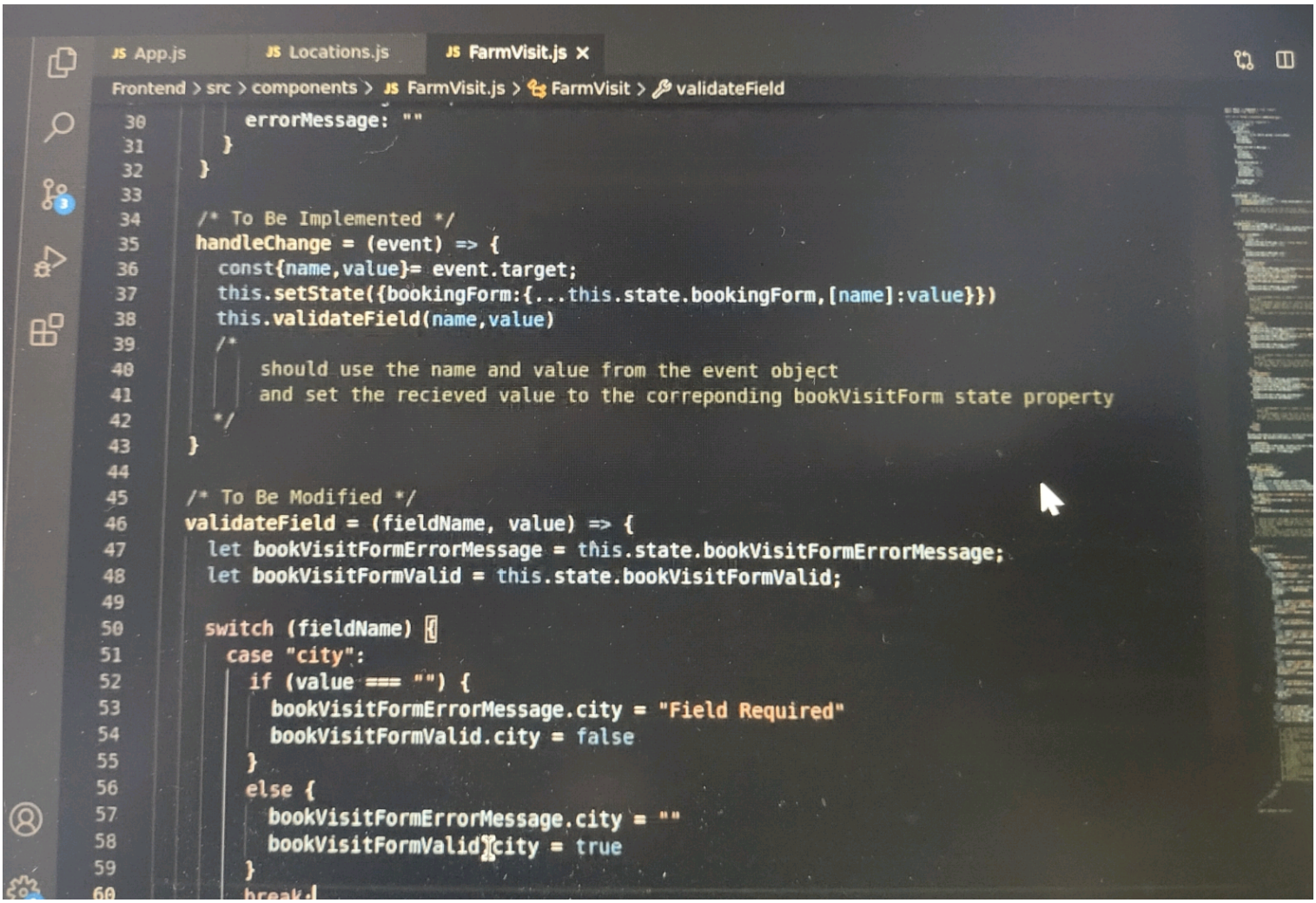

App.js
Locations.js
FarmVisit.js
Frontend > src > components > Locations.js > Locations > render
location Image */}
<img src={location.locationImagePath} alt={location.id} style={{ height:
'200px' }} className="card-img-top" />

