

Arpit Singh

19BCG10069

Kaiburr – Task 4

Technical Task

(Placement)

GitHub Link: <https://github.com/TSM-ArpitSG/Kaiburr/tree/main/Kaibur%20Tasks/Task%204>

Task 4:

WEB UI Forms:

Create a basic WEB UI frontend for an application that you created for **Task #1** or #2 using any UI framework of your choice. You should be able to create, show and delete records from your UI.

Steps:

1. Choose a UI framework:

- You can choose any **UI framework** of your choice like React, Angular, Vue.js, etc. For this task, we will use **ReactJS**.

2. Set up a new React project:

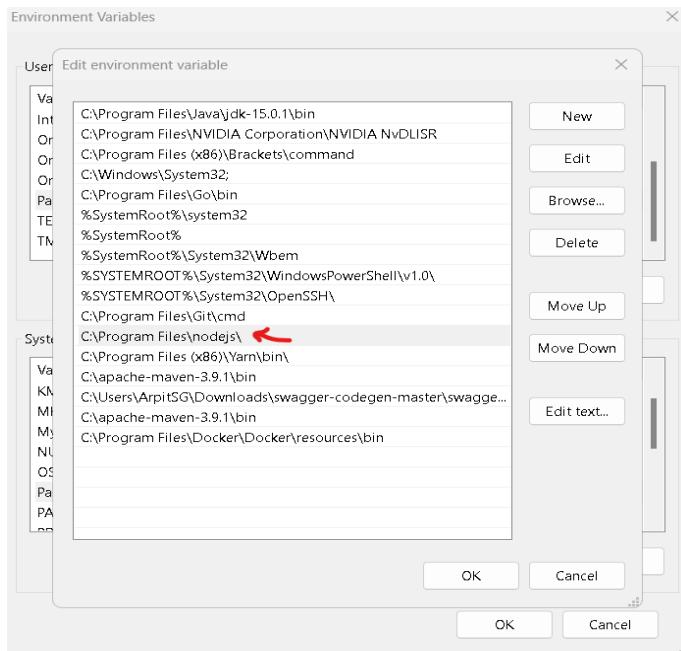
a. Installing Node.js:

- Go to the official **Node.js website at** <https://nodejs.org/>
- Click on the "**Downloads**" button on the home page.
- Select the '**version of Node.js**' that you want to download. If you're not sure which version to download, choose the latest version.
- Choose the appropriate installer based on your '**operating system**' (e.g., **Windows**, macOS, Linux, etc.).
- "Run the installer"** and follow the installation wizard's prompts to complete the installation.



- vi. Setup the **System (Environment) variable** as per the desired directory.

For Example: “**C:\Program Files\nodejs**”.



- vii. Once the installation is complete, open a terminal or command prompt and **type "node -v" to verify that Node.js is installed** correctly. You should see the version number of Node.js printed in the terminal.

❖ Successful Installation can be checked as follows:

```
Administrator: Command Pro + 
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ArpitSG>node -v
v16.15.0

C:\Users\ArpitSG>
```

A screenshot of a Command Prompt window titled 'Administrator: Command Pro'. The window shows the following text:
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.
C:\Users\ArpitSG>node -v
v16.15.0
C:\Users\ArpitSG>
The text 'v16.15.0' is underlined with a red line.

b. **Creating a new REACT APP:**

- i. ‘**Navigate**’ to the Directory where you want to create the React Application. Ex: “**E:\Kaibur_Tasks\Task_4**”

- ii. Run the following **command to install create-react-app**:

```
npm install -g create-react-app.
```

- iii. Run the following **command to create a new React project**:

```
npx create-react-app my-app
```

- iv. This will create a new React project in a folder named “**my-app**”.

- v. **Open** this React Project with an IDE of your choice (**IntelliJ**).

c. **Install necessary packages:**

- i. You will need to “**install some additional packages**” to use **React with your REST API**. Open the terminal and **navigate to the project directory** using the following command:

```
cd E:\Kaibur_Tasks\Task_4\my-app
```

- ii. Run the following command to install the required packages:

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

- iii. The “**axios**” package is used to make **HTTP requests to your REST API**, and the “**react-router-dom**” package is used for **client-side routing**.

- iv. Okay, after this we are ready to make our “**React UI Components**”

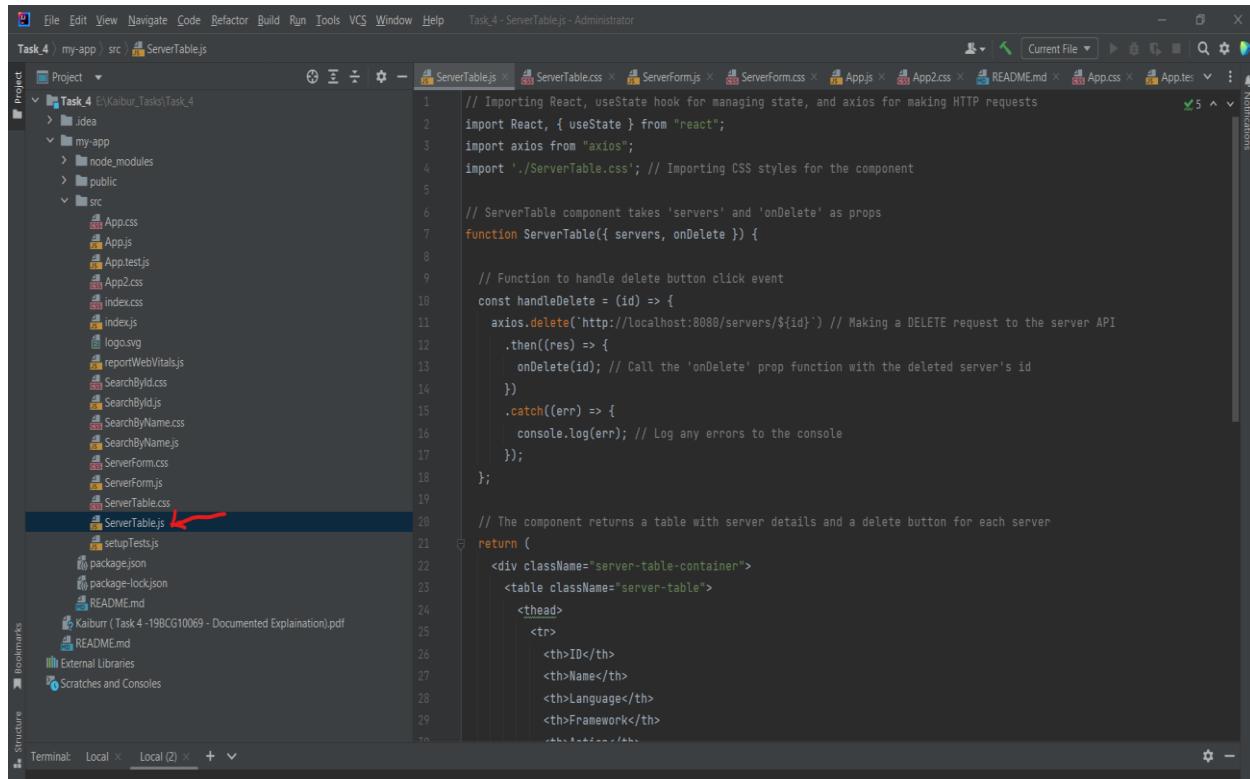
3. **Create UI Components:**

- **Note: All the Files discussed in this step will be created in the “src” directory of “my-app” React Project. Ex: “E:\Kaibur_Tasks\Task_4\my-app\src”.**
- **Proper Logical comments have been added in each code file for better understanding.**



a. ***ServerTable.js(logic):***

i. Create a File named ‘***ServerTable.js***’ with the *following code*:



