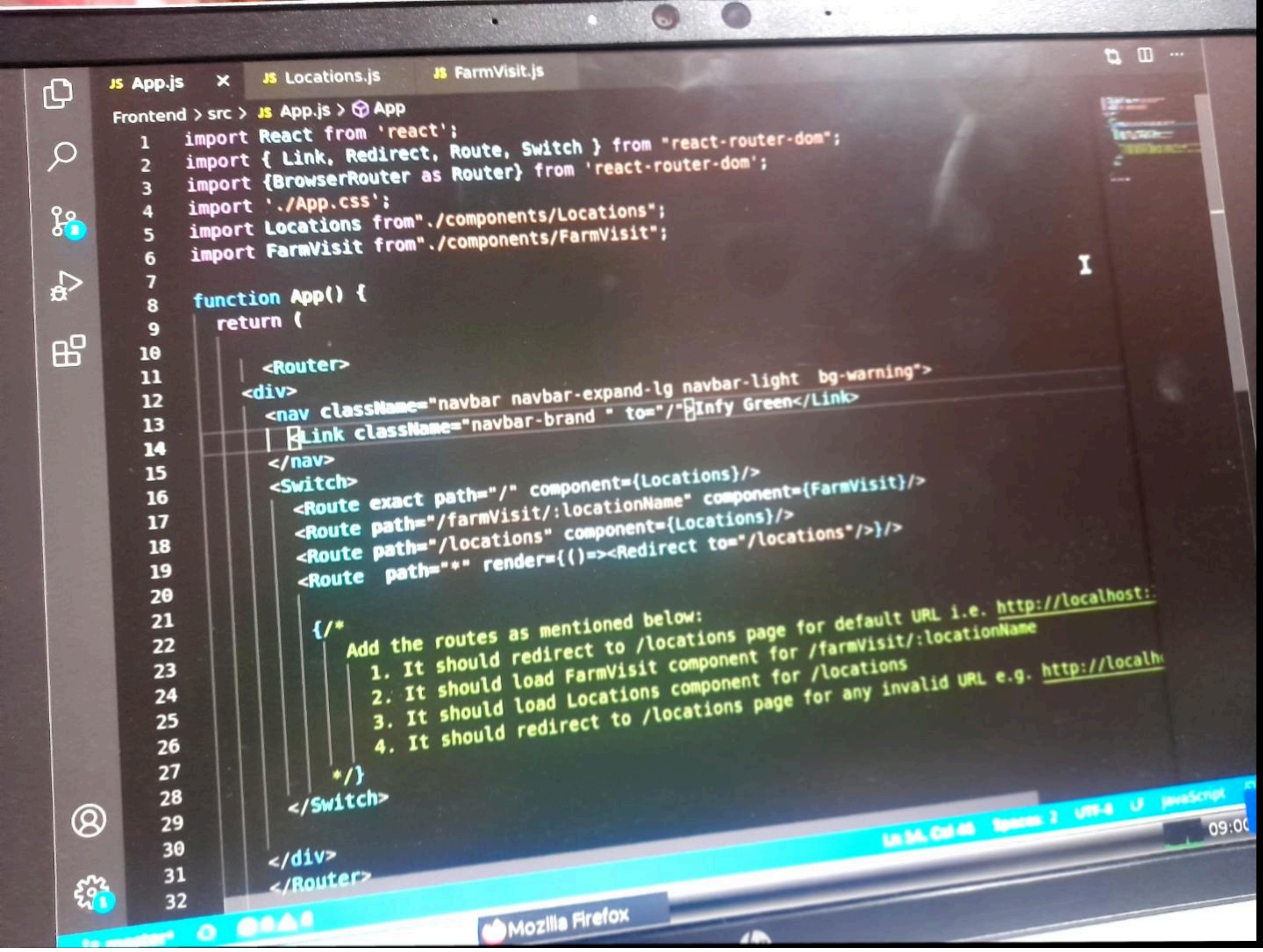
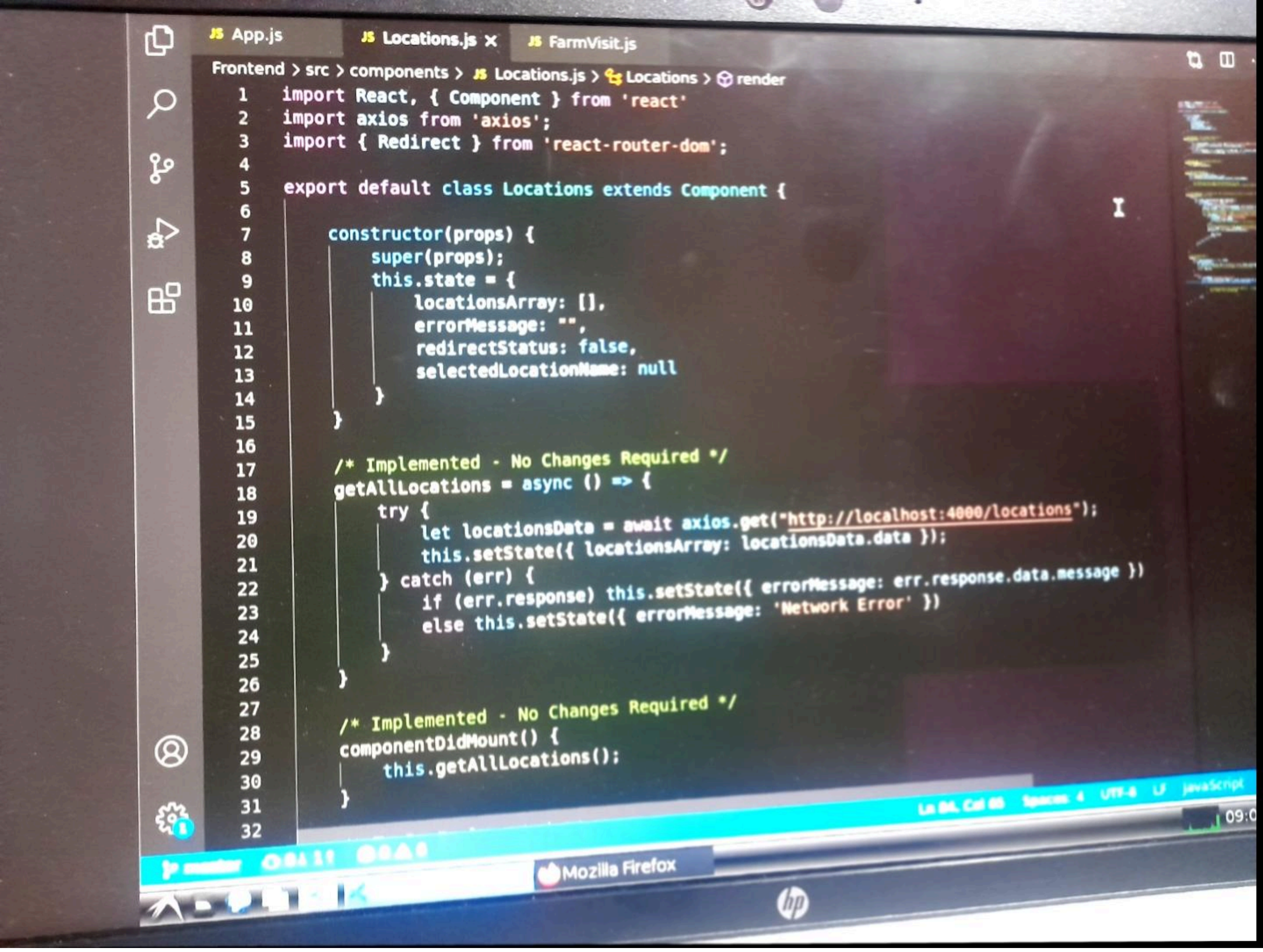
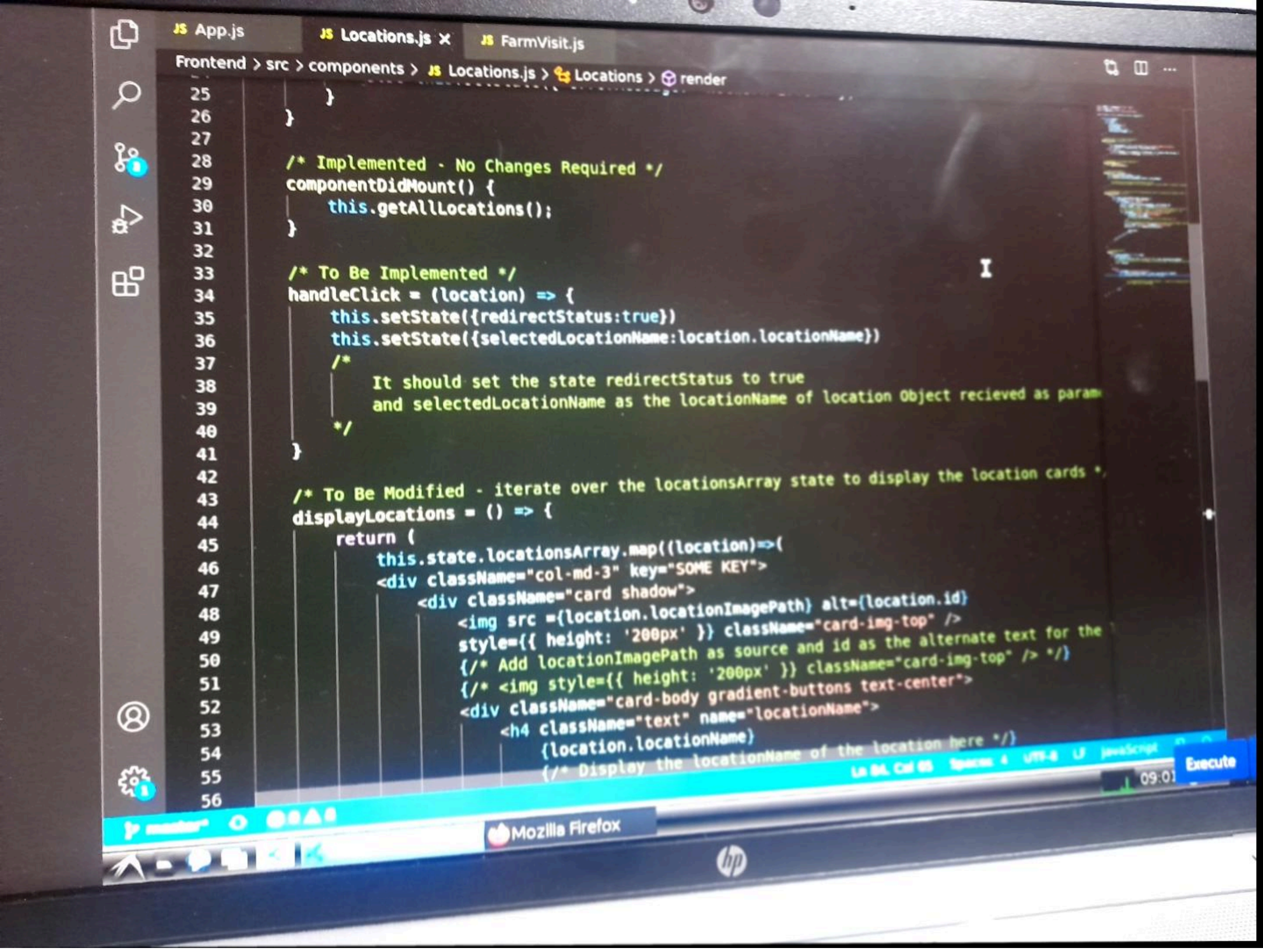