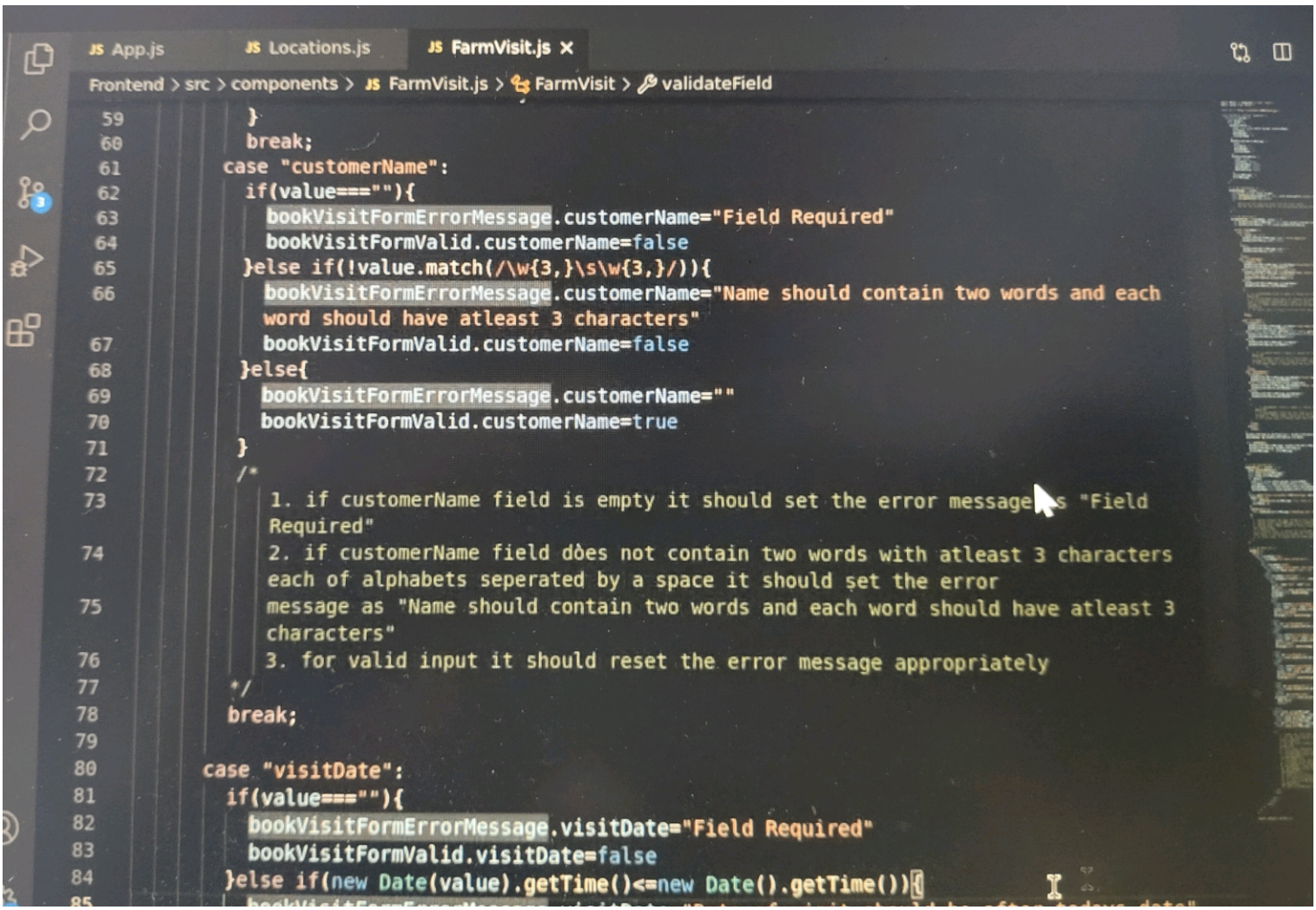
<h4 className="text" name="locationName">
{location.locationName}
{/* Display the locationName of the location here */}
</h4>
{/* Invoke handleClick method by passing proper location Object when
Book Visit button is clicked */}
<button name="bookVisit" className="btn btn-primary" onClick={()
=>this.handleClick(location)}>
Book Visit
</button>