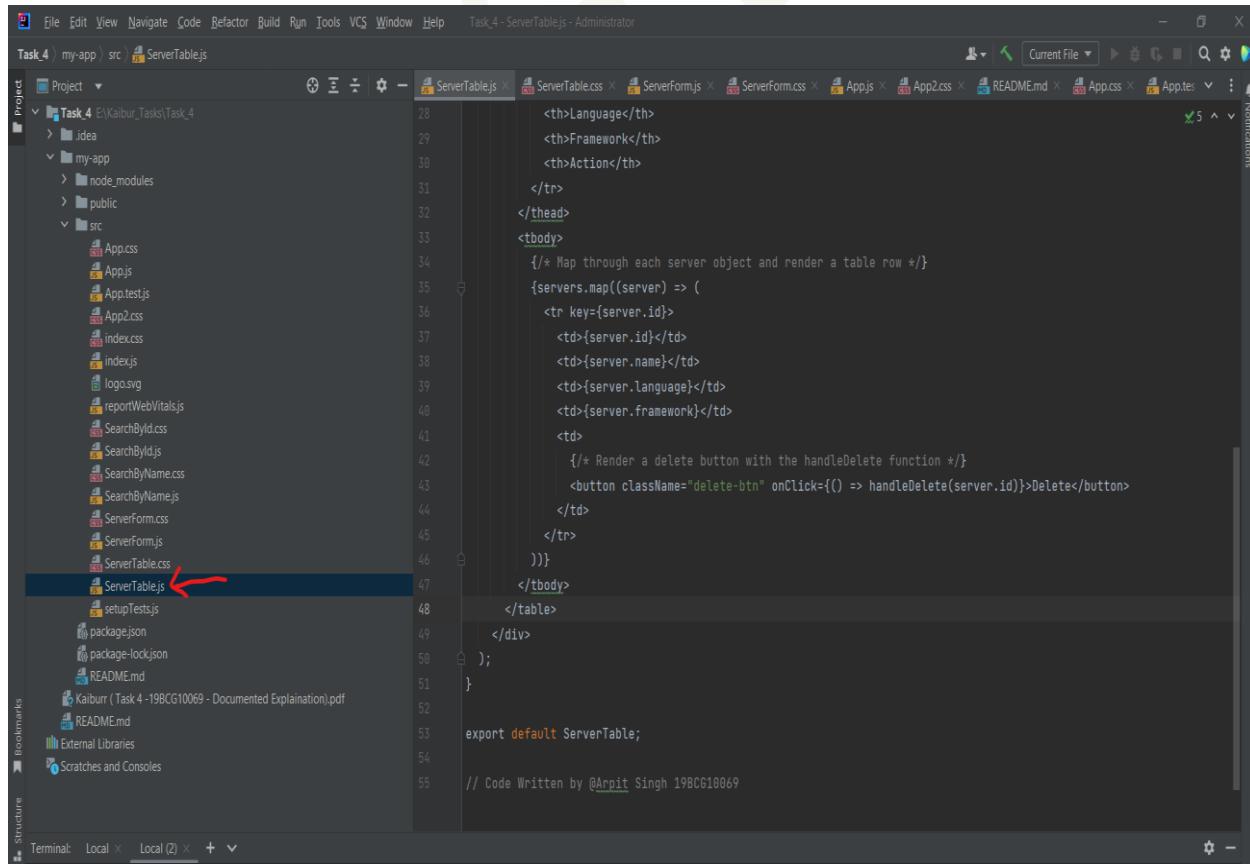
```
// Importing React, useState hook for managing state, and axios for making HTTP requests
import React, { useState } from "react";
import axios from "axios";
import './ServerTable.css'; // Importing CSS styles for the component

// ServerTable component takes 'servers' and 'onDelete' as props
function ServerTable({ servers, onDelete }) {

    // Function to handle delete button click event
    const handleDelete = (id) => {
        axios.delete(`http://localhost:8080/servers/${id}`) // Making a DELETE request to the server API
            .then((res) => {
                onDelete(id); // Call the 'onDelete' prop function with the deleted server's id
            })
            .catch((err) => {
                console.log(err); // Log any errors to the console
            });
    };

    // The component returns a table with server details and a delete button for each server
    return (
        <div className="server-table-container">
            <table className="server-table">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Name</th>
                        <th>Language</th>
                        <th>Framework</th>
                        <th>Action</th>
                    </tr>
                </thead>
                <tbody>
                    {/* Map through each server object and render a table row */}
                    {servers.map((server) => (
                        <tr key={server.id}>
                            <td>{server.id}</td>
                            <td>{server.name}</td>
                            <td>{server.language}</td>
                            <td>{server.framework}</td>
                            <td>
                                {/* Render a delete button with the handleDelete function */}
                                <button className="delete-btn" onClick={() => handleDelete(server.id)}>Delete</button>
                            </td>
                        </tr>
                    ))}
                </tbody>
            </table>
        </div>
    );
}

export default ServerTable;
```



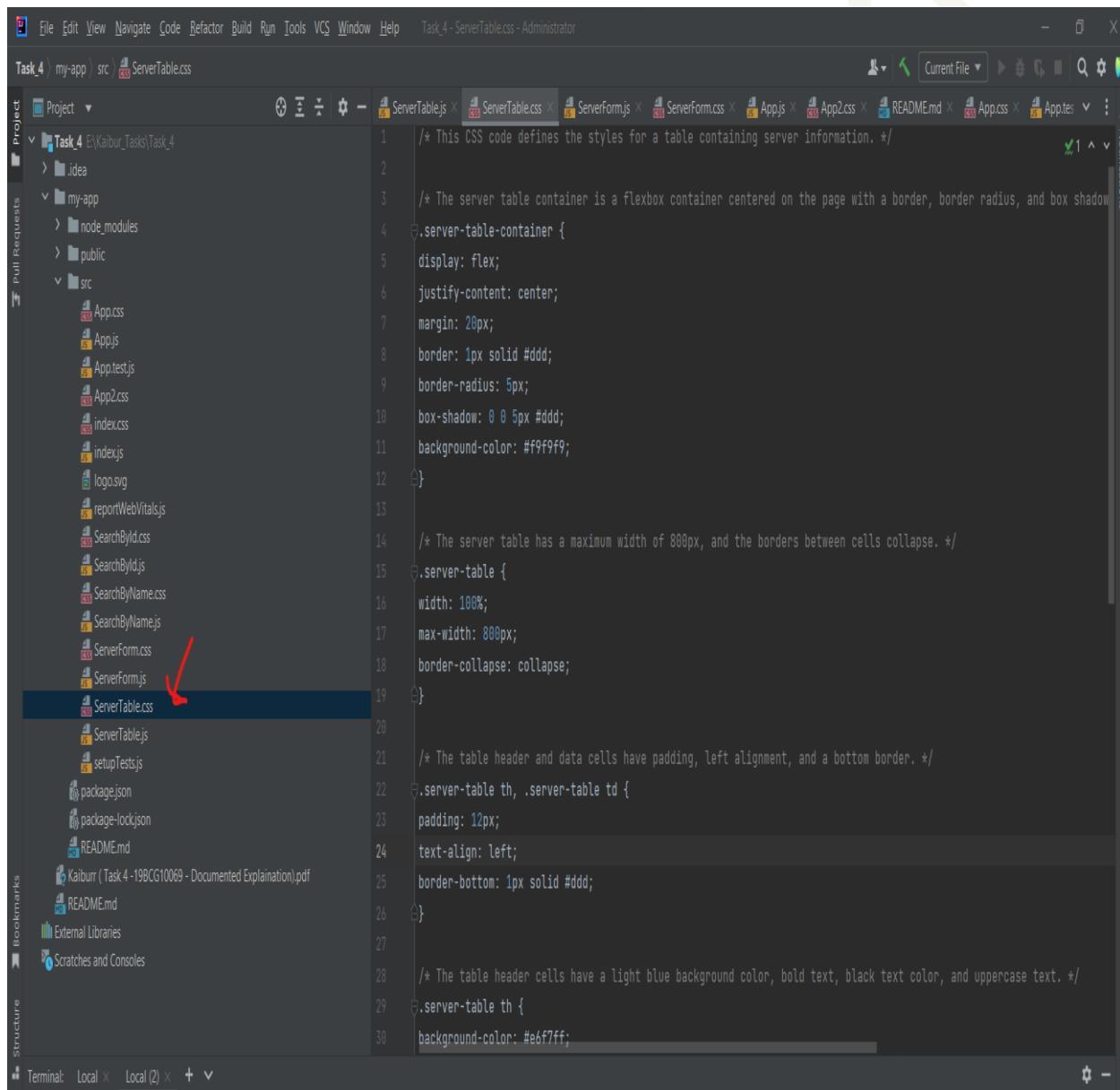
```
<th>Language</th>
<th>Framework</th>
<th>Action</th>
</tr>
</thead>
<tbody>
    {/* Map through each server object and render a table row */}
    {servers.map((server) => (
        <tr key={server.id}>
            <td>{server.id}</td>
            <td>{server.name}</td>
            <td>{server.language}</td>
            <td>{server.framework}</td>
            <td>
                {/* Render a delete button with the handleDelete function */}
                <button className="delete-btn" onClick={() => handleDelete(server.id)}>Delete</button>
            </td>
        </tr>
    ))}
</tbody>
</table>
</div>
);

export default ServerTable;
```

- In summary, **this component takes in an *array of servers as props*** and renders a table with server details and a **delete button** for each server. When the delete button is clicked, a ***DELETE request is sent to the server API*** to delete the corresponding server, and the '**onDelete**' prop function is called to ***update the list of servers***.