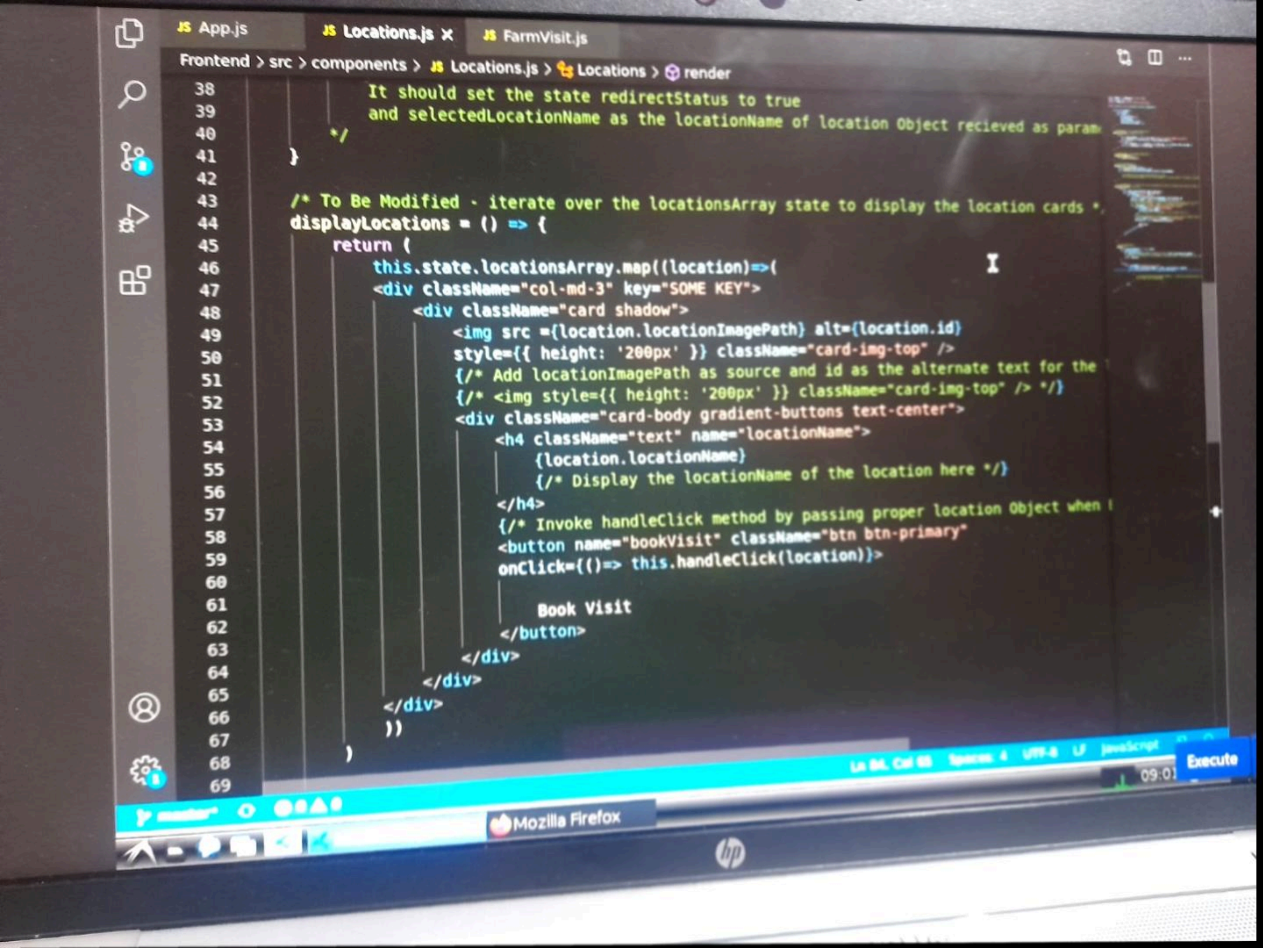
```
import React from 'react';
import { Link, Redirect, Route, Switch } from "react-router-dom";
import {BrowserRouter as Router} from 'react-router-dom';
import './App.css';
import Locations from"./components/Locations";
import FarmVisit from"./components/FarmVisit";

function App() {
  return (

      <Router>
    <div>
      <nav className="navbar navbar-expand-lg navbar-light  bg-warning">
      | <Link className="navbar-brand " to="/">Infy Green</Link>
      </nav>
      <Switch>
        <Route exact path="/" component={Locations}/>
        <Route path="/farmVisit/:locationName" component={FarmVisit}/>
        <Route path="/locations" component={Locations}/>
        <Route  path="*" render={()=><Redirect to="/locations"/>}/>

        {/*
          Add the routes as mentioned below:
            1. It should redirect to /locations page for default URL i.e. http://localhost:
            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://localh
          */}
      </Switch>

    </div>
    </Router>
```


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

```
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();
    }
```

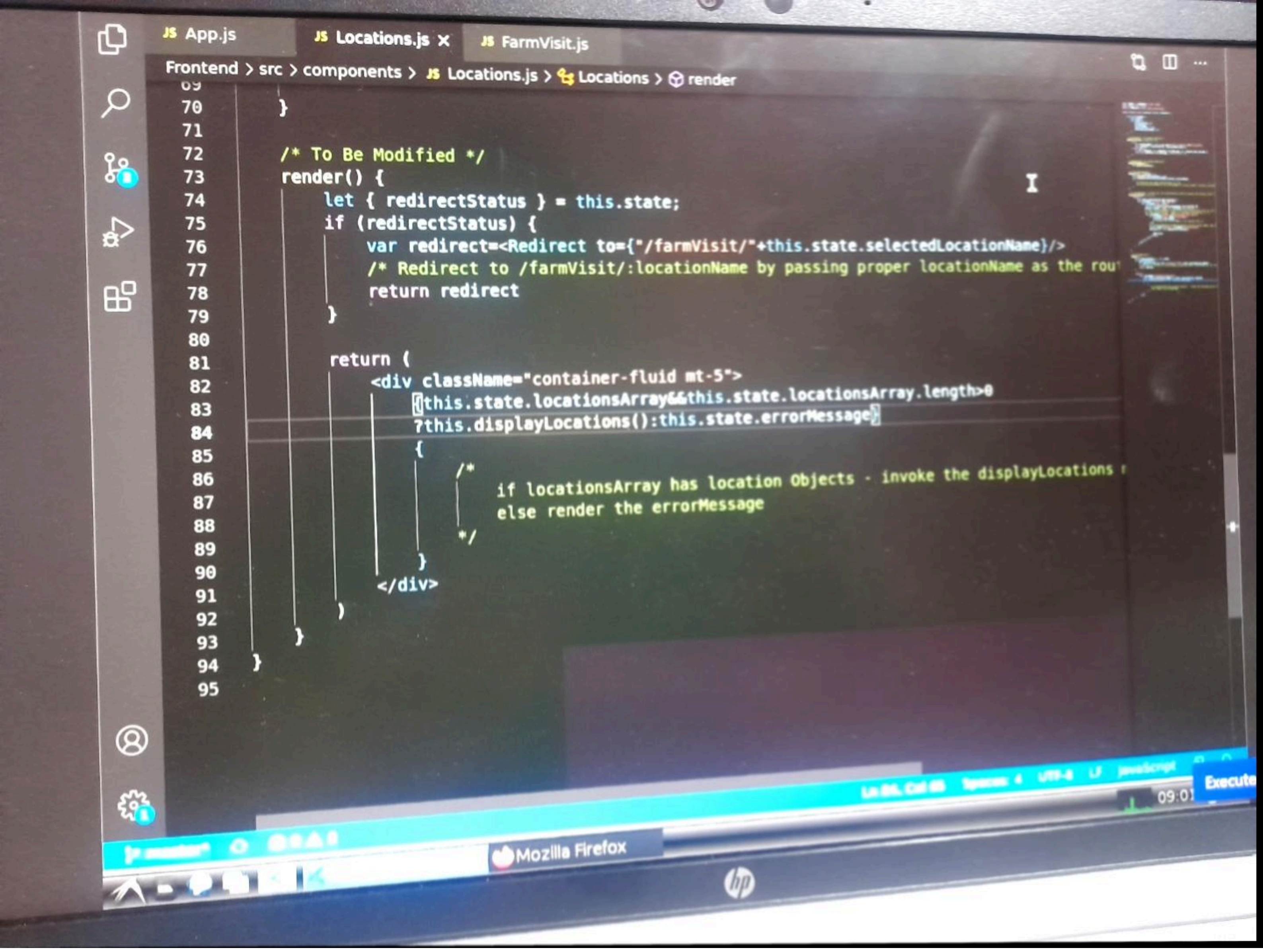

App.js
Locations.js
FarmVisit.js
Frontend > src > components > Locations.js > Locations > render
Mozilla Firefox
Execute
09:01

```
    }
}

/* Implemented - No Changes Required */
componentDidMount() {
    this.getAllLocations();
}

/* To Be Implemented */
handleClick = (location) => {
    this.setState({redirectStatus:true})
    this.setState({selectedLocationName:location.locationName})
    /*
        It should set the state redirectStatus to true
        and selectedLocationName as the locationName of location Object recieved as param
    */
}

/* To Be Modified - iterate over the locationsArray state to display the location cards *
displayLocations = () => {
    return (
        this.state.locationsArray.map((location)=>(
        <div className="col-md-3" key="SOME KEY">
            <div className="card shadow">
                <img src ={location.locationImagePath} alt={location.id}
                style={{ height: '200px' }} className="card-img-top" />
                {/* Add locationImagePath as source and id as the alternate text for the
                {/* <img style={{ height: '200px' }} className="card-img-top" /> */}
                <div className="card-body gradient-buttons text-center">
                    <h4 className="text" name="locationName">
                        {location.locationName}
                        {/* Display the locationName of the location here */}
```