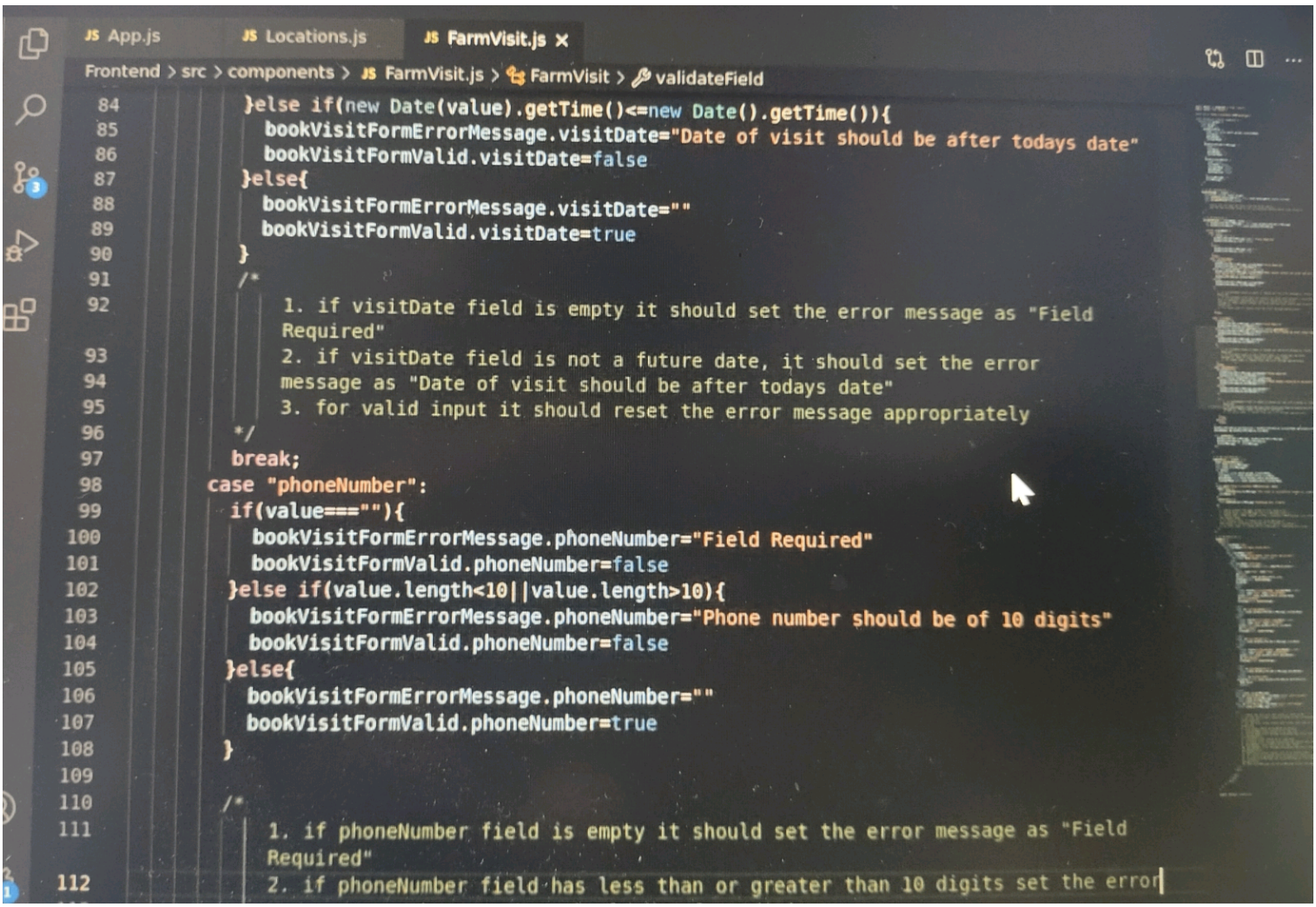
))
)
}
/* To Be Modified */
render() {
let { redirectStatus } = this.state;
if (redirectStatus) {
var redirect=<Redirect to={`/farmVisit/${this.state.selectedLocationName}`}/>
/* Redirect to /farmVisit/:locationName by passing proper locationName as the
route parameter */
return redirect
}