b. **ServerTable.css(style):**

- i. Create a File named '**ServerTable.css**' with the ***following code:***

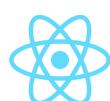


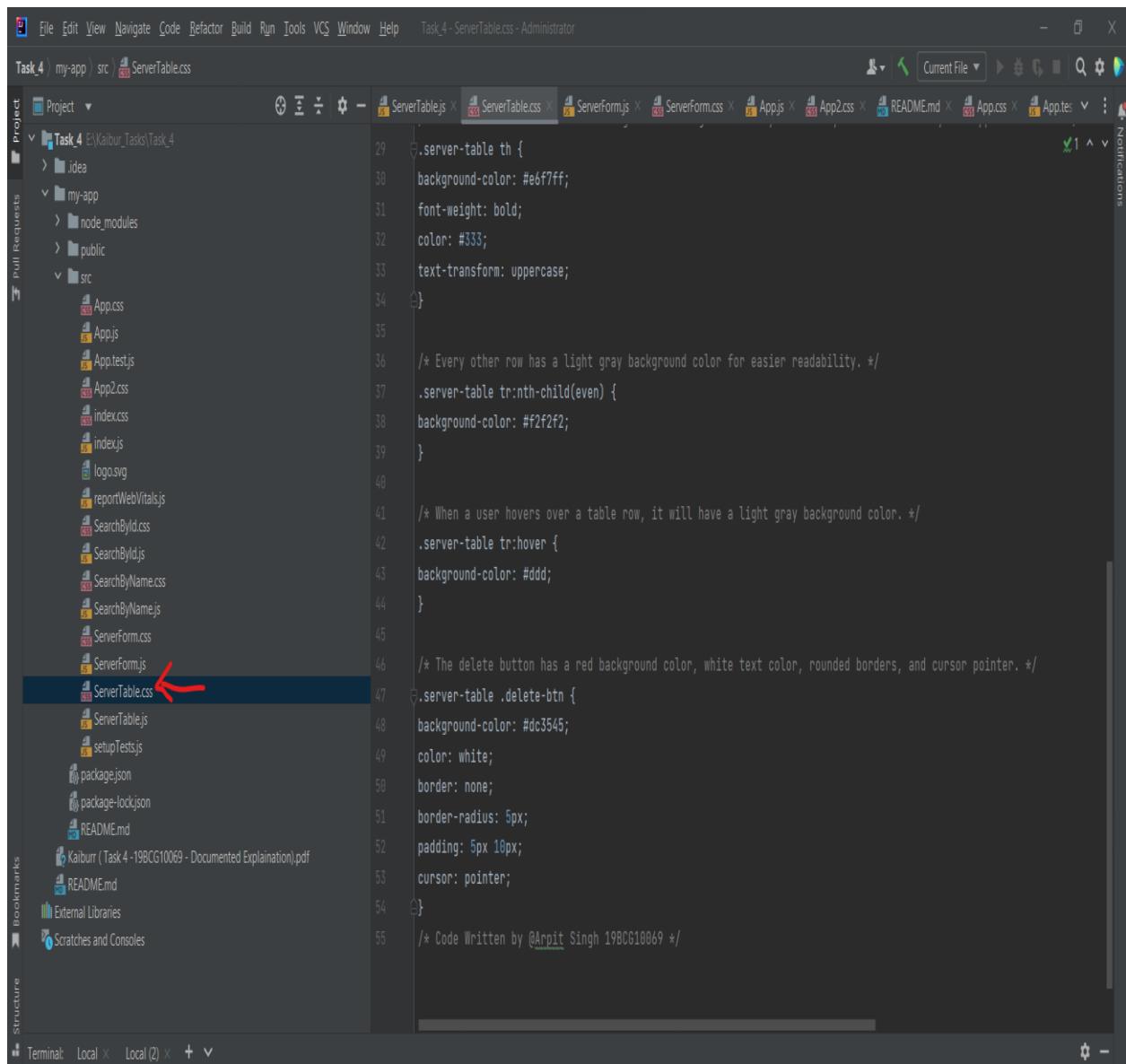
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - ServerTable.css - Administrator
Task_4 my-app src ServerTable.css
Project Current File
Task_4 E:\Kalibur\Tasks\Task_4
  > idea
  > my-app
    > node_modules
    > public
  > src
    App.css
    App.js
    App.test.js
    App2.css
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    SearchById.css
    SearchByDjs.css
    SearchByName.css
    SearchByName.js
    ServerForm.css
    ServerForm.js
    ServerTable.css
    ServerTable.js
    setupTests.js
    package.json
    package-lock.json
    README.md
    Kalibur (Task 4-19BCG10069 - Documented Explanation).pdf
    README.md
    External Libraries
    Scratches and Consoles
Structure Terminal Local Local (2) + 
  
```

```

1  /* This CSS code defines the styles for a table containing server information. */
2
3  /* The server table container is a flexbox container centered on the page with a border, border radius, and box shadow. */
4  .server-table-container {
5    display: flex;
6    justify-content: center;
7    margin: 20px;
8    border: 1px solid #ddd;
9    border-radius: 5px;
10   box-shadow: 0 0 5px #ddd;
11   background-color: #f9f9f9;
12 }
13
14  /* The server table has a maximum width of 800px, and the borders between cells collapse. */
15  .server-table {
16    width: 100%;
17    max-width: 800px;
18    border-collapse: collapse;
19 }
20
21  /* The table header and data cells have padding, left alignment, and a bottom border. */
22  .server-table th, .server-table td {
23    padding: 12px;
24    text-align: left;
25    border-bottom: 1px solid #ddd;
26 }
27
28  /* The table header cells have a light blue background color, bold text, black text color, and uppercase text. */
29  .server-table th {
30    background-color: #e6f7ff;
  
```

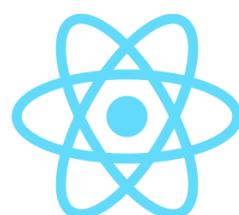




