

## Alphabetiser App

### Requirement

Create a web app that allows the user to input two or more text items and then order them alphabetically.

The app should contain:

- two input fields
- a button to add a new input field
- a button to re-order the existing fields alphabetically

If there are more than two input fields, a delete button should be appended to each field.

### Application Design

Up on the listed requirement, there is NO data storage in server side. Only a web front end application to serve input values and perform alphabetic sorting. Suggest to develop an web application with a standard javascript framework (React.JS) which is able to serve all the required functions.

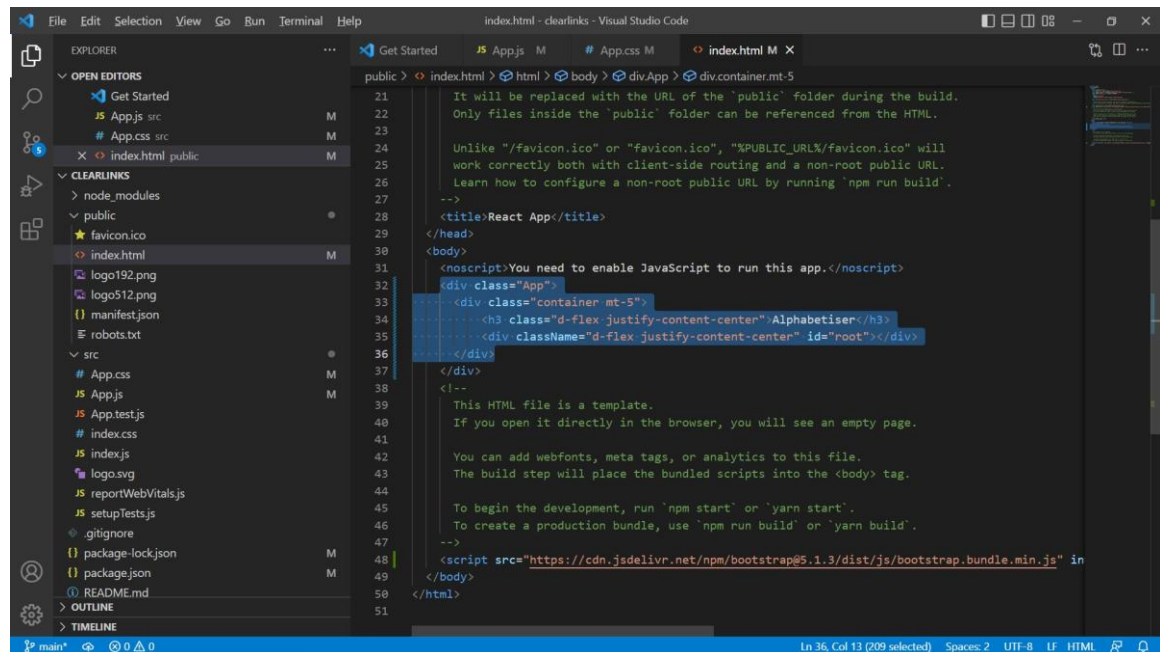
### Development Environment

1. Install Node.JS <https://nodejs.org/en/download/>
2. Create React component by **npx create-react-app alphabetiser**
3. Make reference to Bootstrap V5.1 <https://getbootstrap.com/docs/5.1/getting-started/introduction/>
4. Test run by CLI command **npm start**  
System will open [http://localhost:3000] to view it in your default browser.

