Developing BookStore FrontEnd with React

BSApp1.1-Frontend - Working Steps:

1) Create the new project called bookstore-app-frontend

```
create-react-app bookstore-app-frontend
```

2) 2) Delete the following 5 files from public folder
   a) favicon.ico
   b) logo192.png
   c) logo512.png
   d) manifest.json
   e) robots.txt

3) Delete the following 4 files from public folder
   a)App.css
   b) App.test.js
   c) logo.svg
   d) reportWebVitals.js
   e) setupTests.js

4) Enable the Jquery and BootStrap with your Application.
   a) install jquery and bootstrap

```
npm install jquery bootstrap react-bootstrap
```

   b)import the bootstrap.min.css in App.js

```
import "bootstrap/dist/css/bootstrap.min.css";
```

5) Update index.html

```
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>JLC BookStore App</title>
</head>
<body>
<noscript>You need to enable JavaScript to run this app.</noscript>
<div id="myroot"></div>
</body>
</html>
```

6) Update Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(<App />, document.getElementById('myroot'));
```

7) Nothing to update index.css as of now we will update Later

8) Update App.js

```
import React, { Component } from 'react';

import "bootstrap/dist/css/bootstrap.min.css";

class App extends Component {
render() {
return (
<div className="App">
<h1 className="text-center"> JLC BookStore </h1>
</div>
);
}
}

export default App;
```

9) Run the App

```
npm start
```

10) Install Packages required for Your Project.

```
npm install react-router-dom
npm install axios
```

11) Add the Required CSS classes in Index.css

As given in BookLet

12) Write the following Components As given in BookLet

BooksList.js
AddBook.js
EditBook.js
MyTextInput

JLCHeader.js
JLCFooter.js
JLCBody.js

13) Update App.js

As given in BookLet

14) Run the App

npm start

<u>BookStore App1.1 - Frontend</u>

<u>Files Required</u>

1. index.html	Same as BSApp1.0
2. index.js	Same as BSApp1.0
3. index.css	Same as BSApp1.0
4. App.js	Same as BSApp1.0
5. JLCHeader.js	Same as BSApp1.0
6. JLCFooter.js	Same as BSApp1.0
7. JLCBody.js	Same as BSApp1.0
8. MyTextInput.js	Same as BSApp1.0
9. BookList.js	**Updated in this App**
10. AddBook.js	**Updated in this App**
11. EditBook.js	**Updated in this App**
12. Package.json	**Updated in this App**

9. BookList.js

(Only Updated URL's , Remaining Same as BSApp1.0)

```
 getAllBooks = () => {
  const URL = "http://localhost:5500/myapi/mybooks"; //Updated URL
  console.log(URL);
  axios.get(URL).then((myresponse) => {
   console.log(myresponse.data);
   // console.log(myresponse.headers);
   this.setState({
    mybooks: myresponse.data,
   });
  });
 };

 handleDelete = () => {
  console.log("handleDelete", this.state.mybookIdToDelete);

  const URL =
`http://localhost:5500/myapi/mybooks/${this.state.mybookIdToDelete}`;
//Updated URL

  axios
   .delete(URL)
   .then((myresponse) => {
    console.log(1, myresponse.data);
    this.hideDeleteConfim();
    this.getAllBooks();
    this.props.history.push("/");
   })
   .catch((myerror) => {
    console.log(2, myerror);
   });
 };
…….
…….
```

10. AddBook.js

```
import React, { Component } from 'react';

import 'bootstrap/dist/css/bootstrap.min.css';
import MyTextInput from './MyTextInput';
import axios from 'axios';


class AddBook extends Component {

  state = {
    bookId: 0,
    bookName: '',
    author: '',
    price: 0,
    category: '',
    publications: '',
    errors: {}
  }

  onChangeHandler = (event) => {
    console.log("onChangeHandler");
    this.setState({
      [event.target.name]: event.target.value
    });
  }

  onSubmitHandler = (event) => {

    event.preventDefault();
    const { bookName, author, price, category, publications } = this.state;

    console.log("onSubmitHandler");
    console.log(this.state);
    //Do the Validations

    if (bookName === '') {
      this.setState({
        errors: {
          bookName: "Book Name is Required"
        }
      });
      return;
    }
```