```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - ServerTable.css - Administrator
Task_4 my-app src ServerTable.css
Project Task_4 E:\KaiBurj\Tasks\Task_4
> idea
my-app
> node_modules
> public
src
App.css
App.js
App.test.js
App2.css
index.css
index.js
logo.svg
reportWebVitals.js
SearchByld.css
SearchByld.js
SearchByName.css
SearchByName.js
ServerForm.css
ServerForm.js
ServerTable.css
ServerTable.js
setupTests.js
package.json
package-lock.json
README.md
Kaiburr (Task 4-19BCG10069 - Documented Explanation).pdf
README.md
External Libraries
Scratches and Consoles
Terminal Local Local (2) +
```

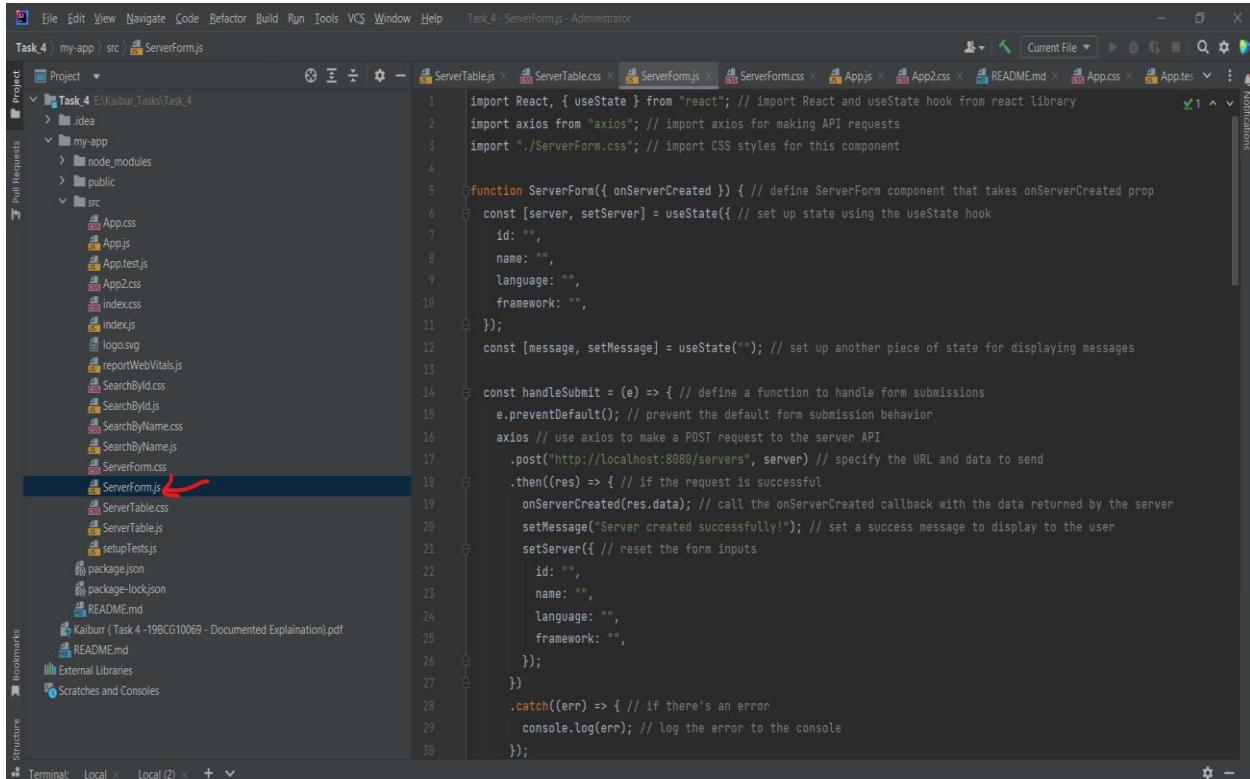
```
29 .server-table th {
30   background-color: #e6f7ff;
31   font-weight: bold;
32   color: #333;
33   text-transform: uppercase;
34 }
35
36 /* Every other row has a light gray background color for easier readability. */
37 .server-table tr:nth-child(even) {
38   background-color: #f2f2f2;
39 }
40
41 /* When a user hovers over a table row, it will have a light gray background color. */
42 .server-table tr:hover {
43   background-color: #ddd;
44 }
45
46 /* The delete button has a red background color, white text color, rounded borders, and cursor pointer. */
47 .server-table .delete-btn {
48   background-color: #dc3545;
49   color: white;
50   border: none;
51   border-radius: 5px;
52   padding: 5px 10px;
53   cursor: pointer;
54 }
55
/* Code Written by @Arpit Singh 19BCG10069 */
```

- In summary, This CSS code provides styling for a server table component. It defines the **layout and appearance** of the table and its rows, including colors, fonts, and borders. It also adds **a hover effect to the rows and a delete button with a red background color**. The container for the table has a **flex display, is centred, and has a box shadow**.

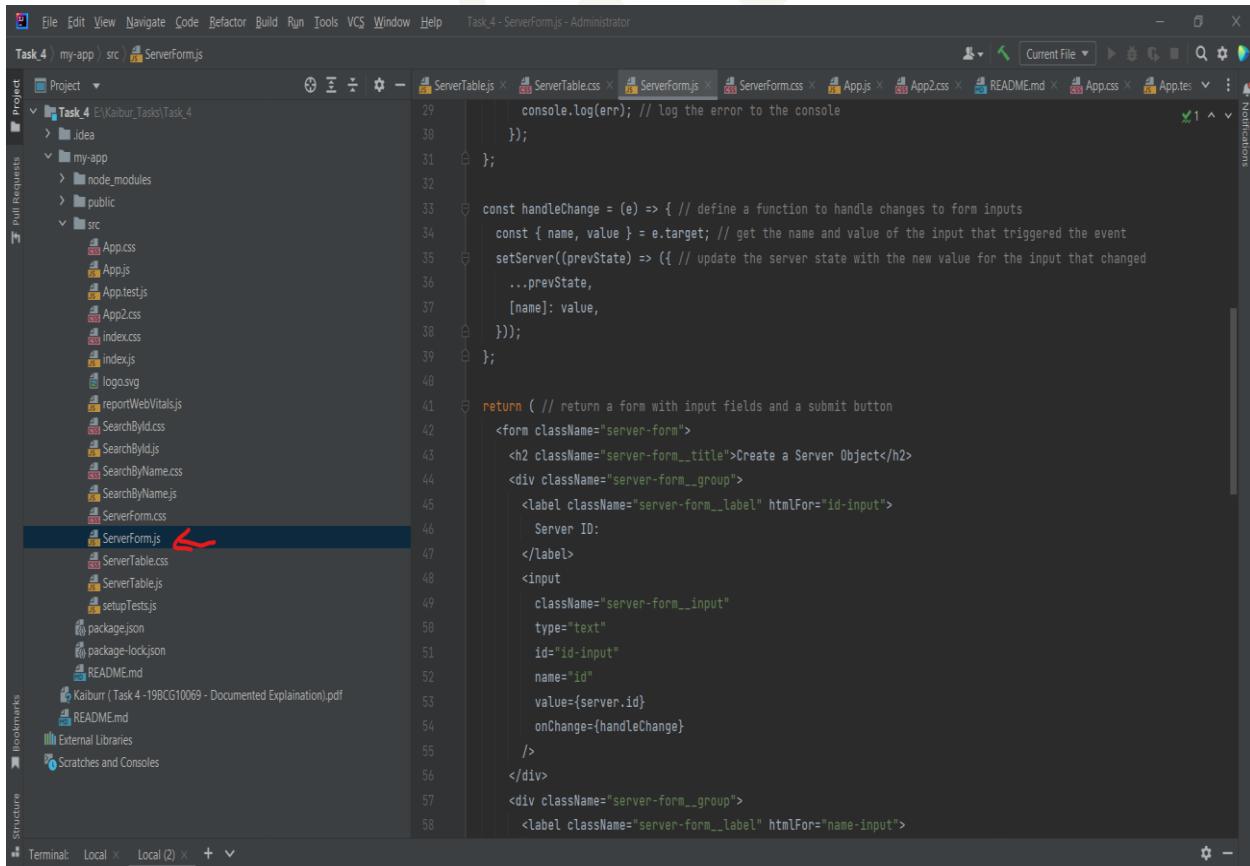


c. *ServerForm.js(logic):*

i. Create a File named ‘**ServerFrom.js**’ with the *following code*:



```
1 import React, { useState } from "react"; // import React and useState hook from react library
2 import axios from "axios"; // import axios for making API requests
3 import "./ServerForm.css"; // import CSS styles for this component
4
5 function ServerForm({ onServerCreated }) { // define ServerForm component that takes onServerCreated prop
6   const [server, setServer] = useState({ // set up state using the useState hook
7     id: "",
8     name: "",
9     language: "",
10    framework: ""
11   });
12   const [message, setMessage] = useState(""); // set up another piece of state for displaying messages
13
14   const handleSubmit = (e) => { // define a function to handle form submissions
15     e.preventDefault(); // prevent the default form submission behavior
16     axios // use axios to make a POST request to the server API
17       .post("http://localhost:8080/servers", server) // specify the URL and data to send
18       .then((res) => { // if the request is successful
19         onServerCreated(res.data); // call the onServerCreated callback with the data returned by the server
20         setMessage("Server created successfully!"); // set a success message to display to the user
21         setServer({ // reset the form inputs
22           id: "",
23           name: "",
24           language: "",
25           framework: ""
26         });
27       })
28       .catch((err) => { // if there's an error
29         console.log(err); // log the error to the console
30       });
31   };
32
33   const handleChange = (e) => { // define a function to handle changes to form inputs
34     const { name, value } = e.target; // get the name and value of the input that triggered the event
35     setServer((prevState) => ({ // update the server state with the new value for the input that changed
36       ...prevState,
37       [name]: value,
38     }));
39   };
40
41   return ( // return a form with input fields and a submit button
42     <form className="server-form">
43       <h2>Create a Server Object</h2>
44       <div className="server-form__group">
45         <label className="server-form__label" htmlFor="id-input">
46           Server ID:
47         </label>
48         <input
49           className="server-form__input"
50           type="text"
51           id="id-input"
52           name="id"
53           value={server.id}
54           onChange={handleChange}
55         />
56       </div>
57       <div className="server-form__group">
58         <label className="server-form__label" htmlFor="name-input">
```



```
29         </label>
30       </div>
31     );
32
33   const handleChange = (e) => { // define a function to handle changes to form inputs
34     const { name, value } = e.target; // get the name and value of the input that triggered the event
35     setServer((prevState) => ({ // update the server state with the new value for the input that changed
36       ...prevState,
37       [name]: value,
38     }));
39   };
40
41   return ( // return a form with input fields and a submit button
42     <form className="server-form">
43       <h2>Create a Server Object</h2>
44       <div className="server-form__group">
45         <label className="server-form__label" htmlFor="id-input">
46           Server ID:
47         </label>
48         <input
49           className="server-form__input"
50           type="text"
51           id="id-input"
52           name="id"
53           value={server.id}
54           onChange={handleChange}
55         />
56       </div>
57       <div className="server-form__group">
58         <label className="server-form__label" htmlFor="name-input">
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - ServerForm.js - Administrator

Task_4 my-app src ServerForm.js

Project Pull Requests Bookmarks Structure Terminal Local Local (2)

```
<div className="server-form__group">
  <label className="server-form__label" htmlFor="name-input">
    Server Name:
  </label>
  <input
    className="server-form__input"
    type="text"
    id="name-input"
    name="name"
    value={server.name}
    onChange={handleChange}
  />
</div>
<div className="server-form__group">
  <label className="server-form__label" htmlFor="language-input">
    Language:
  </label>
  <input
    className="server-form__input"
    type="text"
    id="language-input"
    name="language"
    value={server.language}
    onChange={handleChange}
  />
</div>
<div className="server-form__group">
  <label className="server-form__label" htmlFor="framework-input">
    Framework:
  </label>
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - ServerForm.js - Administrator

Task_4 my-app src ServerForm.js

Project Pull Requests Bookmarks Structure Terminal Local Local (2)

```
<div className="server-form__group">
  <label className="server-form__label" htmlFor="framework-input">
    Framework:
  </label>
  <input
    className="server-form__input"
    type="text"
    id="framework-input"
    name="framework"
    value={server.framework}
    onChange={handleChange}
  />
</div>
<button className="server-form__submit" type="submit" onClick={handleSubmit}>
  Create
</button>
{message && <p className="server-form__message">{message}</p>}
</form>
);