*Continue next page*

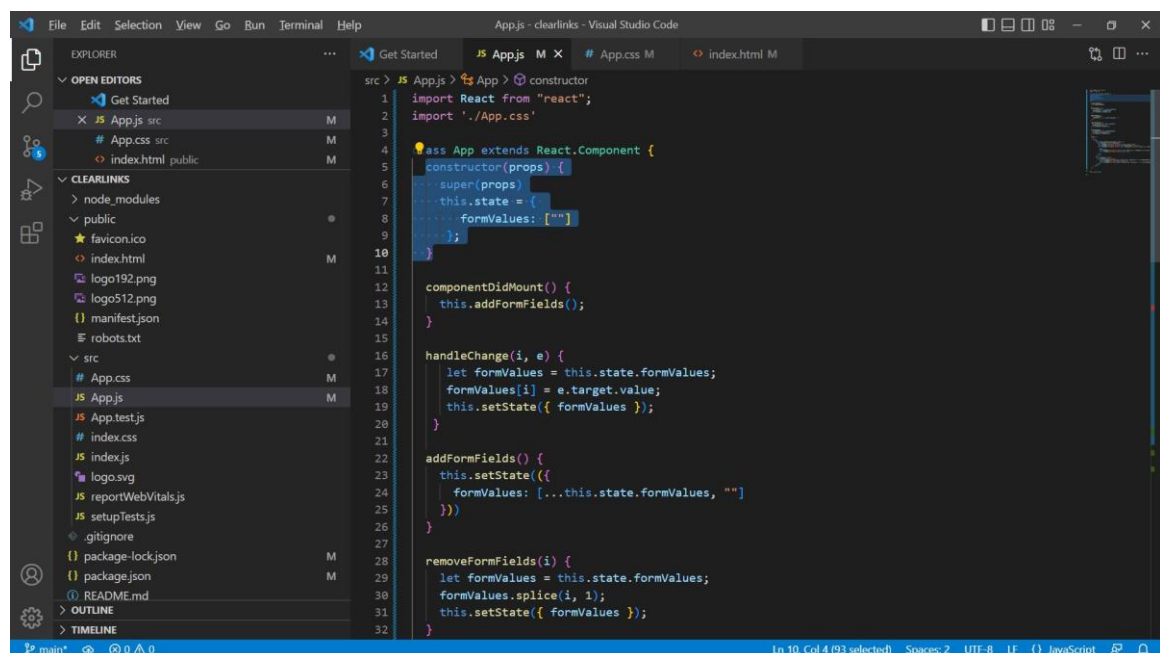
## Development details

### Modify the index.html



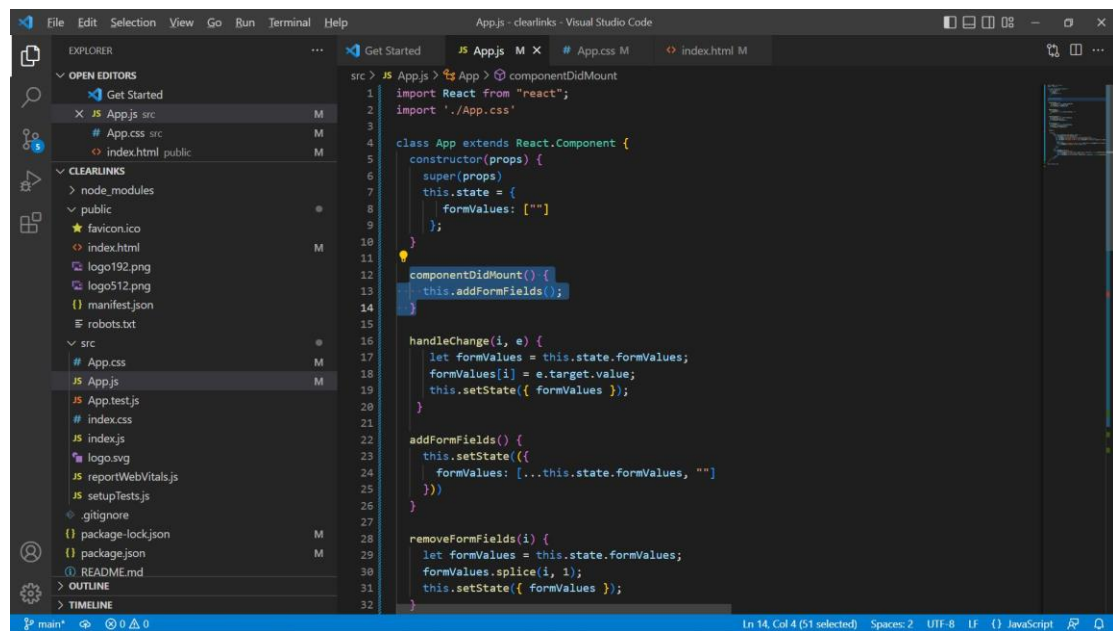
Add a <div> as container for the user interface.