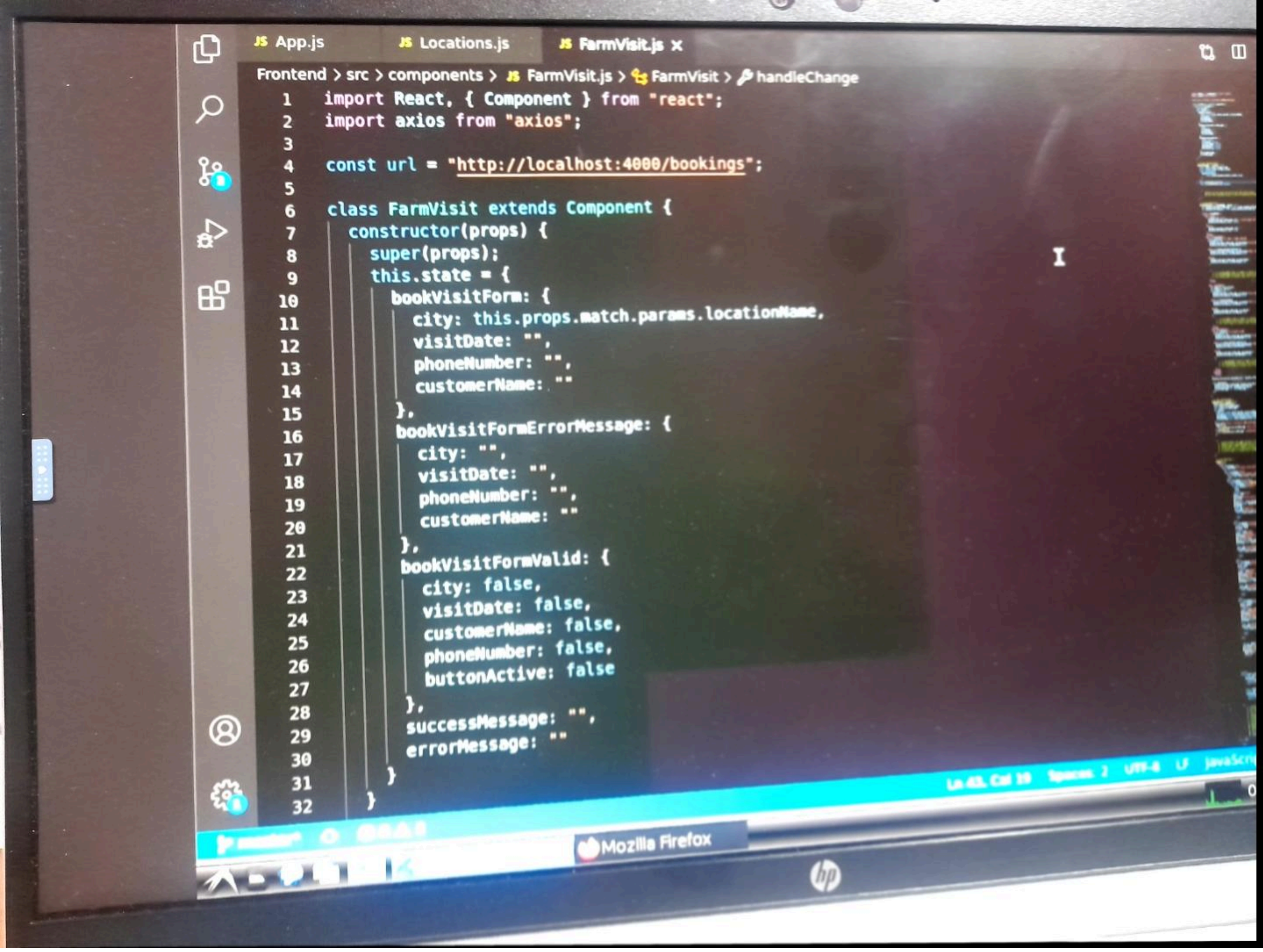


App.js
Locations.js
FarmVisit.js
Frontend > src > components > Locations.js > Locations > render
master*
Execute
Mozilla Firefox

```
            It should set the state redirectStatus to true
            and selectedLocationName as the locationName of location Object recieved as param
        */
    }

    /* To Be Modified - iterate over the locationsArray state to display the location cards *
    displayLocations = () => {
        return (
            this.state.locationsArray.map((location)=>(
            <div className="col-md-3" key="SOME KEY">
                <div className="card shadow">
                    <img src ={location.locationImagePath} alt={location.id}
                    style={{ height: '200px' }} className="card-img-top" />
                    {/* Add locationImagePath as source and id as the alternate text for the
                    {/* <img style={{ height: '200px' }} className="card-img-top" /> */}
                    <div className="card-body gradient-buttons text-center">
                        <h4 className="text" name="locationName">
                            {location.locationName}
                            {/* Display the locationName of the location here */}
                        </h4>
                        {/* Invoke handleClick method by passing proper location Object when
                        <button name="bookVisit" className="btn btn-primary"
                        onClick={()=> this.handleClick(location)}>

                            Book Visit
                        </button>
                    </div>
                </div>
            </div>
            ))
        )
```

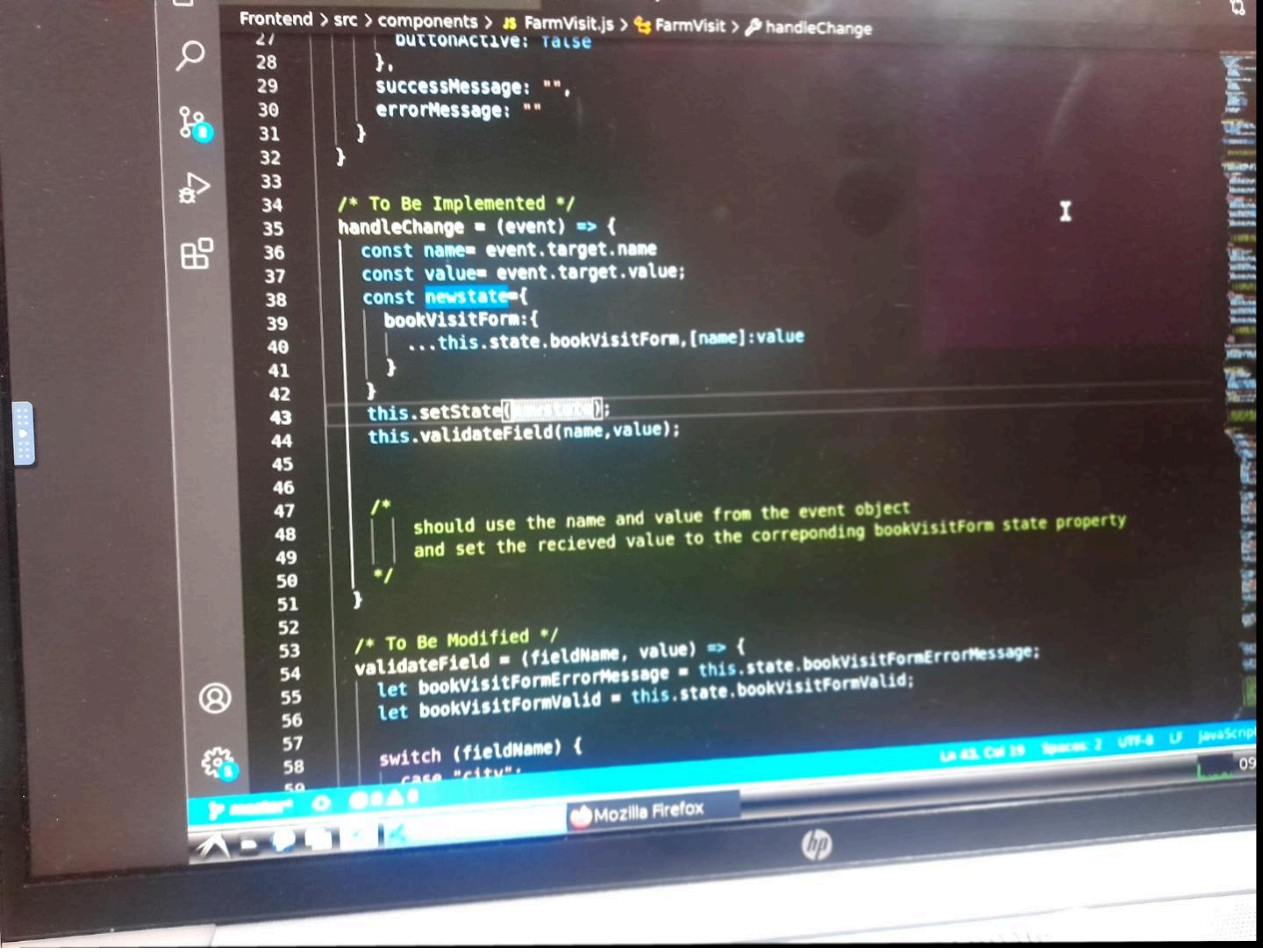

App.js
Locations.js
FarmVisit.js
Frontend > src > components > Locations.js > Locations > render
Mozilla Firefox
Execute

```
    }

    /* 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 rou
            return redirect
        }

         return (
             <div className="container-fluid mt-5">
                 {this.state.locationsArray&&this.state.locationsArray.length>0
                 ?this.displayLocations():this.state.errorMessage}
                 {
                     /*
                         if locationsArray has location Objects - invoke the displayLocations
                         else render the errorMessage
                     */
                 }
             </div>
         )
    }
}
```