}

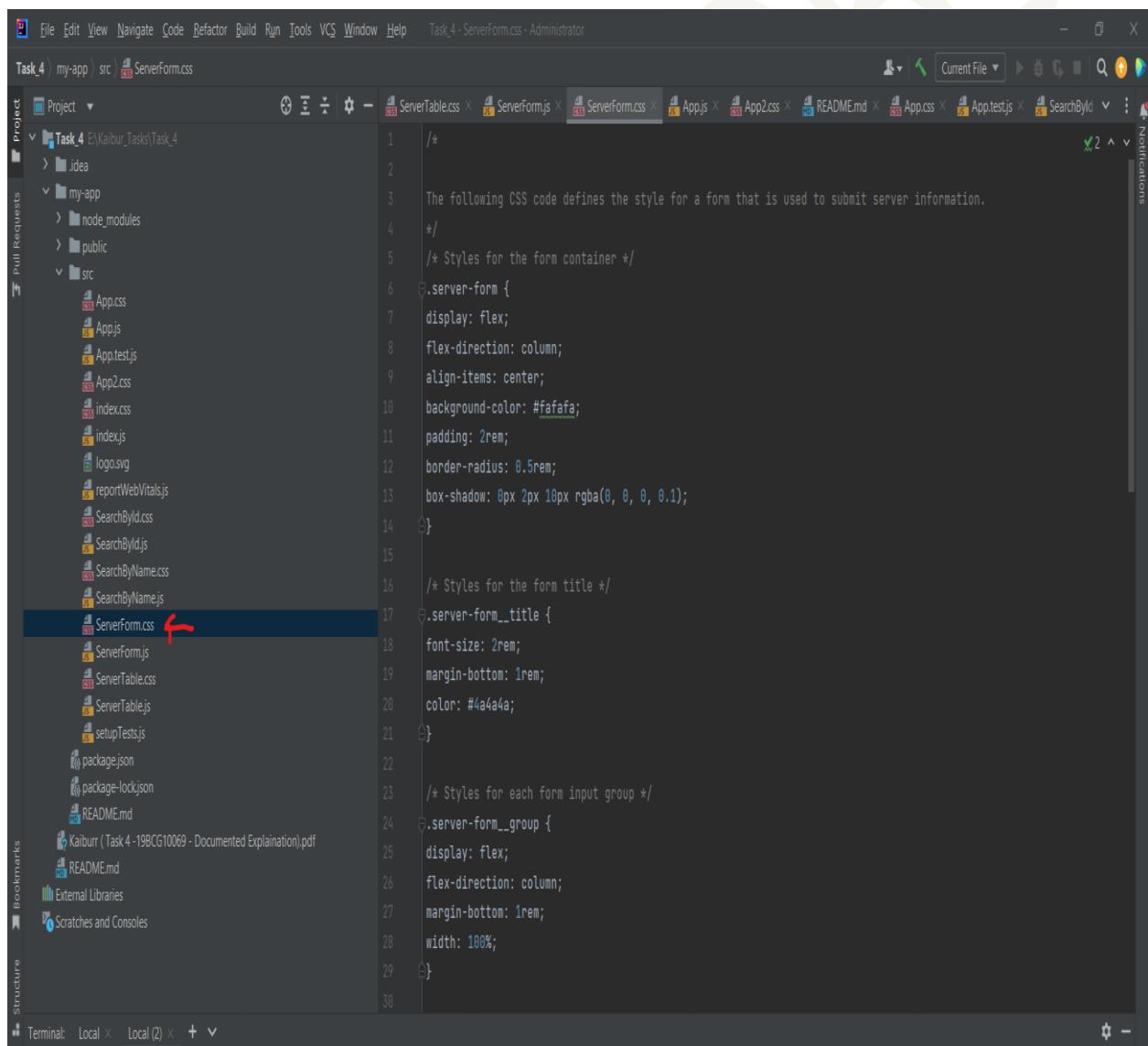
export default ServerForm;

// Code Written by @Arpit Singh 19BCG10069
```

- In summary, the component uses axios to make a ***POST request to a server endpoint, passing the server*** object as the request body. When the server responds with a success status code, the component updates its state with a ***success message*** and clears the form. The component also provides an ***on-Change event listener for each input field to update the component's state*** as the user types into the input field. Finally, the component provides a ***button to submit the form and an onServerCreated prop that gets called with the created server object when the server responds with a success status code.***

d. **ServerForm.css(style):**

- Create a File named '**ServerForm.css**' with the ***following code:***



```

1  /*
2   *
3   * The following CSS code defines the style for a form that is used to submit server information.
4   */
5  /* Styles for the form container */
6  .server-form {
7      display: flex;
8      flex-direction: column;
9      align-items: center;
10     background-color: #fafafa;
11     padding: 2rem;
12     border-radius: 0.5rem;
13     box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.1);
14 }
15 /* Styles for the form title */
16 .server-form__title {
17     font-size: 2rem;
18     margin-bottom: 1rem;
19     color: #4a4a4a;
20 }
21 /* Styles for each form input group */
22 .server-form__group {
23     display: flex;
24     flex-direction: column;
25     margin-bottom: 1rem;
26     width: 100%;
27 }
28
29
30
31
32
33
34
35
36
37
38

```

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - ServerForm.css - Administrator
Task_4 > my-app > src > ServerForm.css
Project
Task_4 E:\Kaibur_Task\Task_4
> idea
my-app
> node_modules
> public
src
  App.css
  App.js
  App-test.js
  App2.css
  index.css
  index.js
  logo.svg
  reportWebVitals.js
  SearchById.css
  SearchById.js
  SearchByName.css
  SearchByName.js
  ServerForm.css
  ServerForm.js
  ServerTable.css
  ServerTable.js
  setupTests.js
  package.json
  package-lock.json
  README.md
  External Libraries
  Scratches and Consoles
  Kalbur (Task 4 -19BCG10069 - Documented Explanation).pdf
  README.md
  External Libraries
  Scratches and Consoles
  Terminal Local Local (2) + Version Control TODO Problems Terminal Services
  Notifications
```

```
29 }
30
31 /* Styles for each form input label */
32 .server-form__label {
33   font-size: 1.2rem;
34   margin-bottom: 0.5rem;
35   color: #4a4a4a;
36 }
37
38 /* Styles for each form input */
39 .server-form__input {
40   padding: 0.5rem;
41   border: none;
42   border-radius: 0.25rem;
43   font-size: 1rem;
44   background-color: #f2f2f2;
45   color: #4a4a4a;
46 }
47
48 /* Styles for the form submit button */
49 .server-form__submit {
50   padding: 0.5rem;
51   border: none;
52   border-radius: 0.25rem;
53   font-size: 1rem;
54   background-color: #4a90e2;
55   color: #fff;
56   cursor: pointer;
57   transition: background-color 0.2s ease-in-out;
58 }
```

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - ServerForm.css - Administrator
Task_4 > my-app > src > ServerForm.css
Project
Task_4 E:\Kaibur_Task\Task_4
> idea
my-app
> node_modules
> public
src
  App.css
  App.js
  App-test.js
  App2.css
  index.css
  index.js
  logo.svg
  reportWebVitals.js
  SearchById.css
  SearchById.js
  SearchByName.css
  SearchByName.js
  ServerForm.css
  ServerForm.js
  ServerTable.css
  ServerTable.js
  setupTests.js
  package.json
  package-lock.json
  README.md
  External Libraries
  Scratches and Consoles
  Kalbur (Task 4 -19BCG10069 - Documented Explanation).pdf
  README.md
  External Libraries
  Scratches and Consoles
  Terminal Local Local (2) + Version Control TODO Problems Terminal Services
  Notifications
```

```
47
48 /* Styles for the form submit button */
49 .server-form__submit {
50   padding: 0.5rem;
51   border: none;
52   border-radius: 0.25rem;
53   font-size: 1rem;
54   background-color: #4a90e2;
55   color: #fff;
56   cursor: pointer;
57   transition: background-color 0.2s ease-in-out;
58 }

59
60 /* Styles for the form submit button when hovered */
61 .server-form__submit:hover {
62   background-color: #3d78c2;
63 }

64
65 /* Styles for the form message container */
66 .server-form__message {
67   font-size: 1.2rem;
68   margin-top: 1rem;
69   color: #4a4a4a;
70 }