### Modify App.js



Initial an array to store the value from user input

Ready for additional field when program startup

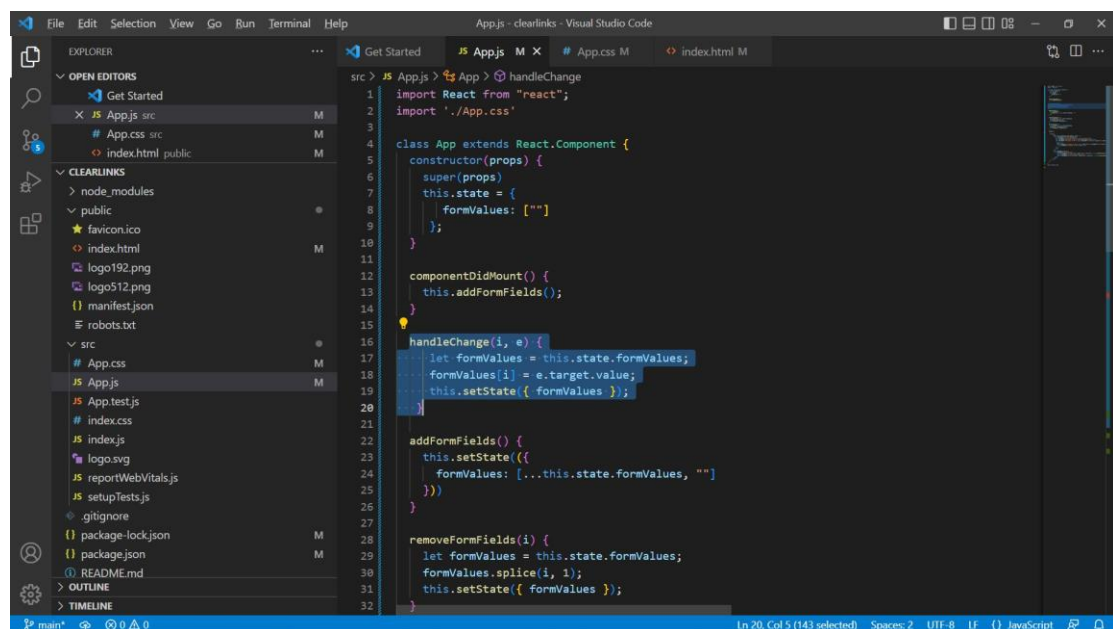


The screenshot shows the Visual Studio Code editor with the 'App.js' file open. The 'componentDidMount' method is highlighted in blue. The code in the file is as follows:

```
1 import React from "react";
2 import './App.css'
3
4 class App extends React.Component {
5   constructor(props) {
6     super(props);
7     this.state = {
8       formValues: [""]
9     };
10  }
11
12  componentDidMount() {
13    this.addFormFields();
14  }
15
16  handleChange(i, e) {
17    let formValues = this.state.formValues;
18    formValues[i] = e.target.value;
19    this.setState({ formValues });
20  }
21
22  addFormFields() {
23    this.setState({
24      formValues: [...this.state.formValues, ""]
25    });
26  }
27
28  removeFormFields(i) {
29    let formValues = this.state.formValues;
30    formValues.splice(i, 1);
31    this.setState({ formValues });
32  }
33 }
```