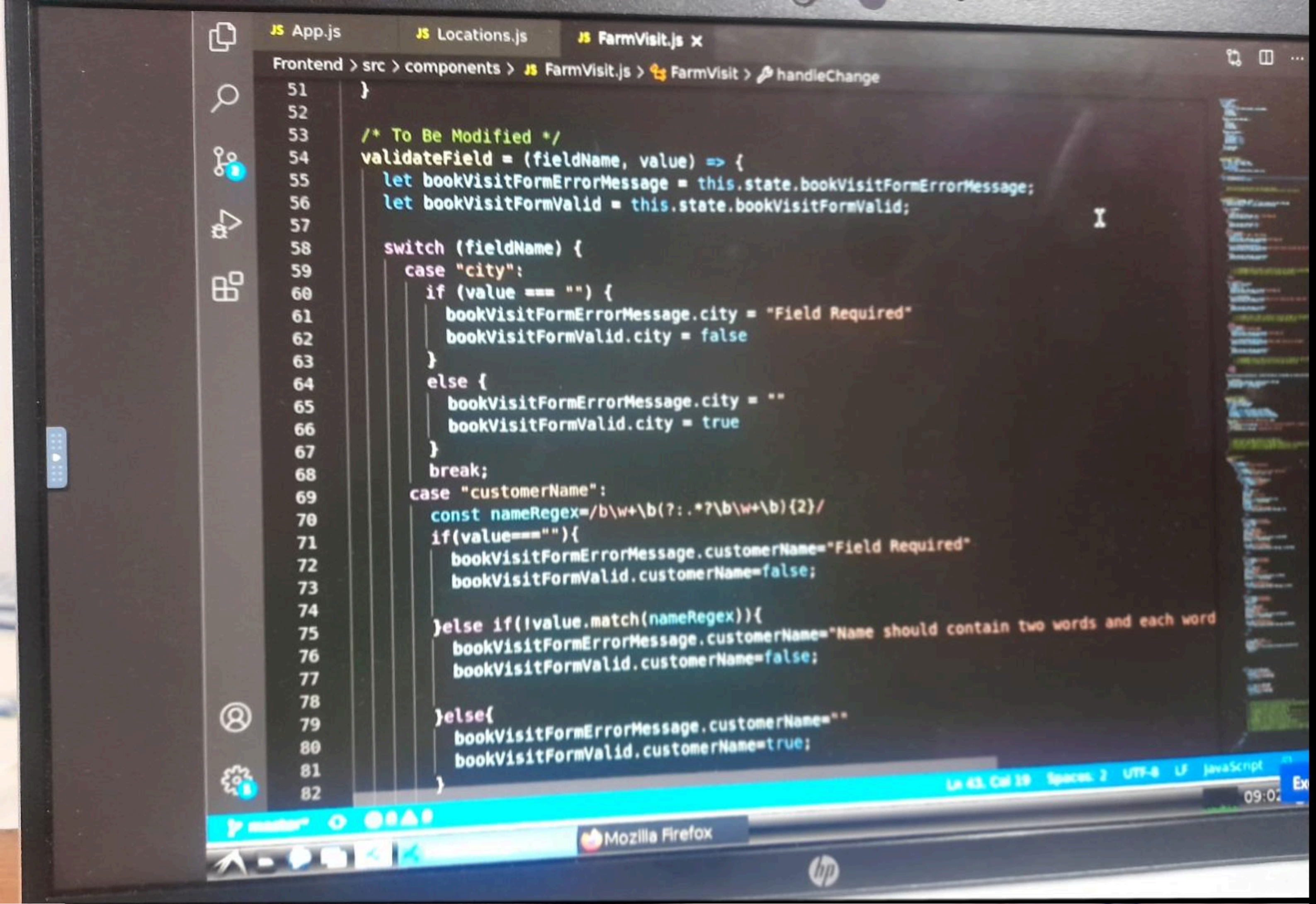
App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > handleChange
Mozilla Firefox

```
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: ""
    }
  }
```