```
if (author === '') {
  this.setState({
    errors: {
      author: "Author is Required"
    }
  });
  return;
}

if (price === '') {
  this.setState({
    errors: {
      price: "Price is Required"
    }
  });
  return;
}
if (category === '') {
  this.setState({
    errors: {
      category: "Category is Required"
    }
  });
  return;
}
if (publications === '') {
  this.setState({
    errors: {
      pub: "Publications is Required"
    }
  });
  return;
}

//Make Call to Server
const URL1 = `http://localhost:5500/myapi/maxId`;
console.log(12345, URL1);
axios.get(URL1).then((myresponse) => {
  console.log(12346, myresponse.data[0].bookId);
  this.setState({
    bookId: myresponse.data[0].bookId + 1,
  });
  console.log(12347, this.state);

  const URL2 = "http://localhost:5500/myapi/mybooks";
  axios.post(URL2, this.state)
```

```
        .then(myresponse => {
          console.log(1, myresponse.data);
          //Form Reset
          this.setState({
            mybook: {},
          });
          this.props.history.push("/");
        }).catch(myerror => {
          console.log(2, myerror);
        });

    });


  }

  render() {
    const { bookName, author, price, category, publications } = this.state;

    return (
      <div className="card-body container col-md-6">
        <h2 className="text-center"> Add Book Form </h2>
        <form onSubmit={this.onSubmitHandler}>


          <MyTextInput myname="bookName"
            mylabel="Book Name"
            myvalue={bookName}
            myerror={this.state.errors.bookName}
            myOnChange={this.onChangeHandler} />

          <MyTextInput myname="author"
            mylabel="Author"
            myvalue={author}
            myerror={this.state.errors.author}
            myOnChange={this.onChangeHandler} />

          <MyTextInput myname="price"
            mylabel="Price"
            myvalue={price}
            myerror={this.state.errors.price}
            myOnChange={this.onChangeHandler} />

          <MyTextInput myname="category"
            mylabel="Category"
            myvalue={category}
            myerror={this.state.errors.category}
            myOnChange={this.onChangeHandler} />

          <MyTextInput myname="publications"
            mylabel="Publications"
            myvalue={publications}
            myerror={this.state.errors.publications}
            myOnChange={this.onChangeHandler} />


          <input type="submit"
            value="Add Book Now"
            className="btn btn-primary btn-lg" />

        </form>
      </div>

    );
  }

}


export default AddBook;
```

11. EditBook.json

(Only Updated URL's , Remaining Same as BSApp1.0)

```
……
……

 componentDidMount() {
  console.log("componentDidMount", this.props);
  this.setState({
   bookId: this.props.match.params.mybookId,
  });
  Const URL =
`http://localhost:5500/myapi/mybooks/${this.props.match.params.mybookId}`;
  axios.get(URL).then((myresponse) => {
   console.log(myresponse.data);
   this.setState({
    bookId: myresponse.data.bookId,
    bookName: myresponse.data.bookName,
    author: myresponse.data.author,
    price: myresponse.data.price,
    category: myresponse.data.category,
      publications: myresponse.data.publications,
    });
  });
}

onChangeHandler = (event) => {
  console.log("onChangeHandler");
  this.setState({
    [event.target.name]: event.target.value,
  });
};

onSubmitHandler = (event) => {
  event.preventDefault();
  const { bookName, author, price, category, publications } = this.state;

  console.log("onSubmitHandler");
  console.log(this.state);
  //Do the Validations

  if (bookName === "") {
    this.setState({
      errors: {
        bookName: "Book Name is Required",
      },
    });
    return;
  }

  if (author === "") {
    this.setState({
      errors: {
        author: "Author is Required",
      },
    });
    return;
  }

  if (price === "") {
    this.setState({
      errors: {
        price: "Price is Required",
      },
    });
    return;
  }
```