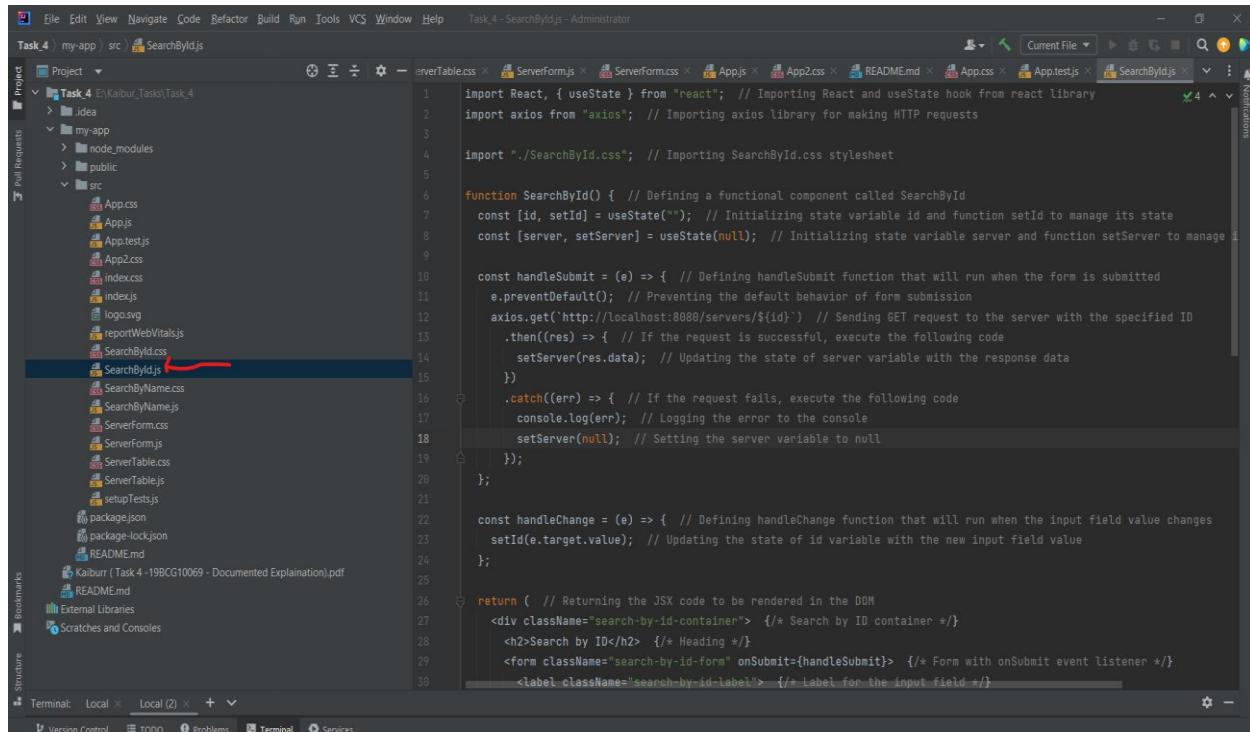
71 /* Code Written by @Argit Singh 19BCG10069 */
```

- In summary, This CSS file contains styles for a form with a ***title, input fields, and a submit button***. The styles define the layout and appearance of the form, including its background color, padding, border radius, box shadow, and font styles. The ***input fields and submit button have specific styles*** for their padding, border, border radius, font size, background color, and col-

or. The **submit button also has a hover effect** to change its background color. Finally, there is a **message style** for displaying messages related to the form.

e. **SearchById.js(logic):**

i. Create a File named '**SearchById.js**' with the **following code**:



```

import React, { useState } from "react"; // Importing React and useState hook from react library
import axios from "axios"; // Importing axios library for making HTTP requests

import "./SearchById.css"; // Importing SearchById.css stylesheet

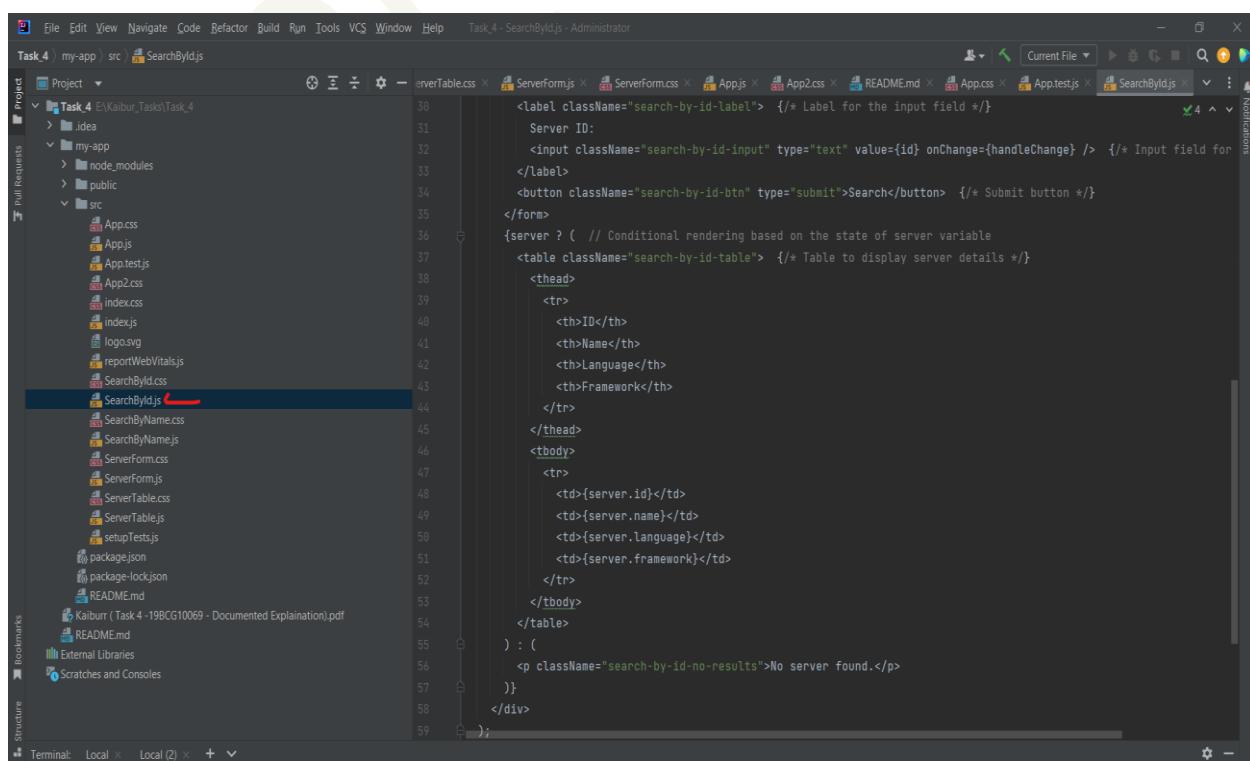
function SearchById() { // Defining a functional component called SearchById
  const [id, setId] = useState(""); // Initializing state variable id and function setId to manage its state
  const [server, setServer] = useState(null); // Initializing state variable server and function setServer to manage its state

  const handleSubmit = (e) => { // Defining handleSubmit function that will run when the form is submitted
    e.preventDefault(); // Preventing the default behavior of form submission
    axios.get(`http://localhost:8080/servers/${id}`) // Sending GET request to the server with the specified ID
      .then((res) => { // If the request is successful, execute the following code
        setServer(res.data); // Updating the state of server variable with the response data
      })
      .catch((err) => { // If the request fails, execute the following code
        console.log(err); // Logging the error to the console
        setServer(null); // Setting the server variable to null
      });
  };

  const handleChange = (e) => { // Defining handleChange function that will run when the input field value changes
    setId(e.target.value); // Updating the state of id variable with the new input field value
  };

  return ( // Returning the JSX code to be rendered in the DOM
    <div className="search-by-id-container"> {/* Search by ID container */}
      <h2>Search by ID</h2> {/* Heading */}
      <form className="search-by-id-form" onSubmit={handleSubmit}> {/* Form with onSubmit event listener */}
        <label className="search-by-id-label"> {/* Label for the input field */}
          <input className="search-by-id-input" type="text" value={id} onChange={handleChange} /> {/* Input field for server ID */}
        </label>
        <button className="search-by-id-btn" type="submit">Search</button> {/* Submit button */}
      </form>
      {server ? ( // Conditional rendering based on the state of server variable
        <table className="search-by-id-table"> {/* Table to display server details */}
          <thead>
            <tr>
              <th>ID</th>
              <th>Name</th>
              <th>Language</th>
              <th>Framework</th>
            </tr>
          </thead>
          <tbody>
            <tr>
              <td>{server.id}</td>
              <td>{server.name}</td>
              <td>{server.language}</td>
              <td>{server.framework}</td>
            </tr>
          </tbody>
        </table>
      ) : (
        <p className="search-by-id-no-results">No server found.</p>
      )}
    </div>
  );
}

```



```

30   <label className="search-by-id-label"> {/* Label for the input field */}
31     Server ID:
32     <input className="search-by-id-input" type="text" value={id} onChange={handleChange} /> {/* Input field for
33     </label>
34     <button className="search-by-id-btn" type="submit">Search</button> {/* Submit button */}
35   </form>
36   {server ? ( // Conditional rendering based on the state of server variable
37     <table className="search-by-id-table"> {/* Table to display server details */}
38       <thead>
39         <tr>
40           <th>ID</th>
41           <th>Name</th>
42           <th>Language</th>
43           <th>Framework</th>
44         </tr>
45       </thead>
46       <tbody>
47         <tr>
48           <td>{server.id}</td>
49           <td>{server.name}</td>
50           <td>{server.language}</td>
51           <td>{server.framework}</td>
52         </tr>
53       </tbody>
54     </table>
55   ) : (
56     <p className="search-by-id-no-results">No server found.</p>
57   )
58 }

```