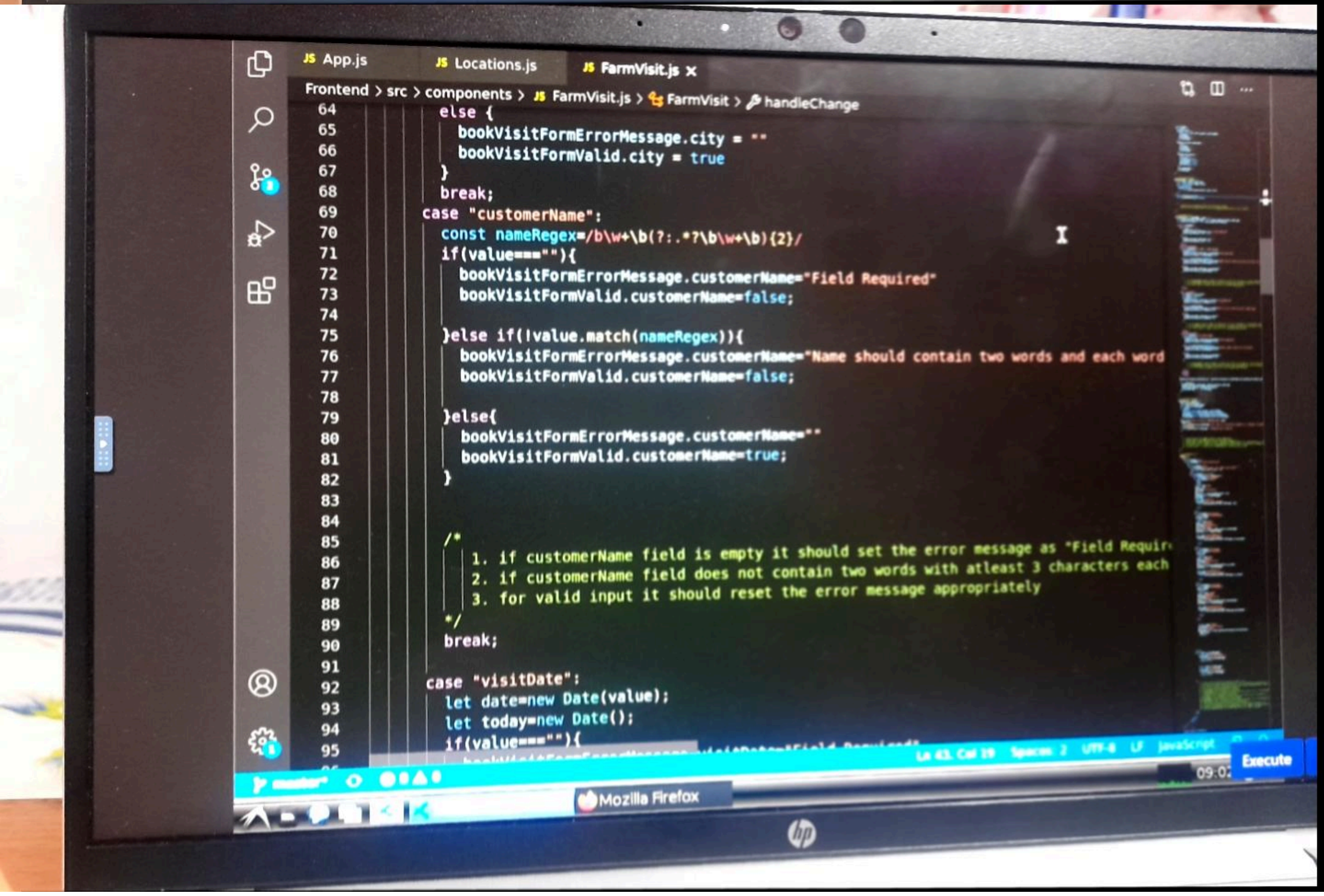


Frontend > src > components > FarmVisit.js > FarmVisit > handleChange

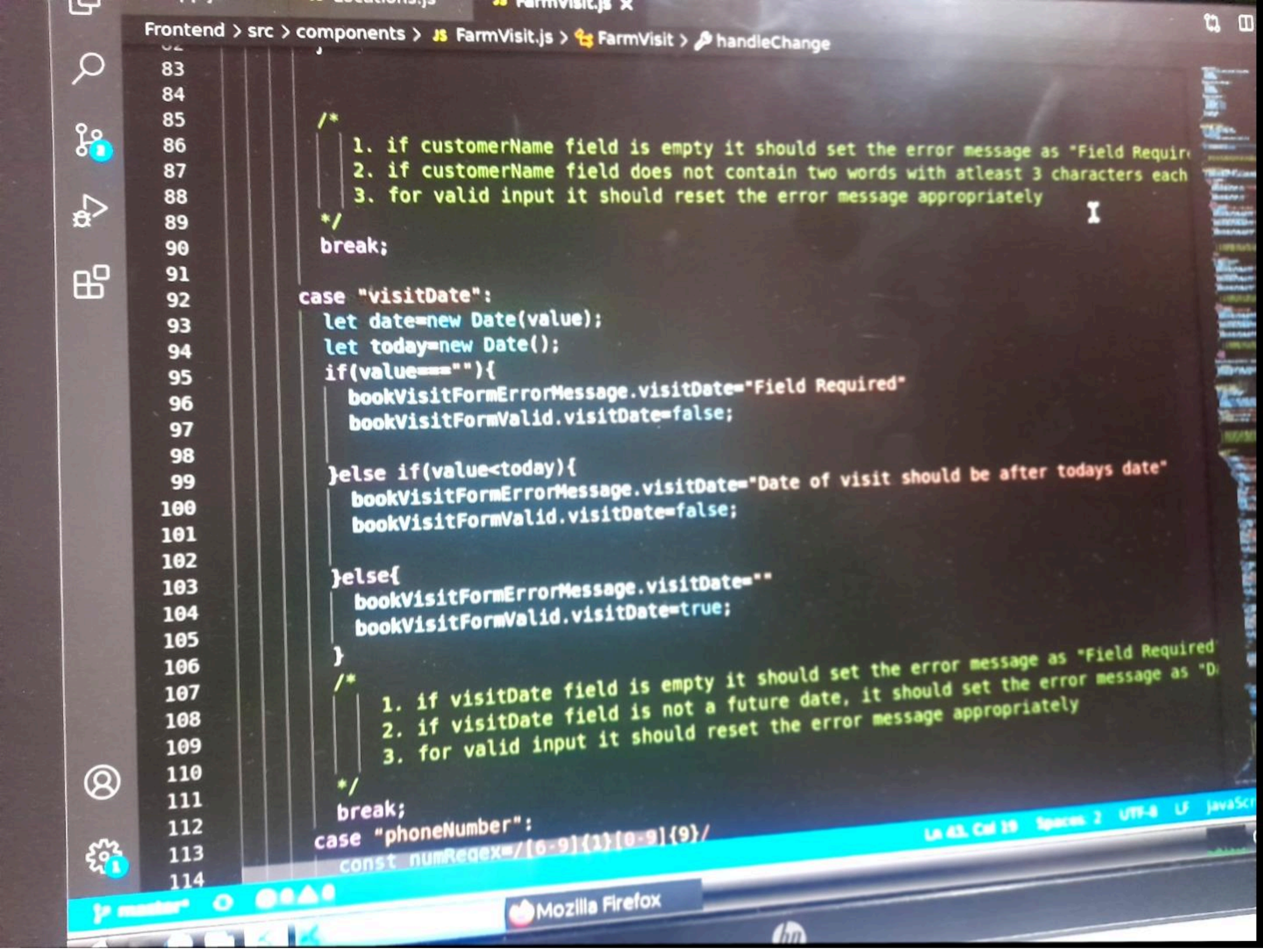
```
            buttonActive: false
        },
        successMessage: "",
        errorMessage: ""
    }
}

/* To Be Implemented */
handleChange = (event) => {
    const name= event.target.name
    const value= event.target.value;
    const newstate={
        bookVisitForm:{
            ...this.state.bookVisitForm,[name]:value
        }
    }
    this.setState(newstate);
    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":
```


App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > handleChange
Mozilla Firefox

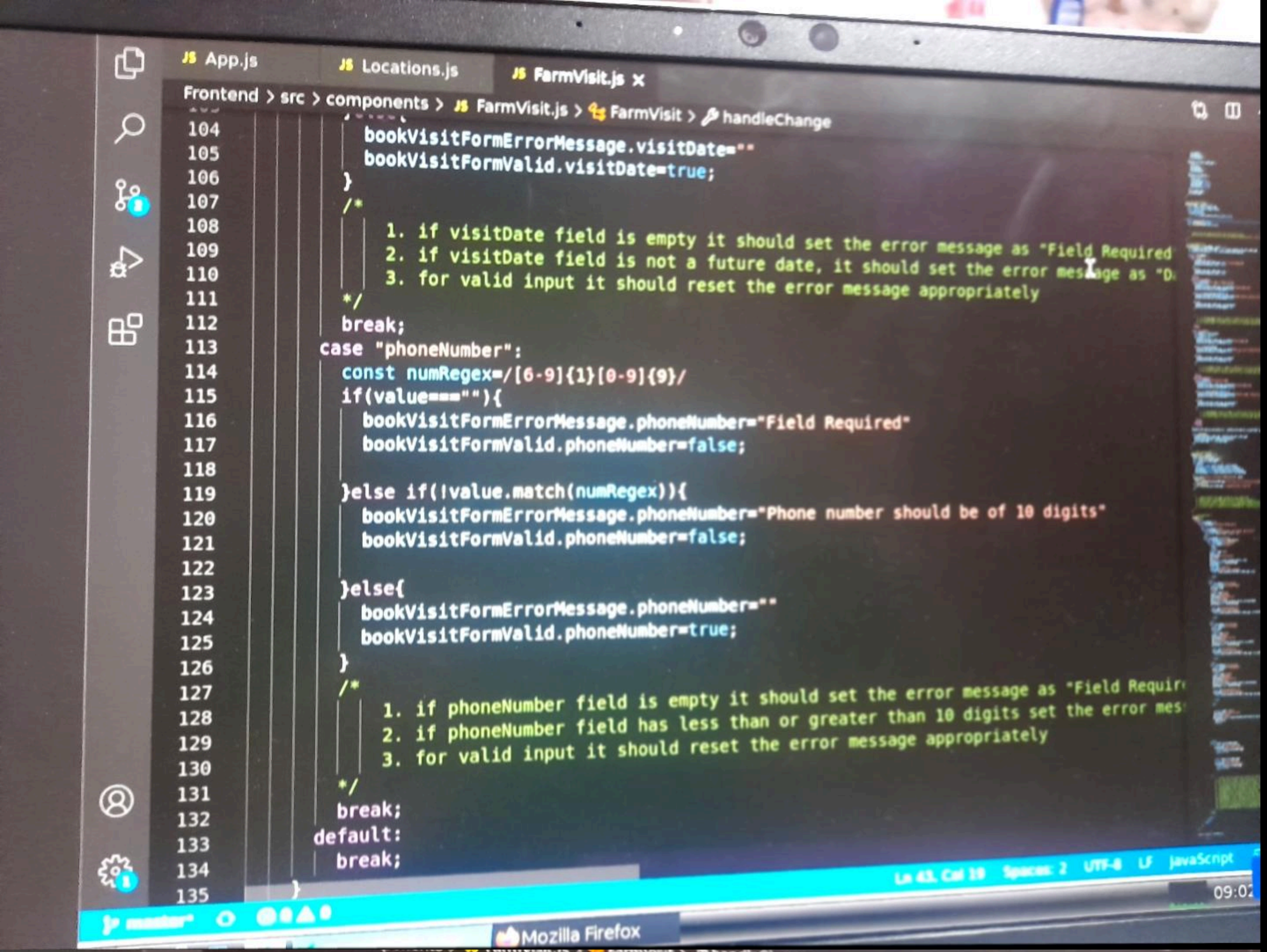
```
}

/* 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;
    case "customerName":
      const nameRegex=/b\w+\b(?:.*?\b\w+\b){2}/
      if(value===""){
        bookVisitFormErrorMessage.customerName="Field Required"
        bookVisitFormValid.customerName=false;

      }else if(!value.match(nameRegex)){
        bookVisitFormErrorMessage.customerName="Name should contain two words and each word
        bookVisitFormValid.customerName=false;

      }else{
        bookVisitFormErrorMessage.customerName=""
        bookVisitFormValid.customerName=true;
      }
```


App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > handleChange
Execute
Mozilla Firefox

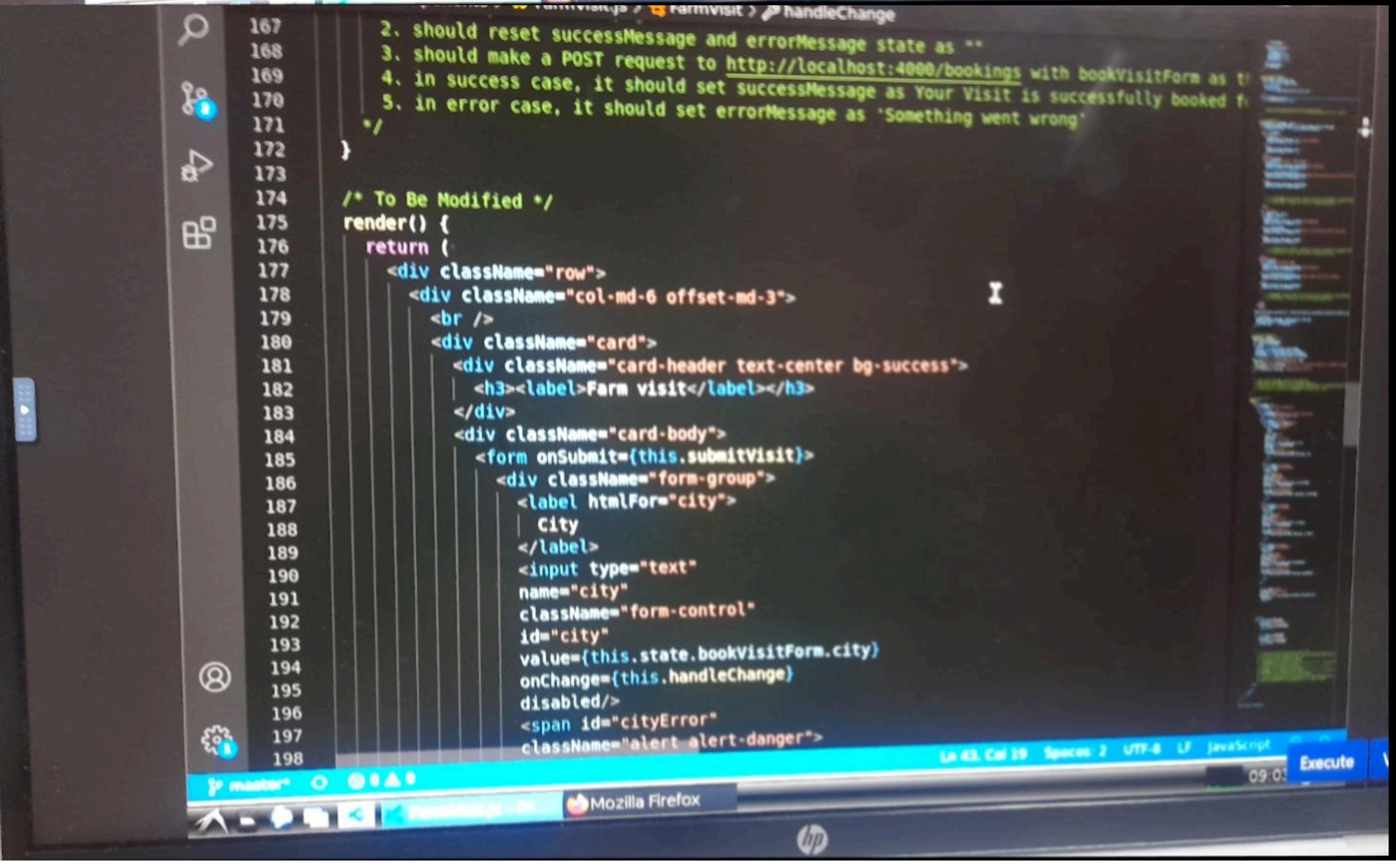
```
      else {
        bookVisitFormErrorMessage.city = ""
        bookVisitFormValid.city = true
      }
      break;
    case "customerName":
      const nameRegex=/b\w+\b(?:.*?\b\w+\b){2}/
      if(value===""){
        bookVisitFormErrorMessage.customerName="Field Required"
        bookVisitFormValid.customerName=false;

      }else if(!value.match(nameRegex)){
        bookVisitFormErrorMessage.customerName="Name should contain two words and each word
        bookVisitFormValid.customerName=false;

      }else{
        bookVisitFormErrorMessage.customerName=""
        bookVisitFormValid.customerName=true;
      }


      /*
        1. if customerName field is empty it should set the error message as "Field Requir
        2. if customerName field does not contain two words with atleast 3 characters each
        3. for valid input it should reset the error message appropriately
      */
      break;

    case "visitDate":
      let date=new Date(value);
      let today=new Date();
      if(value===""){
```


Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > handleChange
master*
Ln 43, Col 19 Spaces: 2 UTF-8 LF JavaScript
Mozilla Firefox

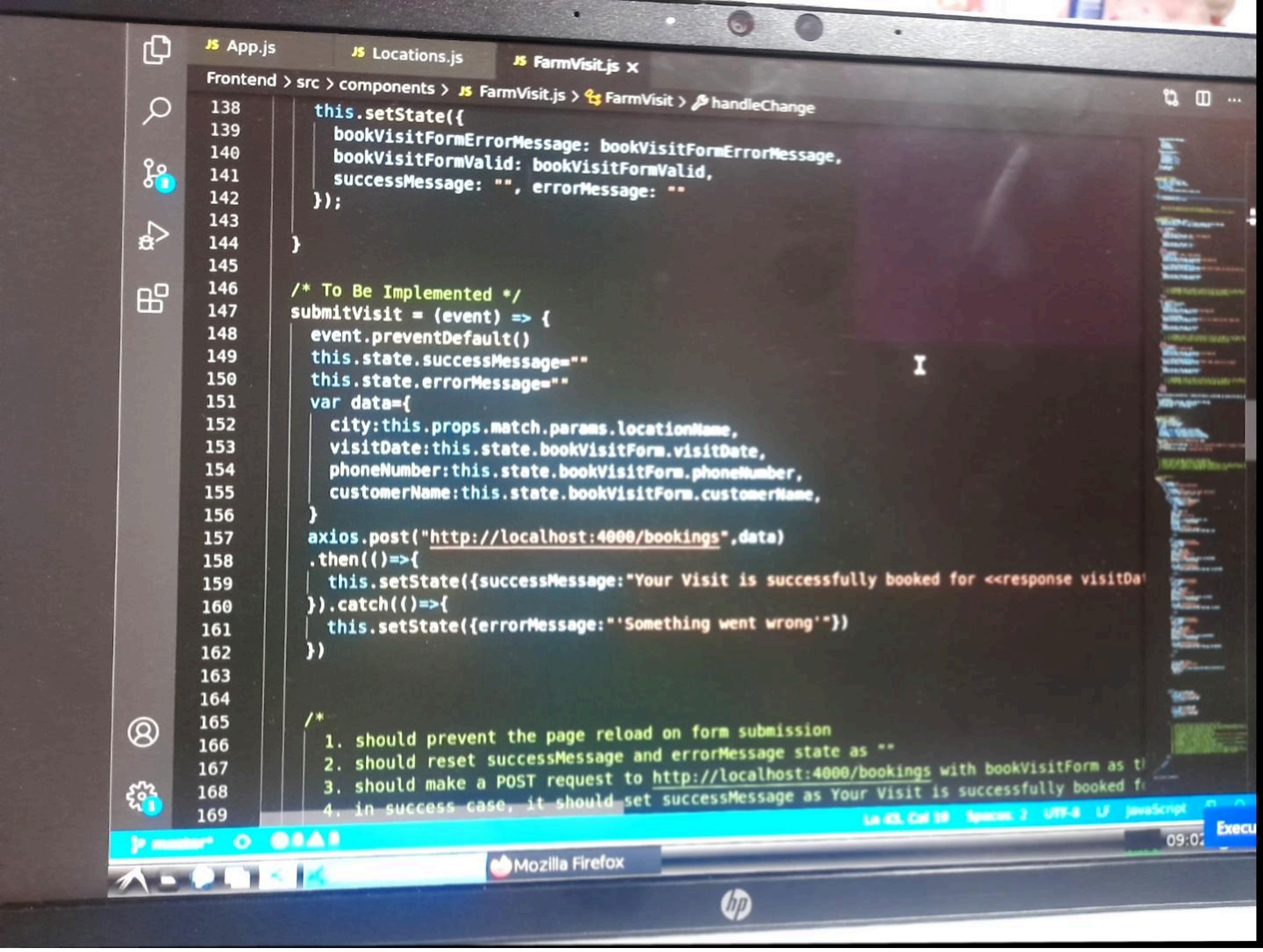
```
      /*
        1. if customerName field is empty it should set the error message as "Field Requir
        2. if customerName field does not contain two words with atleast 3 characters each
        3. for valid input it should reset the error message appropriately
      */
      break;

    case "visitDate":
      let date=new Date(value);
      let today=new Date();
      if(value===""){
        bookVisitFormErrorMessage.visitDate="Field Required"
        bookVisitFormValid.visitDate=false;

      }else if(value<today){
        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 "D
        3. for valid input it should reset the error message appropriately
      */
      break;
    case "phoneNumber":
      const numRegex=/[6-9]{1}[0-9]{9}/
```


App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > handleChange
Ln 43, Col 19 Spaces: 2 UTF-8 LF JavaScript
master*
Mozilla Firefox
09:02

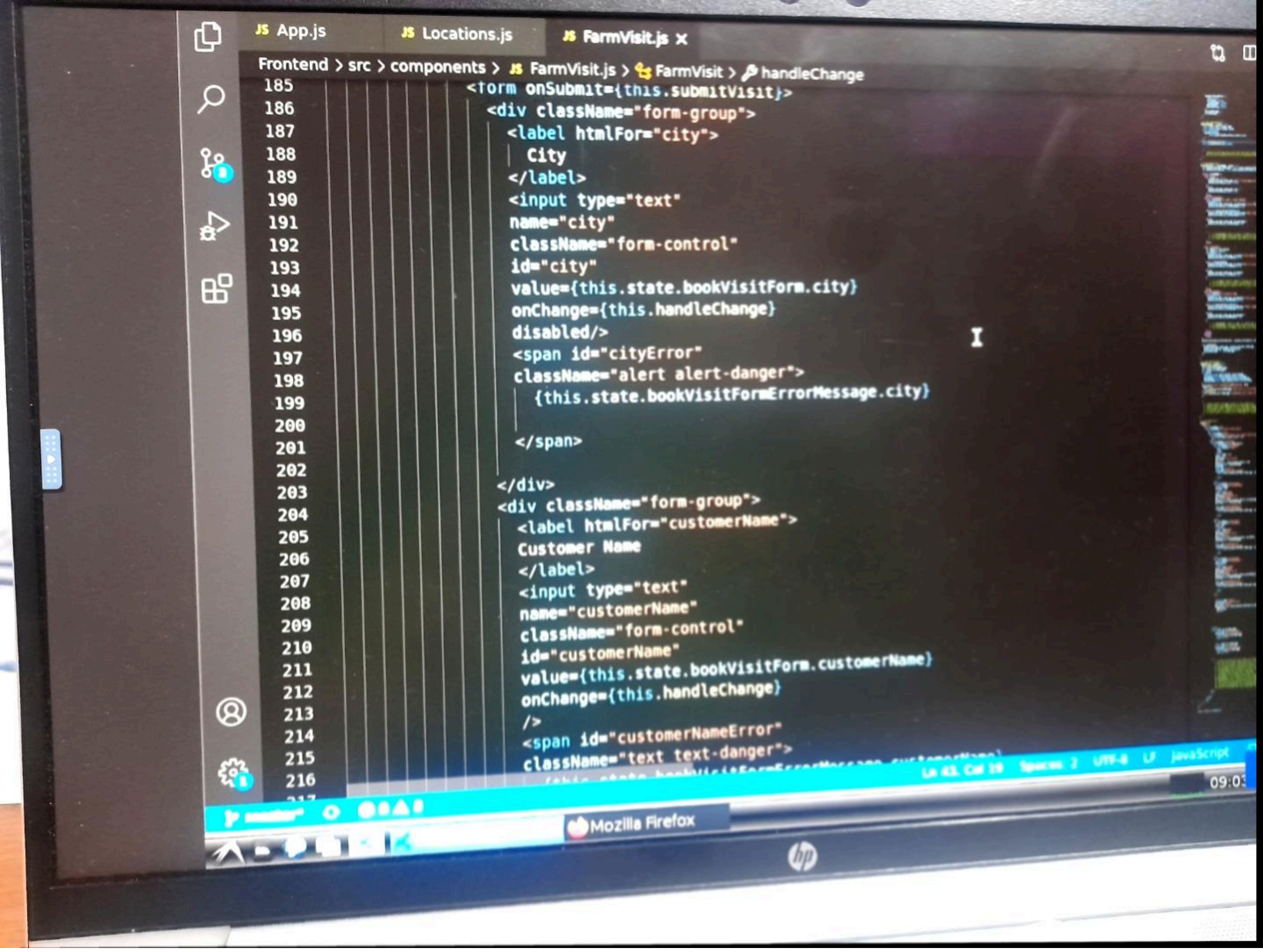
```
        bookVisitFormErrorMessage.visitDate=""
        bookVisitFormValid.visitDate=true;
      }
      /*
        1. if visitDate field is empty it should set the error message as "Field Required
        2. if visitDate field is not a future date, it should set the error message as "D
        3. for valid input it should reset the error message appropriately
      */
      break;
    case "phoneNumber":
      const numRegex=/[6-9]{1}[0-9]{9}/
      if(value===""){
        bookVisitFormErrorMessage.phoneNumber="Field Required"
        bookVisitFormValid.phoneNumber=false;

      }else if(!value.match(numRegex)){
        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 Requir
        2. if phoneNumber field has less than or greater than 10 digits set the error mes
        3. for valid input it should reset the error message appropriately
      */
      break;
    default:
      break;
  }
```