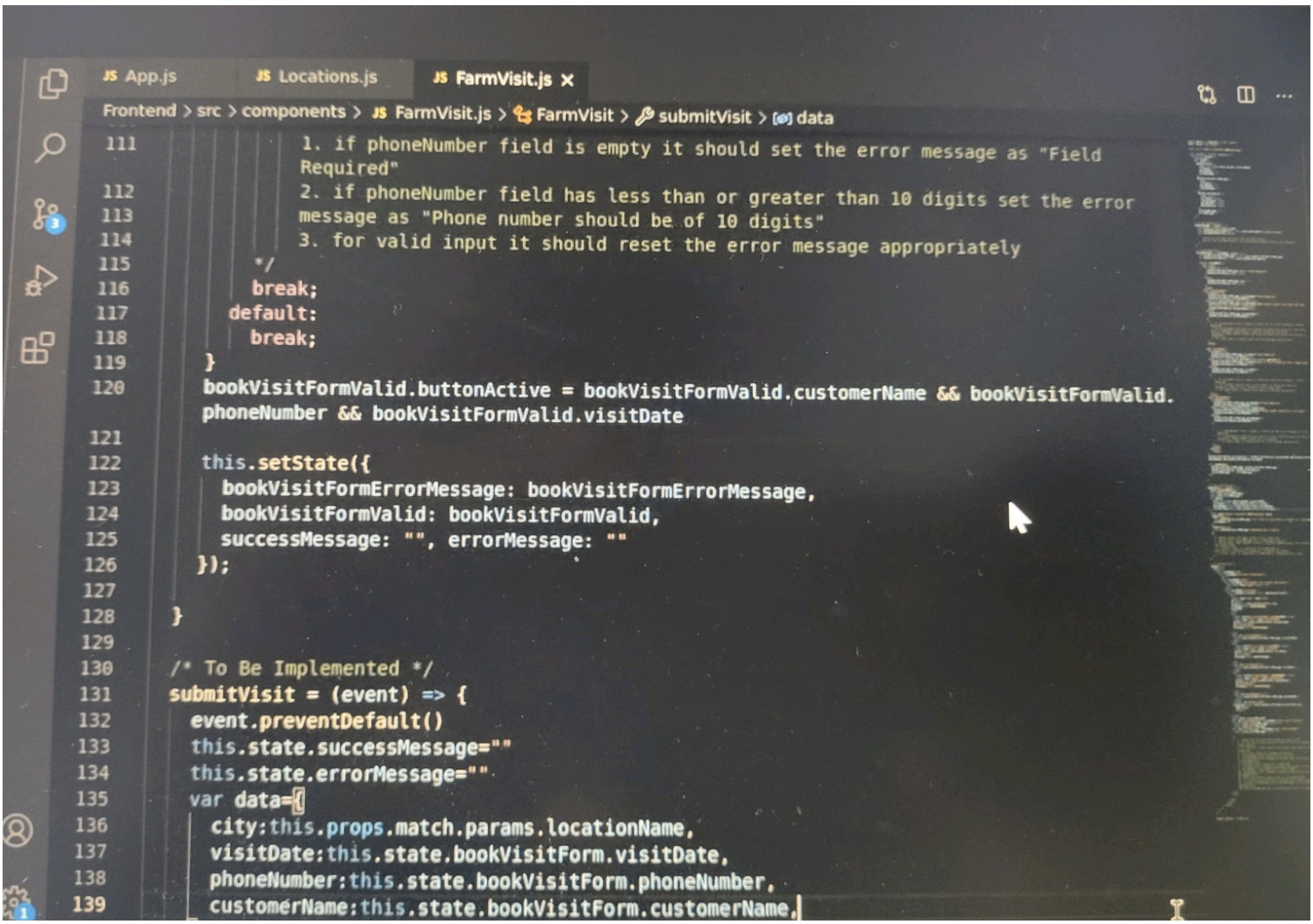

App.js
Locations.js
FarmVisit.js
Frontend > src > components > Locations.js > ...
63
64))
65)
66 }
67
68 /* To Be Modified */
69 render() {
70 let { redirectStatus } = this.state;
71 if (redirectStatus) {
72 var redirect=<Redirect to={`/farmVisit/${this.state.selectedLocationName}`}/>
73 /* Redirect to /farmVisit/:locationName by passing proper locationName as the route parameter */
74 return redirect
75 }
76
77 return (
78
79 {this.state.locationsArray.length>0?this.displayLocations():this.state.errorMessage}
80 {
81 /*
82 if locationsArray has location Objects - invoke the displayLocations method
83 else render the errorMessage
84 */
85 }
86
87)
88 }
89 }