Calling function to AddFormFields() after page render

Function to handle user input

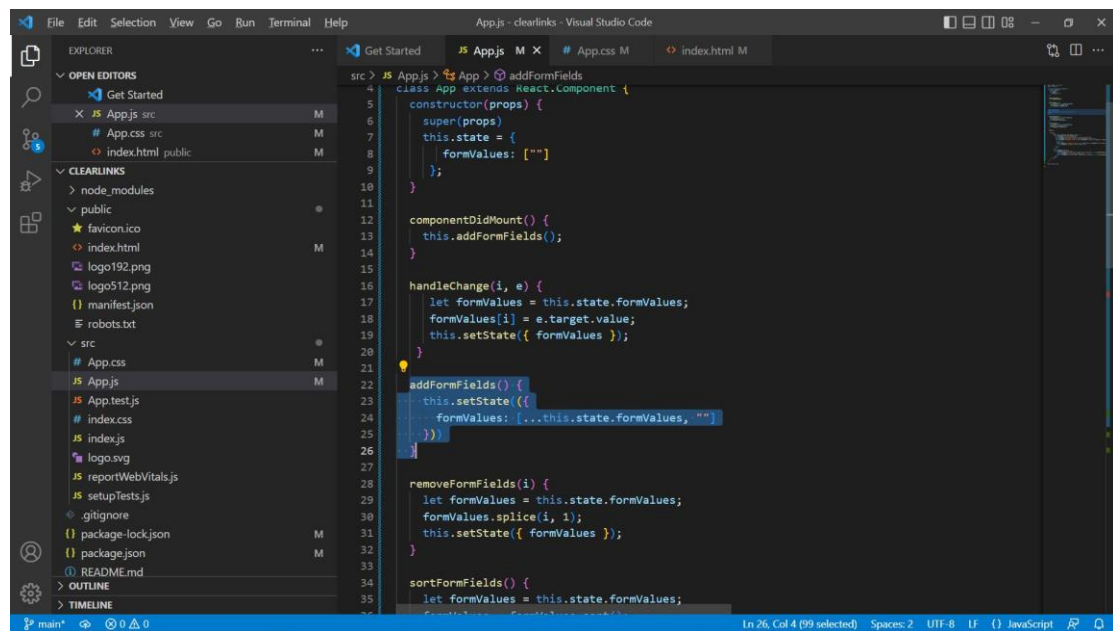


The screenshot shows the Visual Studio Code editor with the 'App.js' file open. The 'handleChange' method is highlighted in blue. The code in the file is as follows:

```
1 import React from "react";
2 import './App.css'
3
4 class App extends React.Component {
5   constructor(props) {
6     super(props);
7     this.state = {
8       formValues: [""]
9     };
10  }
11
12  componentDidMount() {
13    this.addFormFields();
14  }
15
16  handleChange(i, e) {
17    let formValues = this.state.formValues;
18    formValues[i] = e.target.value;
19    this.setState({ formValues });
20  }
21
22  addFormFields() {
23    this.setState({
24      formValues: [...this.state.formValues, ""]
25    });
26  }
27
28  removeFormFields(i) {
29    let formValues = this.state.formValues;
30    formValues.splice(i, 1);
31    this.setState({ formValues });
32  }
33 }
```

When user input value in textbox, a change event will activate a function to store the input value into an array.