```
    if (category === "") {
      this.setState({
        errors: {
          category: "Category is Required",
        },
      });
      return;
    }
    if (publications === "") {
      this.setState({
        errors: {
          pub: "Publications is Required",
        },
      });
      return;
    }

    //Make Call to Server

    const URL = "http://localhost:5500/myapi/mybooks";
    axios
      .put(URL, this.state)
      .then((myresponse) => {
        console.log(1, myresponse.data);
        //Form Reset
        this.setState({
          bookId: 0,
          bookName: "",
          author: "",
          price: 0,
          category: "",
          publications: "",
          errors: {},
        });
        this.props.history.push("/");
      })
      .catch((myerror) => {
        console.log(2, myerror);
      });
  };
……
……
```

12. package.json

```
{
 "name": "bookstore-app-frontend",
 "version": "0.1.0",
 "private": true,
 "dependencies": {
  "@testing-library/jest-dom": "^5.14.1",
  "@testing-library/react": "^11.2.7",
  "@testing-library/user-event": "^12.8.3",
  "axios": "^0.21.1",
  "bootstrap": "^5.0.2",
  "classes": "^0.3.0",
  "classnames": "^2.3.1",
  "jquery": "^3.6.0",
  "react": "^17.0.2",
  "react-bootstrap": "^1.6.1",
  "react-dom": "^17.0.2",
  "react-router-dom": "^5.2.0",
  "react-scripts": "4.0.3",
  "web-vitals": "^1.1.2"
 },
 "scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
 },
 "eslintConfig": {
  "extends": [
   "react-app",
   "react-app/jest"
  ]
 },
 "browserslist": {
  "production": [
   ">0.2%",
   "not dead",
   "not op_mini all"
  ],
  "development": [
   "last 1 chrome version",
   "last 1 firefox version",
   "last 1 safari version"
  ]
 }
}
```

Developing MERN Stack Application

BookStore MERN Stack – Working Steps

1) Copy bookstore-app-frontend as bookstore-mern-app
2) Open the folder bookstore-mern-app
3) Install the Dependencies required for your backend project.

```
npm install express
npm install body-parser
npm install mongoose
npm install nodemon
npm install dotenv
npm install cors
npm install path

npm install express  body-parser mongoose nodemon dotenv cors path
```

4) Copy the following 2 files under bookstore-mern-app
   a) .env.jlc
   b) myserver.jlc

5) Create the folder called myserver under bookstore-mern-app

6) Copy the following two folders under myserver
   a)mymodels/Book.js
   b)mycontrollers/BookController.js

7) Update myserver.js

```
app.use(express.static(path.join(__dirname, "build")));

……

 app.get("*", (req, res) => {
   res.sendFile(
    path.join(__dirname, "build", "index.html")
   );
 });
```

8) Update package.json

```
{
 "name": "bookstore-mern-app",
 "version": "1.1.1",
 "description": "This is the MERN App By Dande",
 "main": "myserver.js",
 "private": true,
 "keywords": [
   "NodeJS",
   "ExpressJS",
   "MongoDB",
   "Angular"
 ],
 "author": "Srinivas Dande",
 "license": "ISC",
 "dependencies": {
         ....
 },
 "scripts": {
   "start": "react-scripts start",
   "build": "react-scripts build",
   "test": "react-scripts test",
   "eject": "react-scripts eject",
   "server": "nodemon --watch myserver.js"
 },
```

9) Build Project