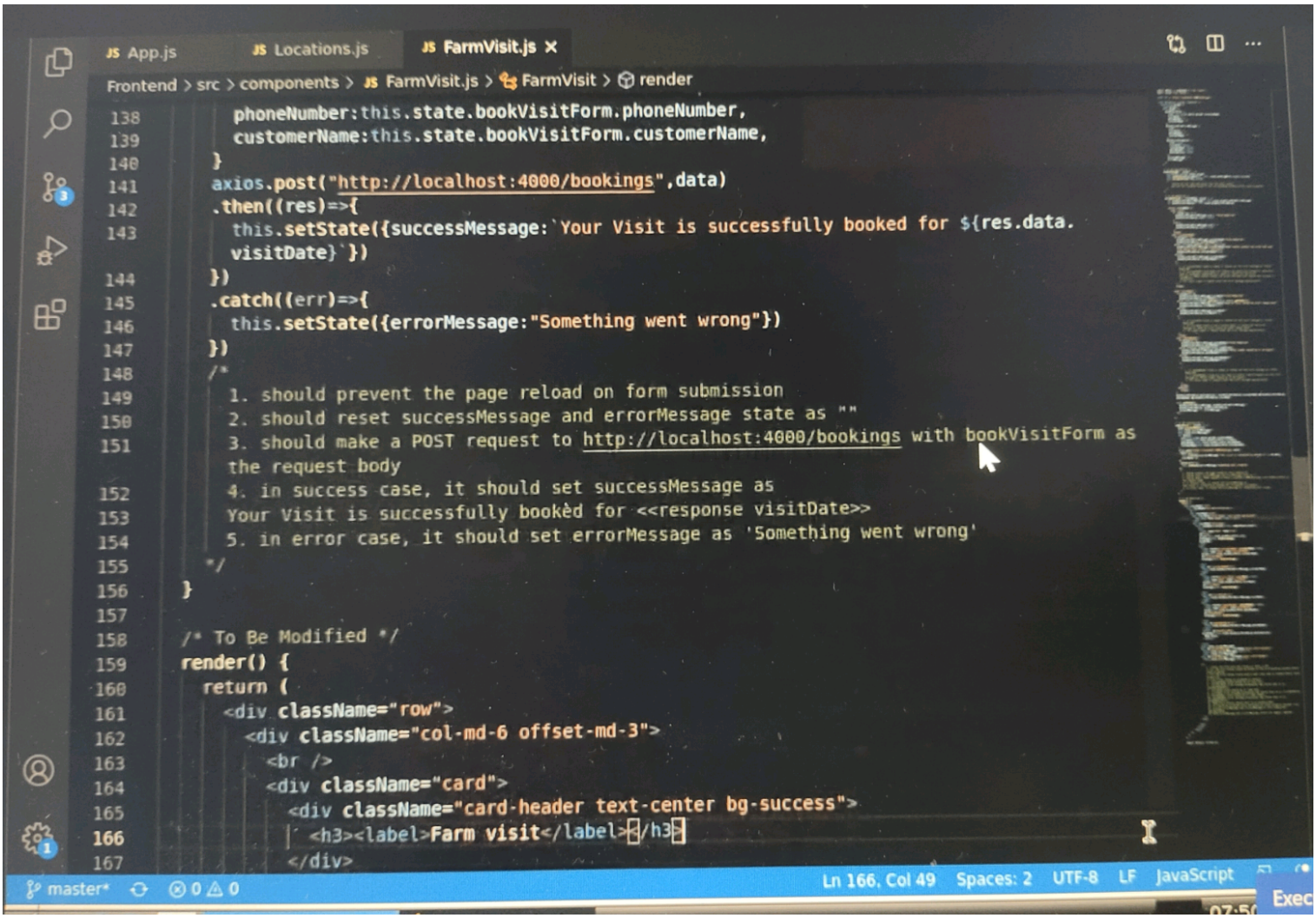

App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > render
import React, { Component } from "react";
import axios from "axios";
const url = "http://localhost:4000/bookings";
class FarmVisit extends Component {
constructor(props) {
super(props);
this.state = {
bookVisitForm: {
city: this.props.match.params.locationName,
visitDate: "",
phoneNumber: "",
customerName: ""
},
bookVisitFormErrorMessage: {
city: "",
visitDate: "",
phoneNumber: "",
customerName: ""
},
bookVisitFormValid: {
city: false,
visitDate: false,
customerName: false,
phoneNumber: false,
buttonActive: false
},
successMessage: "",
errorMessage: ""
}


App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > validateField
errorMessage: ""
}
}
/* To Be Implemented */
handleChange = (event) => {
const{name,value}= event.target;
this.setState({bookingForm:{...this.state.bookingForm,[name]:value}})
this.validateField(name,value)
/*
should use the name and value from the event object
and set the recieved value to the correponding bookVisitForm state property
*/
}
/* To Be Modified */
validateField = (fieldName, value) => {
let bookVisitFormErrorMessage = this.state.bookVisitFormErrorMessage;
let bookVisitFormValid = this.state.bookVisitFormValid;
switch (fieldName) {
case "city":
if (value === "") {
bookVisitFormErrorMessage.city = "Field Required"
bookVisitFormValid.city = false
}
else {
bookVisitFormErrorMessage.city = ""
bookVisitFormValid.city = true
}
break;


App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > validateField
59 }
60 break;
61 case "customerName":
62 if(value===""){
63 bookVisitFormErrorMessage.customerName="Field Required"
64 bookVisitFormValid.customerName=false
65 }else if(!value.match(/\w{3,}\s\w{3,}/)){
66 bookVisitFormErrorMessage.customerName="Name should contain two words and each word should have atleast 3 characters"
67 bookVisitFormValid.customerName=false
68 }else{
69 bookVisitFormErrorMessage.customerName=""
70 bookVisitFormValid.customerName=true
71 }
72 /*
73 1. if customerName field is empty it should set the error message as "Field Required"
74 2. if customerName field does not contain two words with atleast 3 characters each of alphabets seperated by a space it should set the error
75 message as "Name should contain two words and each word should have atleast 3 characters"
76 3. for valid input it should reset the error message appropriately
77 */
78 break;
79
80 case "visitDate":
81 if(value===""){
82 bookVisitFormErrorMessage.visitDate="Field Required"
83 bookVisitFormValid.visitDate=false
84 }else if(new Date(value).getTime()<=new Date().getTime()){