Provide [Add] button to add additional input box

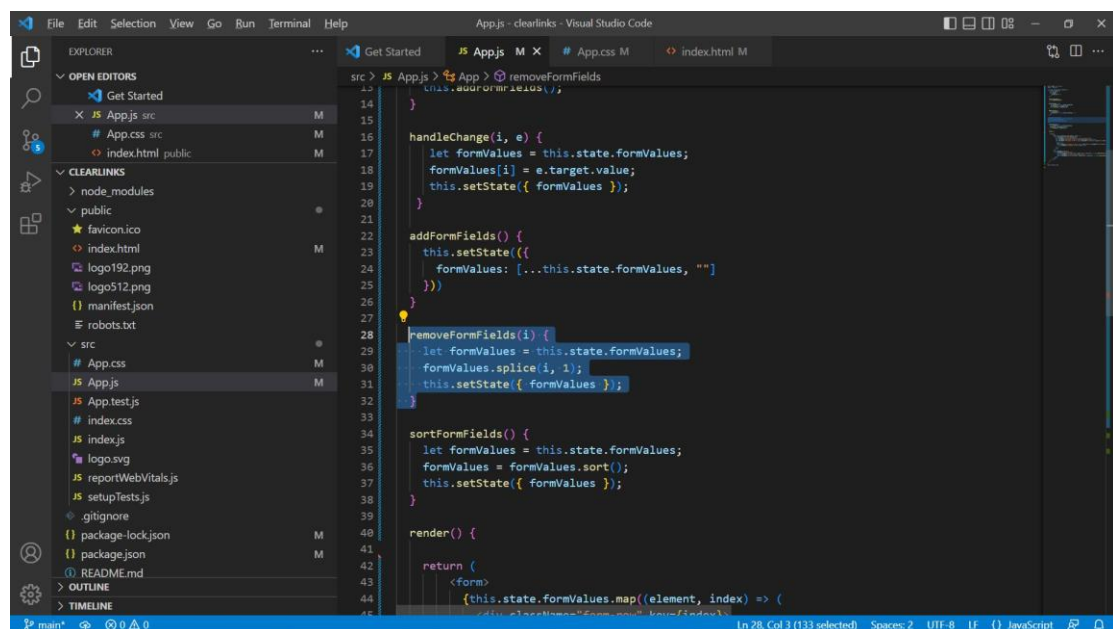


The screenshot shows the Visual Studio Code editor with the 'App.js' file open. The 'EXPLORER' sidebar on the left shows the project structure, including 'src' and 'public' folders. The main editor area displays the following code:

```
src > JS App.js > App > addFormFields
4  class App extends React.Component {
5    constructor(props) {
6      super(props);
7      this.state = {
8        formValues: []
9      };
10   }
11
12   componentDidMount() {
13     this.addFormFields();
14   }
15
16   handleChange(i, e) {
17     let formValues = this.state.formValues;
18     formValues[i] = e.target.value;
19     this.setState({ formValues });
20   }
21
22   addFormFields() {
23     this.setState({
24       formValues: [...this.state.formValues, ""]
25     });
26   }
27
28   removeFormFields(i) {
29     let formValues = this.state.formValues;
30     formValues.splice(i, 1);
31     this.setState({ formValues });
32   }
33
34   sortFormFields() {
35     let formValues = this.state.formValues;
```

System should add a empty value into an array.

Provide [Remove] button for delete input box



The screenshot shows the Visual Studio Code editor with the 'App.js' file open. The 'EXPLORER' sidebar on the left shows the project structure. The main editor area displays the following code:

```
src > JS App.js > App > removeFormFields
14  this.addFormFields();
15  }
16
17  handleChange(i, e) {
18    let formValues = this.state.formValues;
19    formValues[i] = e.target.value;
20    this.setState({ formValues });
21  }
22
23  addFormFields() {
24    this.setState({
25      formValues: [...this.state.formValues, ""]
26    });
27  }
28
29  removeFormFields(i) {
30    let formValues = this.state.formValues;
31    formValues.splice(i, 1);
32    this.setState({ formValues });
33  }
34
35  sortFormFields() {
36    let formValues = this.state.formValues;
37    formValues = formValues.sort();
38    this.setState({ formValues });
39  }
40
41  render() {
42    return (
43      <form>
44        {this.state.formValues.map((element, index) => (
45          <div className="form-input"> {element}</div>
```

Function to serve remove textbox and application should remove the same element from an array.

Provide [Sort] button to sort input box value in alphabetical order

The screenshot shows a VS Code editor window with the following components:

- Explorer Sidebar (Left):** Displays the project file structure.
  - src** (Folder): Contains `App.js`, `App.test.js`, `index.css`, `index.js`, `logo.svg`, `reportWebVitals.js`, `setupTests.js`, `gitignore`, `package-lock.json`, and `package.json`.
  - public** (Folder): Contains `favicon.ico`, `index.html`, `logo192.png`, `logo512.png`, and `manifest.json`.
  - node\_modules** (Folder): A sub-folder under `src`.
- App.js (Open File):** Contains the main application logic.
  - State:** `formValues` is an array of objects, each with `label` and `value` properties.
  - sortFormFields:** A function that sorts the `formValues` array by `label` and updates the state.
  - render:** A function that returns a `Form` component. It maps over `formValues` to create multiple rows. Each row contains a label, an input field, and a button. The buttons are styled as `btn btn-primary` or `btn btn-secondary` based on the row index.
- Bottom Bar:** Shows the current file is `App.js` at line 38, column 4 (137 selected). The status bar also indicates the file is a JavaScript file.