FarmVisit.js > FarmVisit > handleChange
Ln 43, Col 19 Spaces: 2 UTF-8 LF JavaScript
master*
Mozilla Firefox
Execute
09:03

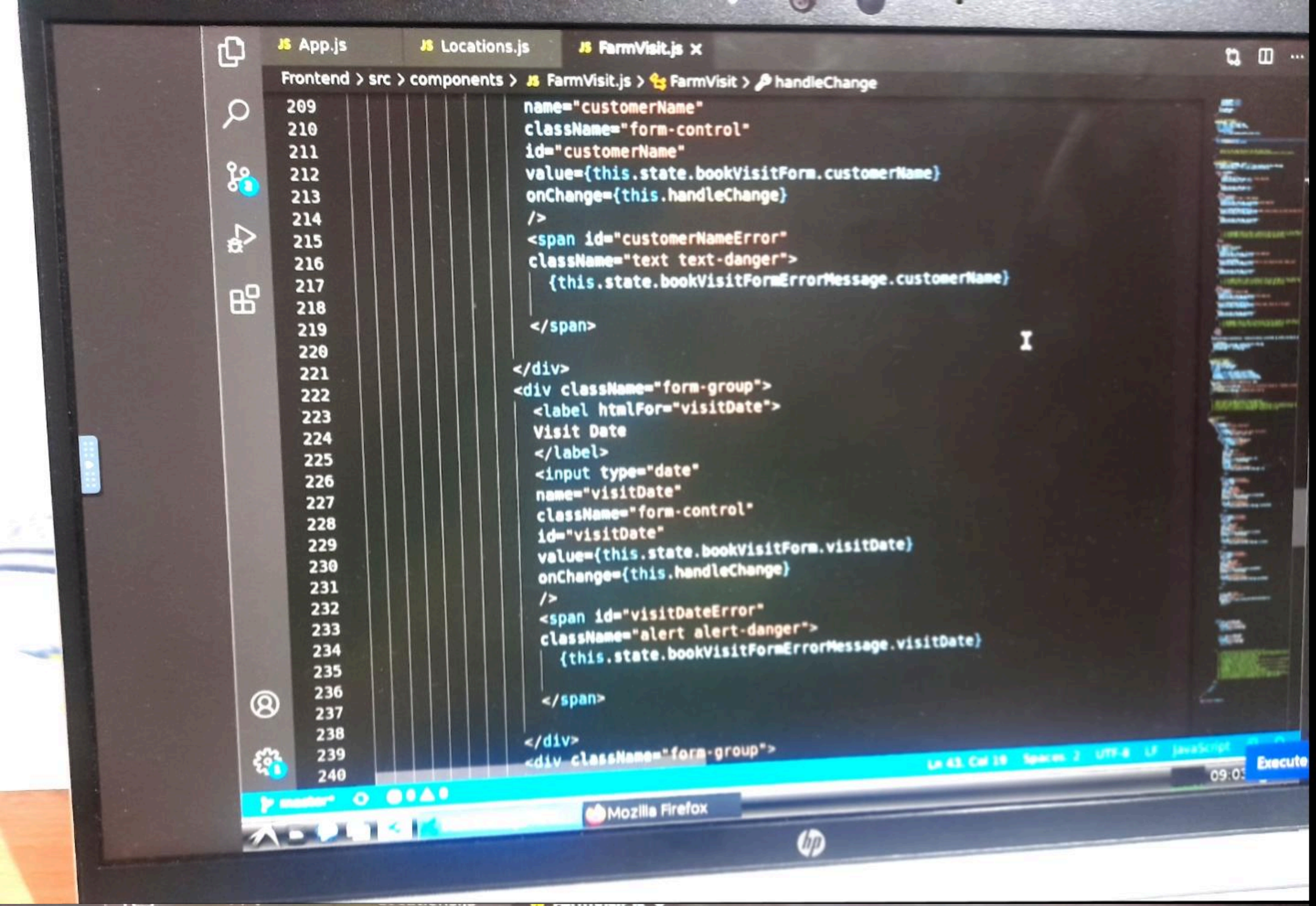
```
      2. should reset successMessage and errorMessage state as ""
      3. should make a POST request to http://localhost:4000/bookings with bookVisitForm as t
      4. in success case, it should set successMessage as Your Visit is successfully booked f
      5. in error case, it should set errorMessage as 'Something went wrong'
    */
  }

  /* To Be Modified */
  render() {
    return (
      <div className="row">
        <div className="col-md-6 offset-md-3">
          <br />
          <div className="card">
            <div className="card-header text-center bg-success">
              <h3><label>Farm visit</label></h3>
            </div>
            <div className="card-body">
              <form onSubmit={this.submitVisit}>
                <div className="form-group">
                  <label htmlFor="city">
                    City
                  </label>
                  <input type="text"
                  name="city"
                  className="form-control"
                  id="city"
                  value={this.state.bookVisitForm.city}
                  onChange={this.handleChange}
                  disabled/>
                  <span id="cityError"
                  className="alert alert-danger">
```


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

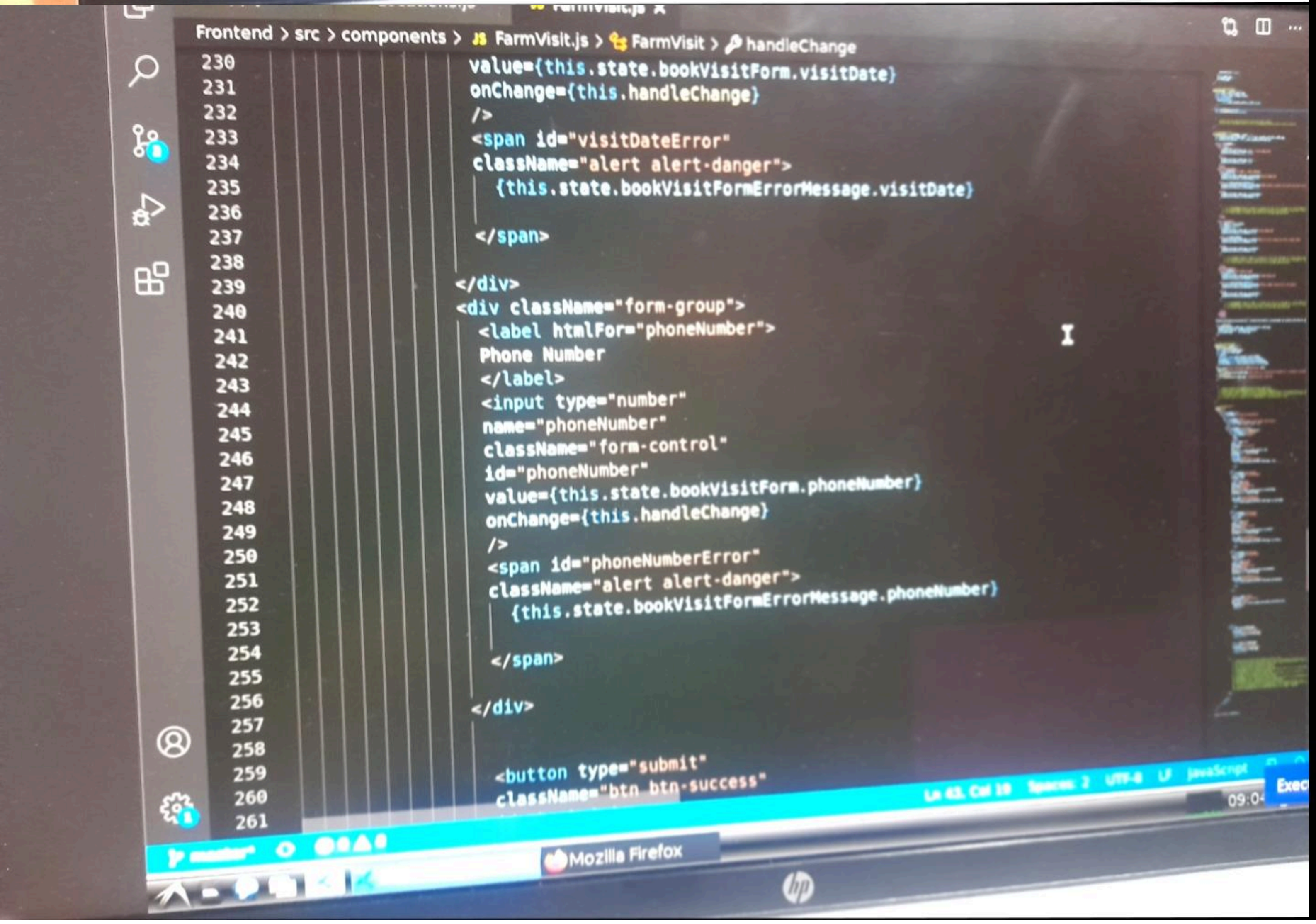
```
      this.setState({
        bookVisitFormErrorMessage: bookVisitFormErrorMessage,
        bookVisitFormValid: bookVisitFormValid,
        successMessage: "", errorMessage: ""
      });

    }

  /* 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,
    }
    axios.post("http://localhost:4000/bookings",data)
    .then(()=>{
      this.setState({successMessage:"Your Visit is successfully booked for <<response visitDa
    }).catch(()=>{
      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 t
      4. in success case, it should set successMessage as Your Visit is successfully booked f
```


App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > handleChange
master
Mozilla Firefox
09:03

```
<form onSubmit={this.submitVisit}>
  <div className="form-group">
    <label htmlFor="city">
      City
    </label>
    <input type="text"
    name="city"
    className="form-control"
    id="city"
    value={this.state.bookVisitForm.city}
    onChange={this.handleChange}
    disabled/>
    <span id="cityError"
    className="alert alert-danger">
      {this.state.bookVisitFormErrorMessage.city}

    </span>

  </div>
  <div className="form-group">
    <label htmlFor="customerName">
    Customer Name
    </label>
    <input type="text"
    name="customerName"
    className="form-control"
    id="customerName"
    value={this.state.bookVisitForm.customerName}
    onChange={this.handleChange}
    />
    <span id="customerNameError"
    className="text text-danger">
```