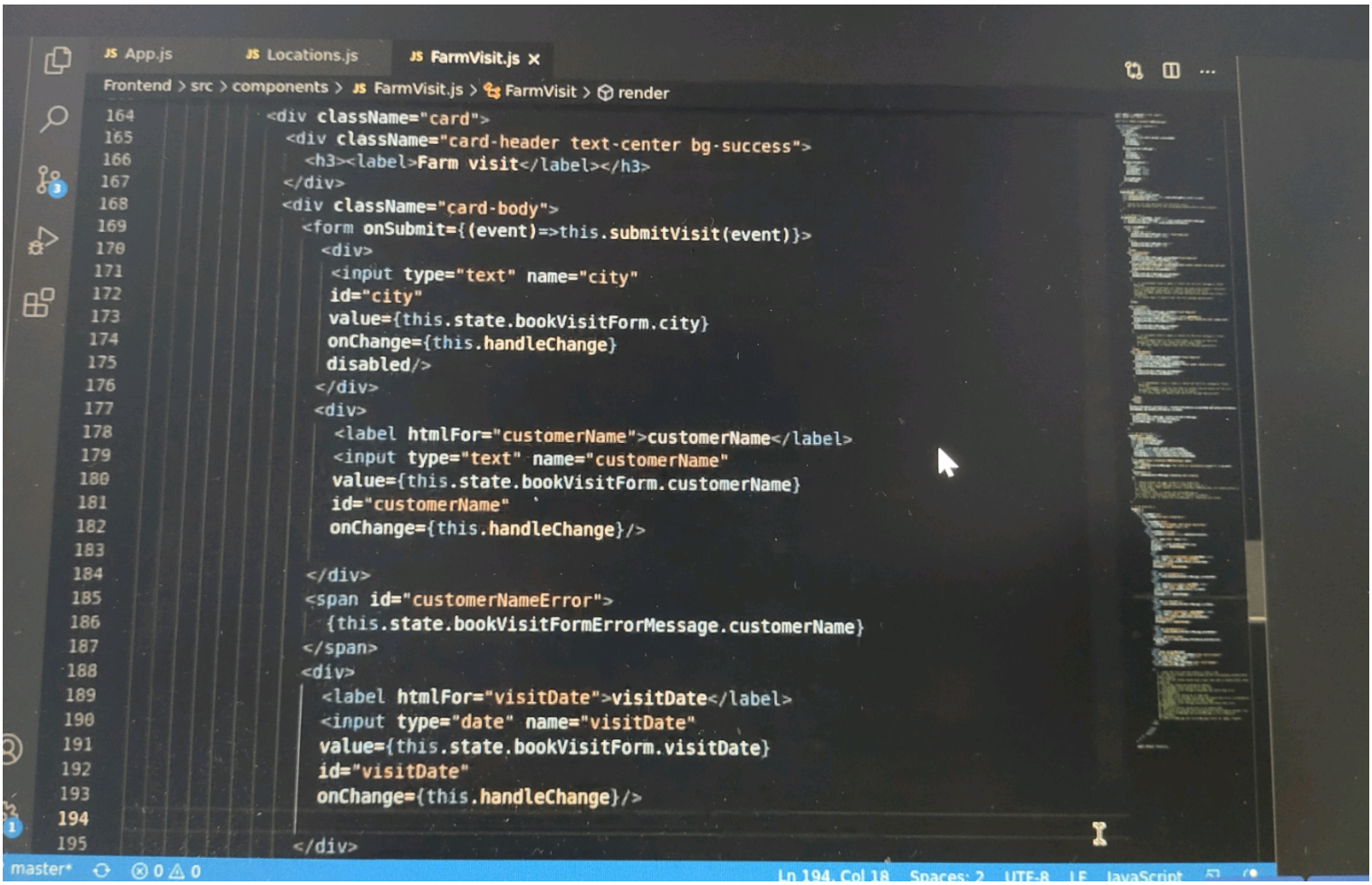

App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > validateField
}else if(new Date(value).getTime()<=new Date().getTime()){
bookVisitFormErrorMessage.visitDate="Date of visit should be after todays date"
bookVisitFormValid.visitDate=false
}else{
bookVisitFormErrorMessage.visitDate=""
bookVisitFormValid.visitDate=true
}
/*
1. if visitDate field is empty it should set the error message as "Field Required"
2. if visitDate field is not a future date, it should set the error message as "Date of visit should be after todays date"
3. for valid input it should reset the error message appropriately
*/
break;
case "phoneNumber":
if(value===""){
bookVisitFormErrorMessage.phoneNumber="Field Required"
bookVisitFormValid.phoneNumber=false
}else if(value.length<10||value.length>10){
bookVisitFormErrorMessage.phoneNumber="Phone number should be of 10 digits"
bookVisitFormValid.phoneNumber=false
}else{
bookVisitFormErrorMessage.phoneNumber=""
bookVisitFormValid.phoneNumber=true
}
/*
1. if phoneNumber field is empty it should set the error message as "Field Required"
2. if phoneNumber field has less than or greater than 10 digits set the error


App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > submitVisit > data
1. if phoneNumber field is empty it should set the error message as "Field Required"
2. if phoneNumber field has less than or greater than 10 digits set the error message as "Phone number should be of 10 digits"
3. for valid input it should reset the error message appropriately
*/
break;
default:
break;
}
bookVisitFormValid.buttonActive = bookVisitFormValid.customerName && bookVisitFormValid.phoneNumber && bookVisitFormValid.visitDate
this.setState({
bookVisitFormErrorMessage: bookVisitFormErrorMessage,
bookVisitFormValid: bookVisitFormValid,
successMessage: "", errorMessage: ""
});
}
/* To Be Implemented */
submitVisit = (event) => {
event.preventDefault()
this.state.successMessage=""
this.state.errorMessage=""
var data={
city:this.props.match.params.locationName,
visitDate:this.state.bookVisitForm.visitDate,
phoneNumber:this.state.bookVisitForm.phoneNumber,
customerName:this.state.bookVisitForm.customerName,


App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > render
phoneNumber:this.state.bookVisitForm.phoneNumber,
customerName:this.state.bookVisitForm.customerName,
}
axios.post("http://localhost:4000/bookings",data)
.then((res)=>{
this.setState({successMessage:`Your Visit is successfully booked for ${res.data.
visitDate}` })
})
.catch((err)=>{
this.setState({errorMessage:"Something went wrong"})
})
/*
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 bookVisitForm as
the request body
4. in success case, it should set successMessage as
Your Visit is successfully bookèd for <<response visitDate>>
5. in error case, it should set errorMessage as 'Something went wrong'
*/
}
/* To Be Modified */
render() {
return (