Sort an array by javascript array.sort() function.

Show user interface to end user

The screenshot shows a VS Code editor window titled 'App.js - clearlinks - Visual Studio Code'. The Explorer sidebar on the left displays the project structure:

- EXPLORER
  - OPEN EDITORS
    - Get Started
    - App.js src
    - App.css src
    - index.html public
  - CLEARLINKS
    - node\_modules
    - public
      - favicon.ico
      - index.html
      - logo192.png
      - logo512.png
      - manifest.json
      - robots.txt
    - src
      - App.css
      - App.js
      - App.test.js
      - index.css
      - index.js
      - logo.svg
      - reportWebVitals.js
      - setupTests.js
      - .gitignore
      - package-lock.json
      - package.json
      - README.md

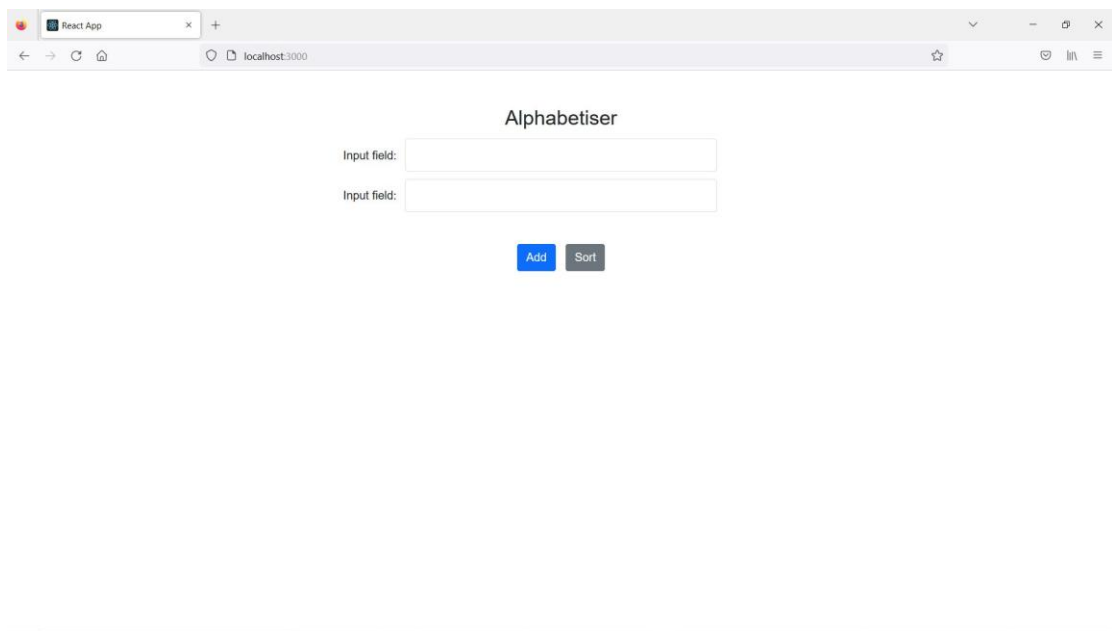
The main editor area shows the 'App.js' file with the following code:

```

src > JS App.js > App > render
40  render() {
41
42    return (
43      <Form>
44        {this.state.formValues.map((element, index) => {
45          <div className="form-row" key={index}>
46            <div className="form-group form-inline justify-content-center">
47              <div className="form-group col-md-1"><label>Input field:</label></div>
48              <div className="form-group col-md-4"><input className="form-control" type="text" na
49            </div>
50            {index > 1 ?
51              <div className="form-group col-md-1"><button type="button" className="btn btn-dan
52                : <div className="form-group col-md-1">&nbsp;&nbsp;&nbsp;</div>
53            </div>
54          </div>
55        </div>
56      </div>
57    <div className="form-row">
58      <div className="button-section">
59        <button className="btn btn-primary" type="button" onClick={() => this.addFormFiel
60          <span className="mr-3">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</span>
61        <button className="btn btn-secondary" type="button" onClick={() => this.sortFormF
62      </div>
63    </div>
64  </form>
65  );
66  }
67  }
68  }
69  export default App;
70
71
```