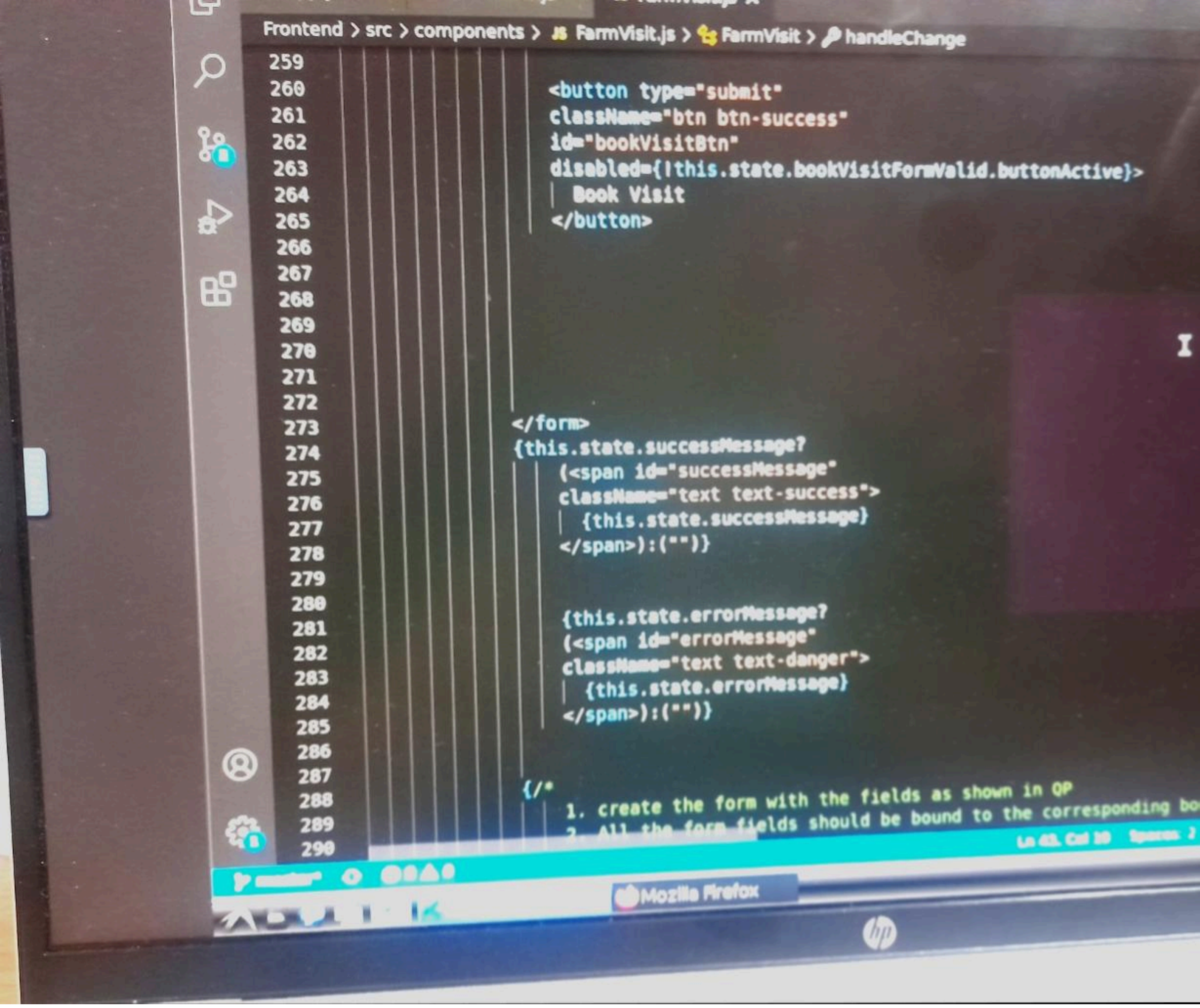

App.js
Locations.js
FarmVisit.js
Frontend > src > components > FarmVisit.js > FarmVisit > handleChange
Execute
Mozilla Firefox

```
                  name="customerName"
                  className="form-control"
                  id="customerName"
                  value={this.state.bookVisitForm.customerName}
                  onChange={this.handleChange}
                  />
                  <span id="customerNameError"
                  className="text text-danger">
                    {this.state.bookVisitFormErrorMessage.customerName}

                  </span>

                </div>
                <div className="form-group">
                  <label htmlFor="visitDate">
                  Visit Date
                  </label>
                  <input type="date"
                  name="visitDate"
                  className="form-control"
                  id="visitDate"
                  value={this.state.bookVisitForm.visitDate}
                  onChange={this.handleChange}
                  />
                  <span id="visitDateError"
                  className="alert alert-danger">
                    {this.state.bookVisitFormErrorMessage.visitDate}

                  </span>

                </div>
                <div className="form-group">
```

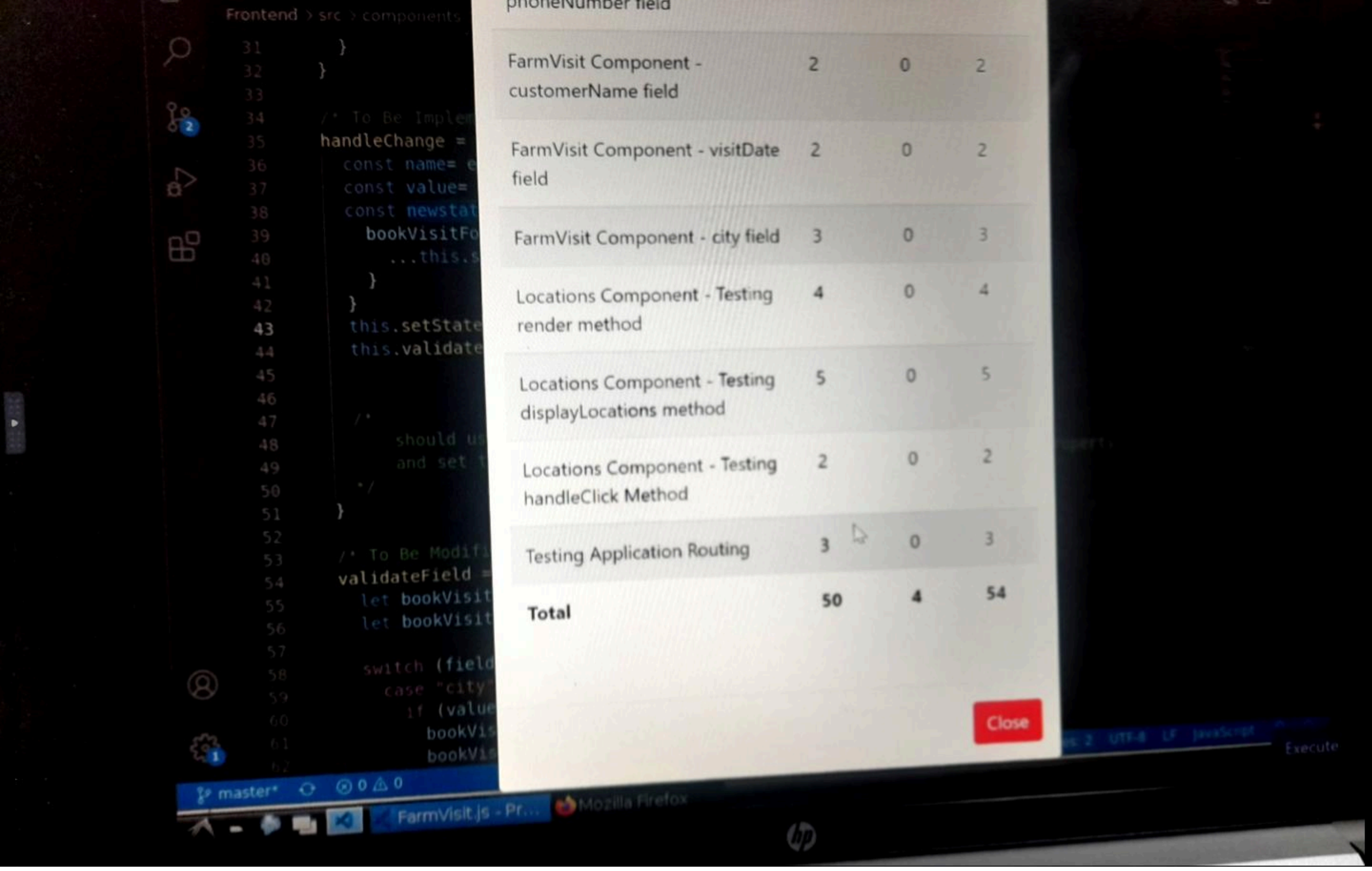

Frontend > src > components > FarmVisit.js > FarmVisit > handleChange
Execute
Mozilla Firefox

```
                  value={this.state.bookVisitForm.visitDate}
                  onChange={this.handleChange}
                  />
                  <span id="visitDateError"
                  className="alert alert-danger">
                    {this.state.bookVisitFormErrorMessage.visitDate}

                  </span>

                </div>
```

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

                  </span>

                </div>


                  <button type="submit"
                  className="btn btn-success"
```


Frontend > src > components > FarmVisit.js > FarmVisit > handleChange
Mozilla Firefox

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







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


                {this.state.errorMessage?
                (<span id="errorMessage"
                className="text text-danger">
                  {this.state.errorMessage}
                </span>):("")}


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



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**

```
    }
  }

  /* To Be Implem
  handleChange =
    const name= e
    const value=
    const newstat
      bookVisitFo
        ...this.s
      }
    }
    this.setState
    this.validate


    /*
        should us
        and set t
    */
  }

  /* To Be Modifi
  validateField =
    let bookVisit
    let bookVisit

    switch (field
      case "city"
        if (value
          bookVis
          bookVis
```