<h3><label>Farm visit</label></h3>

master*
Ln 166, Col 49
Spaces: 2
UTF-8
LF
JavaScript


App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > render

<h3><label>Farm visit</label></h3>

<form onSubmit={(event)=>this.submitVisit(event)}>

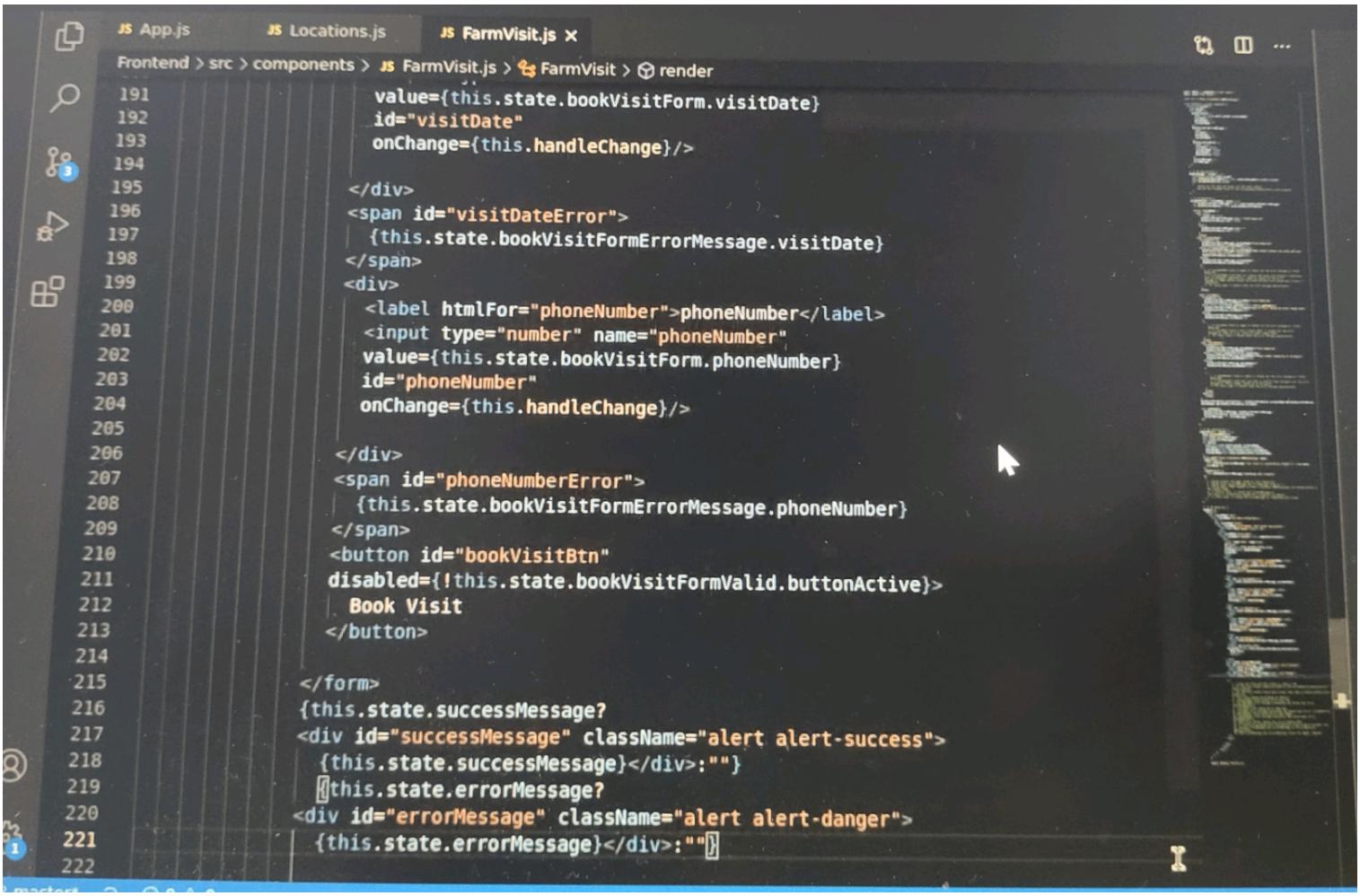
<input type="text" name="city"
id="city"
value={this.state.bookVisitForm.city}
onChange={this.handleChange}
disabled/>

<label htmlFor="customerName">customerName</label>
<input type="text" name="customerName"
value={this.state.bookVisitForm.customerName}
id="customerName"
onChange={this.handleChange}/>

{this.state.bookVisitFormErrorMessage.customerName}

<label htmlFor="visitDate">visitDate</label>
<input type="date" name="visitDate"
value={this.state.bookVisitForm.visitDate}
id="visitDate"
onChange={this.handleChange}/>

master*
Ln 194, Col 18
Spaces: 2
UTF-8
LF
JavaScript


App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > render
value={this.state.bookVisitForm.visitDate}
id="visitDate"
onChange={this.handleChange}/>