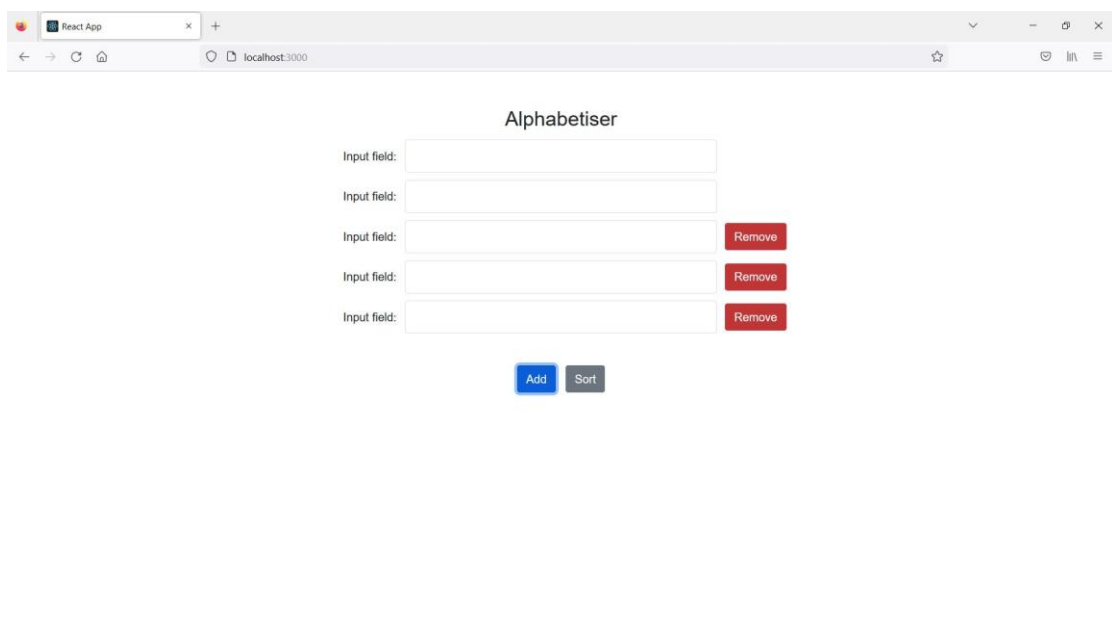
Render the web page for end user.