```
npm run build
```

10) Run the App

```
npm run server
or
 node myserver.js
```

11) Open http://localhost:5500/

BookStore App - MERN Stack

Files Required

1. index.html	2. index.js
3. index.css	4. App.js
5. JLCHeader.js	6. JLCFooter.js
7. JLCBody.js	8. MyTextInput.js
9. BookList.js	10. AddBook.js
11. EditBook.js	12. .env.jlc
13.myserver.js	14. Book.js
15. BookController.js	**16.package.json**

Note : Copy the Above Files from BookSTore Front End and BookSTore Backend Apps developed in the Previous Sessions.

Only Two Files Updated – myserver.js and packag.json

13.myserver.js

```
const express = require('express');
const bodyParser = require('body-parser');
const dotenv = require('dotenv');
const mongoose = require('mongoose');
const path = require('path');
const cors = require("cors");

const BookController = require('./myserver/mycontrollers/BookController');

dotenv.config({    path: '.env.jlc'  });

const app = express();
const PORT = process.env.PORT || 5500;

/**   * Connect to MongoDB.   */
mongoose.connect(process.env.MONGODB_URI, {
  useUnifiedTopology: true,
  useFindAndModify: false,
  useNewUrlParser: true,
}).then(() => {
  console.log('MongoDB connected successfully.');
}, error => {
  console.error(error);
  console.log('MongoDB connection error. Please make sure MongoDB is running.');
  process.exit();
});

/**   * Add middlewares   */
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
  extended: false
}));

app.use(express.static(path.join(__dirname, "build")));
app.use(cors());

app.get('/hello', (req, res) => {
  return res.send('Hello Guys')
})

app.get('/myapi/mybooks', BookController.getAllBooks);
app.get('/myapi/mybooks/:bookId', BookController.findBookByBookId);
app.get('/myapi/maxId', BookController.findMaxBookId);
app.post('/myapi/mybooks', BookController.addBook);
app.put('/myapi/mybooks', BookController.updateBook);
app.delete('/myapi/mybooks/:bookId', BookController.deleteBook);

app.get("*", (req, res) => {
  res.sendFile(
    path.join(__dirname, "build", "index.html")
  );
});

// start express server on port 3001
app.listen(PORT, () => {
  console.log('Express server is running at http://localhost:%d', PORT);
  console.log('Press CTRL-C to stop\n');
});
```

16.package.json

```
{
 "name": "bookstore-mern-app",
 "version": "1.1.1",
 "description": "This is the MERN App By Dande",
 "main": "myserver.js",
 "private": true,
 "keywords": [
  "NodeJS",
  "ExpressJS",
  "MongoDB",
  "Angular"
 ],
 "author": "Srinivas Dande",
 "license": "ISC",
 "dependencies": {
  "@testing-library/jest-dom": "^5.14.1",
  "@testing-library/react": "^11.2.7",
  "@testing-library/user-event": "^12.8.3",
  "axios": "^0.21.1",
  "body-parser": "^1.19.0",
  "bootstrap": "^5.0.2",
  "classes": "^0.3.0",
  "classnames": "^2.3.1",
  "cors": "^2.8.5",
  "dotenv": "^10.0.0",
  "express": "^4.17.1",
  "jquery": "^3.6.0",
  "mongoose": "^5.13.3",
  "nodemon": "^2.0.12",
  "path": "^0.12.7",
  "react": "^17.0.2",
  "react-bootstrap": "^1.6.1",
  "react-dom": "^17.0.2",
  "react-router-dom": "^5.2.0",
  "react-scripts": "4.0.3",
  "web-vitals": "^1.1.2"
 },
 "scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject",
  "server": "nodemon --watch myserver.js"
 },
 "eslintConfig": {
  "extends": [
   "react-app",
   "react-app/jest"
  ]
 },
 "browserslist": {
  "production": [
   ">0.2%",
   "not dead",
   "not op_mini all"
  ],
  "development": [
   "last 1 chrome version",
   "last 1 firefox version",
   "last 1 safari version"
  ]
 }
}
```