{this.state.bookVisitFormErrorMessage.visitDate}

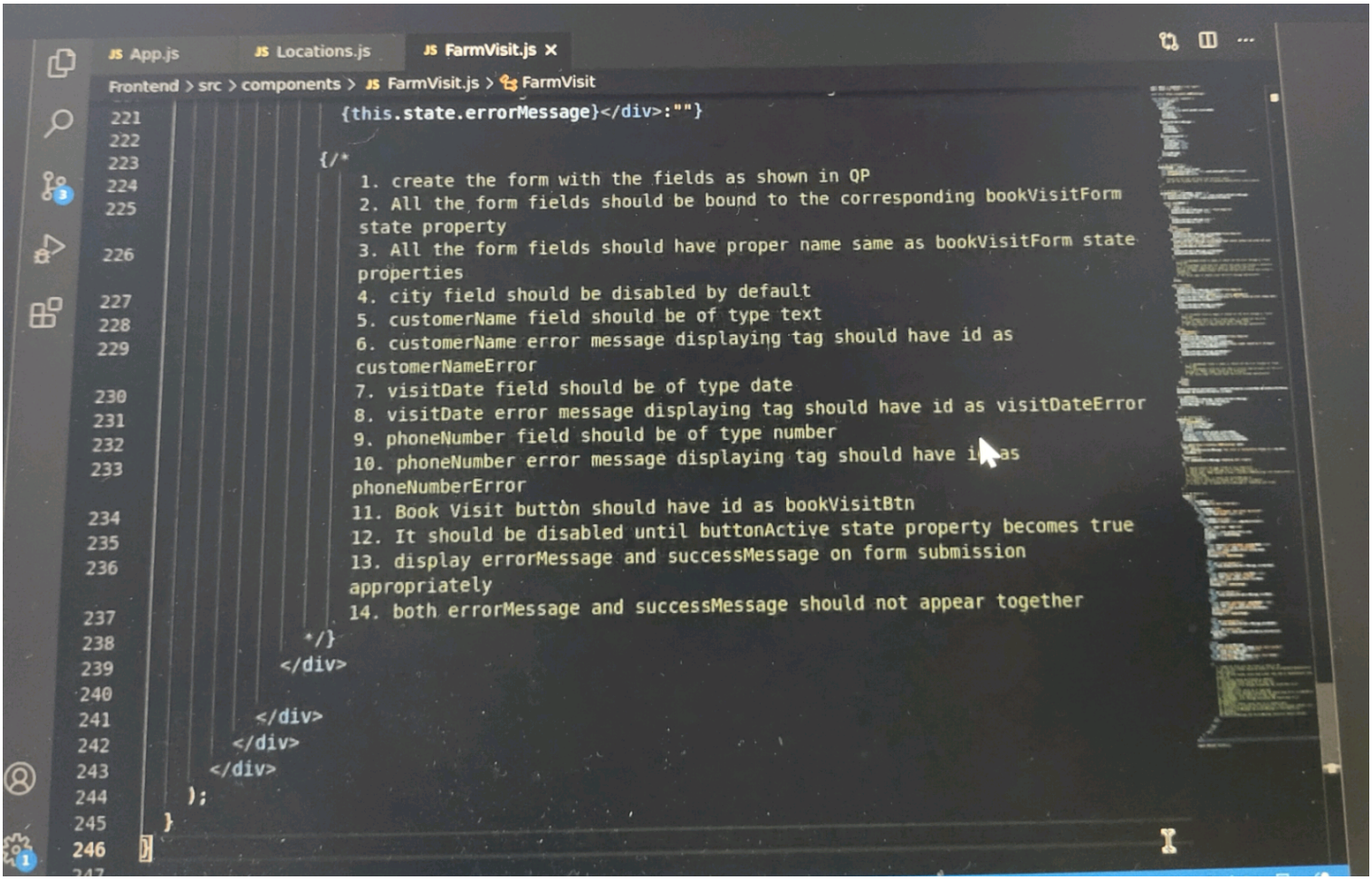
<label htmlFor="phoneNumber">phoneNumber</label>
<input type="number" name="phoneNumber"
value={this.state.bookVisitForm.phoneNumber}
id="phoneNumber"
onChange={this.handleChange}/>

{this.state.bookVisitFormErrorMessage.phoneNumber}

<button id="bookVisitBtn"
disabled={!this.state.bookVisitFormValid.buttonActive}>
Book Visit
</button>
</form>
{this.state.successMessage?

{this.state.successMessage} :""}
{this.state.errorMessage?

{this.state.errorMessage} :""}


App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit
{this.state.errorMessage} :""}
{/*
1. create the form with the fields as shown in QP
2. All the form fields should be bound to the corresponding bookVisitForm state property
3. All the form fields should have proper name same as bookVisitForm state properties
4. city field should be disabled by default
5. customerName field should be of type text
6. customerName error message displaying tag should have id as customerNameError
7. visitDate field should be of type date
8. visitDate error message displaying tag should have id as visitDateError
9. phoneNumber field should be of type number
10. phoneNumber error message displaying tag should have id as phoneNumberError
11. Book Visit button should have id as bookVisitBtn
12. It should be disabled until buttonActive state property becomes true
13. display errorMessage and successMessage on form submission appropriately
14. both errorMessage and successMessage should not appear together
*/}

);
}
}