## User Interface



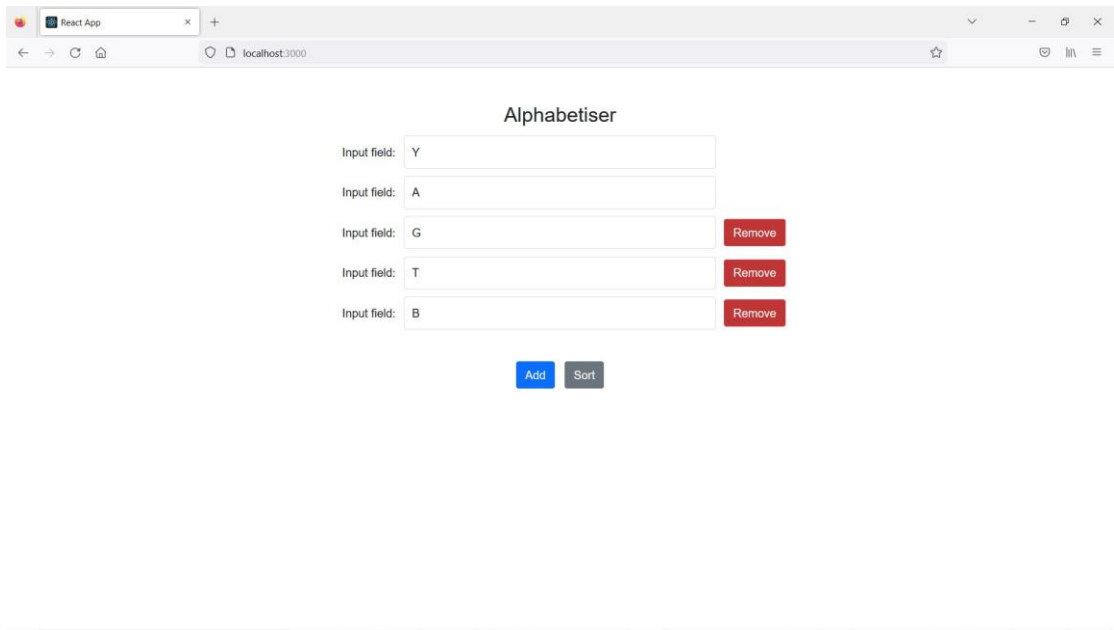
Shows two input field after application startup. (requirement 1)

Provide [Add] button at bottom

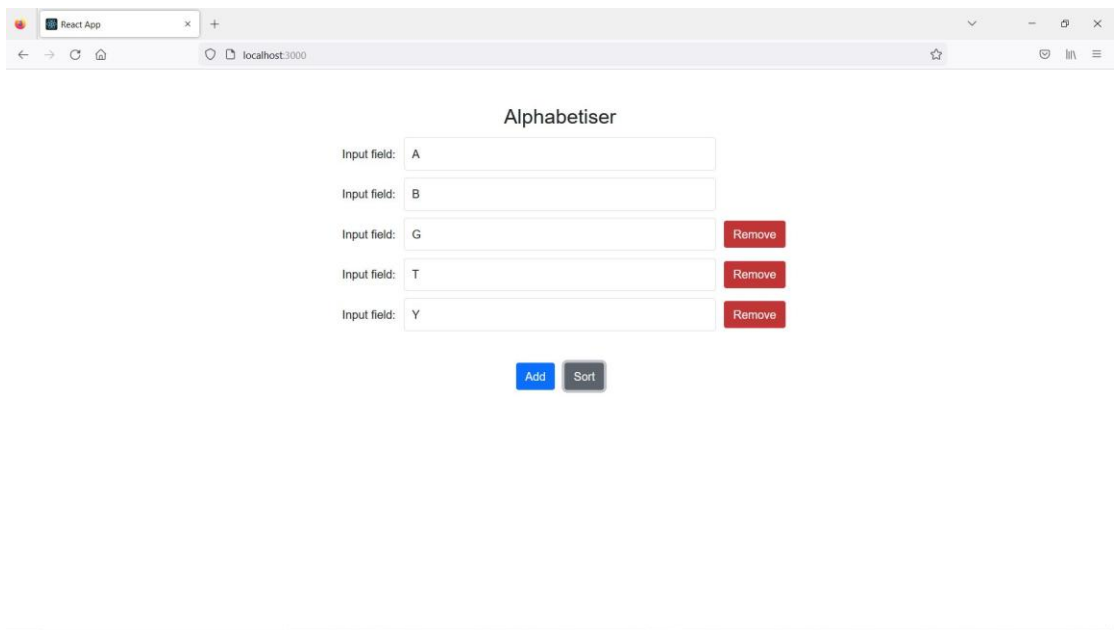


Add as many as input field that user like to have. (requirement 2)

Allows to input any value in input fields



Press [Sort] button to reorder existing field in alphabetical order



(requirement 3)

End of document