```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - SearchById.js - Administrator
Task_4 > my-app > src > SearchById.js
Project > Task_4 E:\Kaibur_Tasks\Task_4
my-app
src
  App.css
  App.js
  App.test.js
  App2.css
  index.css
  index.js
  logo.svg
  reportWebVitals.js
  SearchByName.css
  SearchById.css
  SearchByName.js
  ServerForm.css
  ServerForm.js
  ServerTable.css
  ServerTable.js
  setupTests.js
  package.json
  package-lock.json
  README.md
  README.md
  External Libraries
  Scratches and Consoles

Terminal Local Local (2) + Version Control TODO Problems Terminal Services

41           <th>Name</th>
42           <th>Language</th>
43           <th>Framework</th>
44       </tr>
45   </thead>
46   <tbody>
47     <tr>
48       <td>{server.id}</td>
49       <td>{server.name}</td>
50       <td>{server.language}</td>
51       <td>{server.framework}</td>
52     </tr>
53   </tbody>
54 >   </table>
55   >   <p className="search-by-id-no-results">No server found.</p>
56   > }
57   > );
58 }
59
60
61
62 export default SearchById; // Exporting the SearchById component for use in other parts of the application
63
64

```

- In summary, this code defines a functional component ***SearchById*** that allows the user to ***search for a server in a database by its ID***. It renders a form with an input field for the user to ***enter the server ID and a submit button***. When the user submits the form, an HTTP ***GET request is sent to an API endpoint*** with the ID parameter. If the request is ***successful, the server data is returned and displayed in a table***. If the request fails, an error is logged to the console and a message is displayed stating that ***no server was found***.

f. **SearchById.css(style)**:

- Create a File named '***SearchById.css***' with the ***following code***:

```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - SearchById.css - Administrator
Task_4 > my-app > src > SearchById.css
Project > Task_4 E:\Kaibur_Tasks\Task_4
my-app
src
  App.css
  App.js
  App.test.js
  App2.css
  index.css
  index.js
  logo.svg
  reportWebVitals.js
  SearchByName.css
  SearchById.css
  SearchByName.js
  ServerForm.css
  ServerForm.js
  ServerTable.css
  ServerTable.js
  setupTests.js
  package.json
  package-lock.json
  README.md
  README.md
  External Libraries
  Scratches and Consoles

Terminal Local Local (2) + Version Control TODO Problems Terminal Services

1 /* The following CSS defines the styles for the SearchById component */
2
3 /* Styles for the container that holds the search form and search results */
4 .search-by-id-container {
5   display: flex;
6   flex-direction: column;
7   align-items: center;
8   margin-top: 20px;
9 }
10
11 /* Styles for the search form */
12 .search-by-id-form {
13   display: flex;
14   flex-direction: row;
15   align-items: center;
16   margin-bottom: 20px;
17 }
18
19 /* Styles for the label that accompanies the search input */
20 .search-by-id-label {
21   margin-right: 10px;
22 }
23
24 /* Styles for the search input */
25 .search-by-id-input {
26   padding: 5px;
27   border-radius: 5px;
28   border: 1px solid #ccc;
29   margin-right: 10px;
30 }


```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - SearchById.css - Administrator

Task_4 > my-app > src > SearchById.css

Project Pull Requests Bookmarks Structure

Terminal Local Local (2) + Version Control TODO Problems Terminal Services

```
border: 1px solid #ccc;
margin-right: 10px;

/* Styles for the search button */

.search-by-id-btn {
  padding: 5px 10px;
  border-radius: 5px;
  background-color: #4CAF50;
  color: white;
  border: none;
  cursor: pointer;
}

/* Styles for the search results table */

.search-by-id-table {
  border-collapse: collapse;
  width: 100%;
  max-width: 800px;
  margin-bottom: 20px;
}

/* Styles for the table headers and data cells */

.search-by-id-table th, .search-by-id-table td {
  text-align: left;
  padding: 8px;
}

/* Styles for the table headers */

.search-by-id-table th {
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - SearchById.css - Administrator

Task_4 > my-app > src > SearchById.css

Project Pull Requests Bookmarks Structure

Terminal Local Local (2) + Version Control TODO Problems Terminal Services

```
.search-by-id-table th {
  background-color: #4CAF50;
  color: white;
}

/* Styles for even rows in the table */

.search-by-id-table tr:nth-child(even) {
  background-color: #f2f2f2;
}

/* Styles for table rows on hover */

.search-by-id-table tr:hover {
  background-color: #ddd;
}

/* Styles for the "No server found" message */

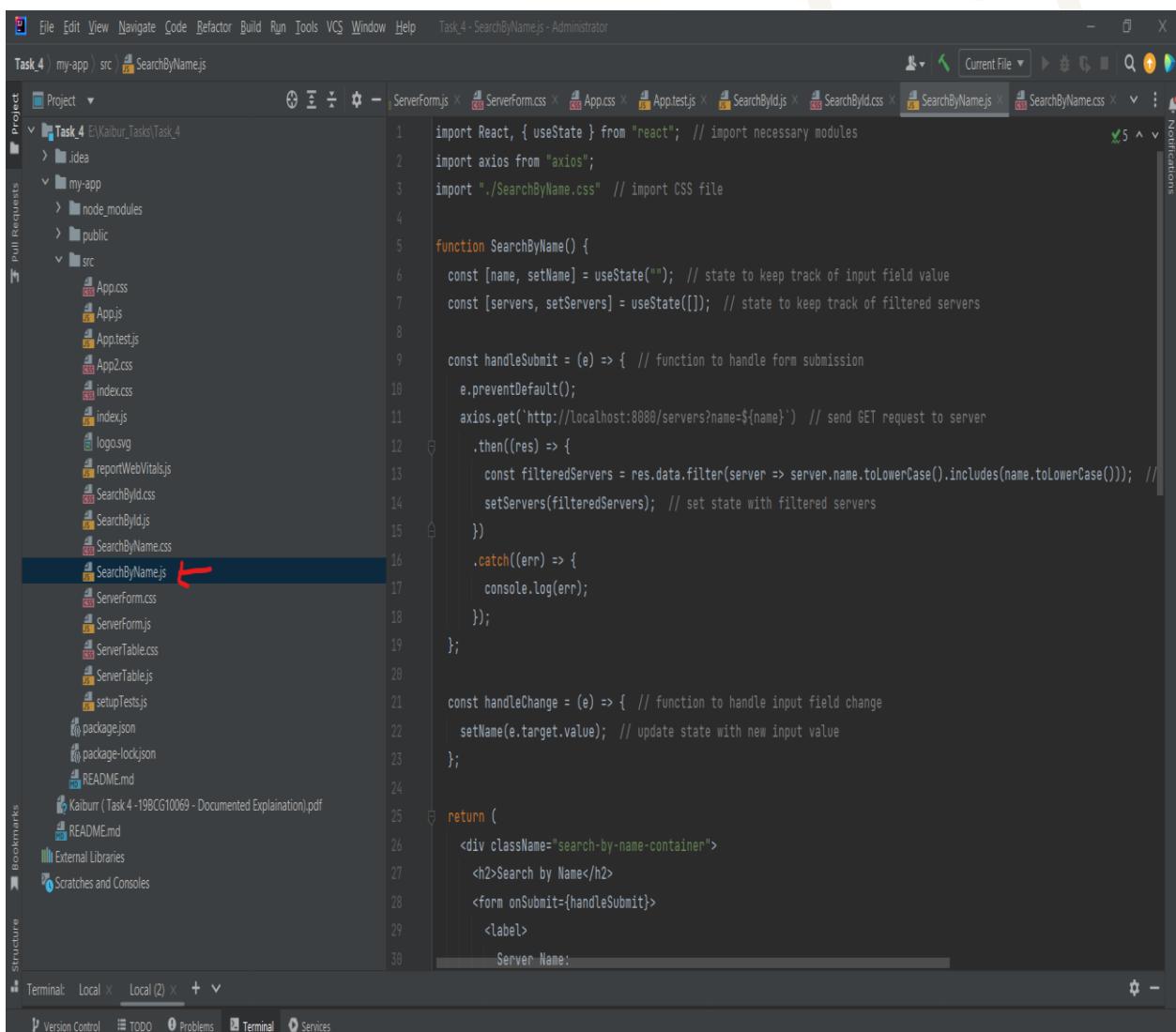
.search-by-id-no-results {
  color: red;
  font-style: italic;
}

/* In summary, this CSS file defines the styles for the SearchById component, including the search form, search results
```

- In summary, This CSS file contains styling rules for the components in the **SearchById React component**. It styles the **container, form, input fields, button, table, and message** that are used to search for and display server data. The styling includes **flexbox layout, borders, colors, font styles, and responsive design** for different screen sizes.

g. **SearchByName.js(logic)**:

- Create a File named '**SearchByName.js**' with the **following code**:

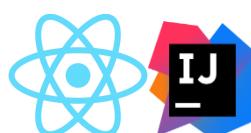


The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "Task_4". The "src" directory contains files like App.css, App.js, App.test.js, App2.css, index.css, index.js, logo.svg, reportWebVitals.js, SearchById.css, SearchById.js, SearchByName.css, and SearchByName.js. A red arrow points to the "SearchByName.js" file in the list.
- Code Editor:** Displays the content of the "SearchByName.js" file. The code uses React hooks (useState) and axios to handle a search by name. It includes state management for the input value and servers, a handleSubmit function to send a GET request, and a handleChange function to update the input state.
- Toolbars and Status Bar:** Standard IntelliJ IDEA toolbars and status bar at the bottom.

```

1 import React, { useState } from "react"; // import necessary modules
2 import axios from "axios";
3 import "./SearchByName.css" // import CSS file
4
5 function SearchByName() {
6   const [name, setName] = useState(""); // state to keep track of input field value
7   const [servers, setServers] = useState([]); // state to keep track of filtered servers
8
9   const handleSubmit = (e) => { // function to handle form submission
10     e.preventDefault();
11     axios.get(`http://localhost:8080/servers?name=${name}`) // send GET request to server
12       .then((res) => {
13         const filteredServers = res.data.filter(server => server.name.toLowerCase().includes(name.toLowerCase())); //
14         setServers(filteredServers); // set state with filtered servers
15       })
16       .catch((err) => {
17         console.log(err);
18       });
19   };
20
21   const handleChange = (e) => { // function to handle input field change
22     setName(e.target.value); // update state with new input value
23   };
24
25   return (
26     <div className="search-by-name-container">
27       <h2>Search by Name</h2>
28       <form onSubmit={handleSubmit}>
29         <label>
30           Server Name:
31         </label>
32         <input type="text" value={name} onChange={handleChange} />
33       </form>
34     </div>
35   );
36 }
37
38 export default SearchByName;
  
```



```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - SearchByName.js - Administrator
Task_4 > my-app > src > SearchByName.js
Project Current File
Task_4 E:\Kalbur_Task\Task_4
my-app
node_modules
public
src
App.css
App.js
App.test.js
App2.css
index.css
index.js
logo.svg
reportWebVitals.js
SearchByld.css
SearchByld.js
SearchByName.css
SearchByName.js
ServerForm.css
ServerForm.js
ServerTable.css
ServerTable.js
setupTests.js
package.json
package-lock.json
README.md
Kalbur (Task 4 -19BCG10069 - Documented Explanation).pdf
External Libraries
Scratches and Consoles
Structure
Terminal Local Local (2) + Version Control TODO Problems Terminal Services

```

```

30     Server Name:
31     <input type="text" value={name} onChange={handleChange} className="search-input" /> // input field for server
32   </label>
33   <button type="submit" className="search-button">Search</button> // submit button
34 </form>
35   &lt;div>
36     &lt;table className="server-table"&gt;
37       &lt;thead&gt;
38         &lt;tr&gt;
39           &lt;th>ID</th>
40           &lt;th>Name</th>
41           &lt;th>Language</th>
42           &lt;th>Framework</th>
43         &lt;/tr&gt;
44       &lt;/thead&gt;
45       &lt;tbody&gt;
46         {servers.map((server) => (
47           &lt;tr key={server.id}&gt;
48             &lt;td&gt;{server.id}&lt;/td&gt;
49             &lt;td&gt;{server.name}&lt;/td&gt;
50             &lt;td&gt;{server.language}&lt;/td&gt;
51             &lt;td&gt;{server.framework}&lt;/td&gt;
52           &lt;/tr&gt;
53         )));
54       &lt;/tbody&gt;
55     &lt;/table&gt;
56   ) : ( // if no servers found, display message
57     <p className="no-server">No servers found.</p>
58   )
59 &lt;/div>

```

```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - SearchByName.js - Administrator
Task_4 > my-app > src > SearchByName.js
Project Current File
Task_4 E:\Kalbur_Task\Task_4
my-app
node_modules
public
src
App.css
App.js
App.test.js
App2.css
index.css
index.js
logo.svg
reportWebVitals.js
SearchByld.css
SearchByld.js
SearchByName.css
SearchByName.js
ServerForm.css
ServerForm.js
ServerTable.css
ServerTable.js
setupTests.js
package.json
package-lock.json
README.md
Kalbur (Task 4 -19BCG10069 - Documented Explanation).pdf
External Libraries
Scratches and Consoles
Structure
Terminal Local Local (2) + Version Control TODO Problems Terminal Services

```

```

42     <th>Framework</th>
43   </tr>
44 </thead>
45 <tbody>
46   {servers.map((server) => (
47     <tr key={server.id}>
48       <td>{server.id}</td>
49       <td>{server.name}</td>
50       <td>{server.language}</td>
51       <td>{server.framework}</td>
52     </tr>
53   )));
54 </tbody>
55 </table>
56 ) : ( // if no servers found, display message
57   <p className="no-server">No servers found.</p>
58 )
59 </div>
60
61
62 export default SearchByName;
63
64 /* Code Written by @Arpit Singh 19BCG10069 */

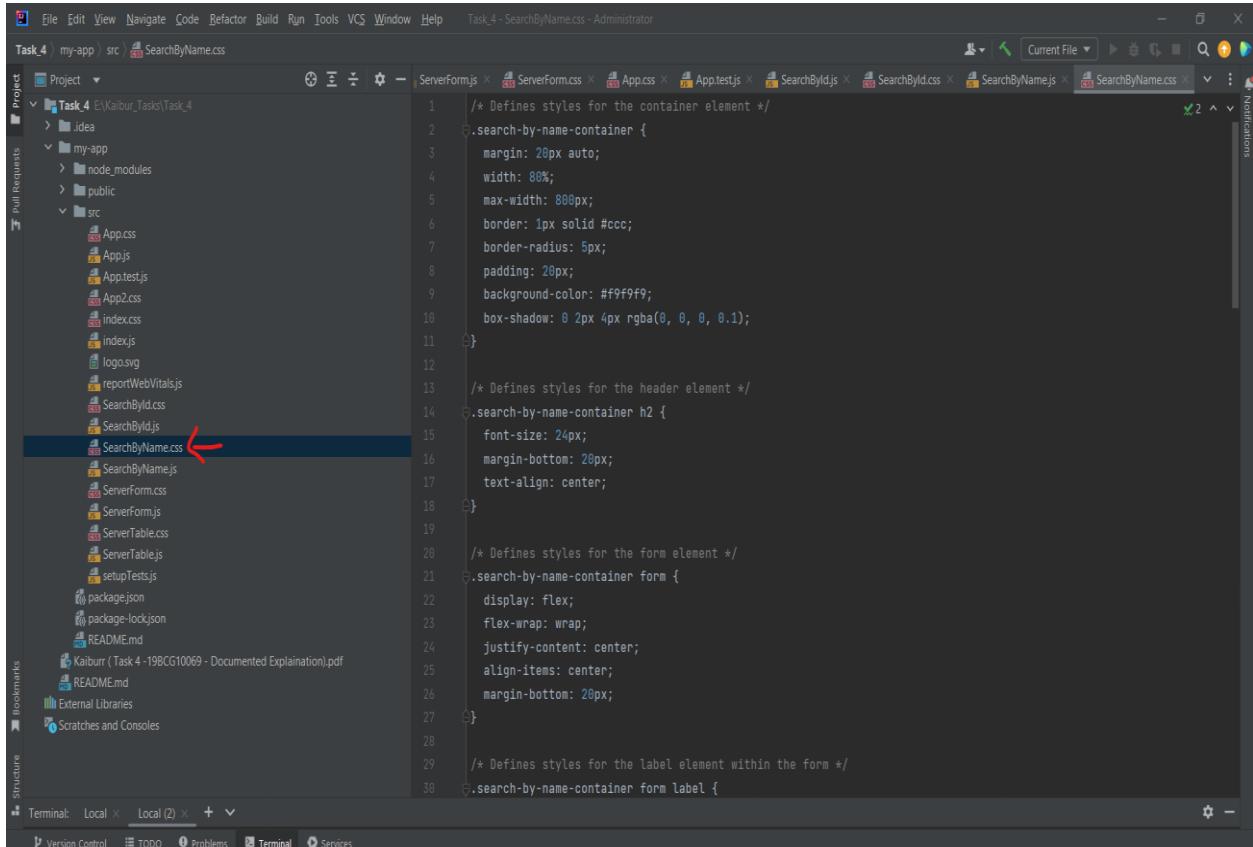
```

- In summary, this code creates a **React component for searching servers by name**. It includes a form with an input field and submit button for entering and submitting the server name to search for. Upon form submission, a **GET request is sent to a server to retrieve a list of servers**. The retrieved servers are then **filtered based on the entered server name**, and the filtered servers are displayed in a table if any are found. If **no servers are found, a message is displayed instead**.



h. SearchByName.css(style):

i. Create a File named ‘SearchByName.css’ with the *following code*:



```
/* Defines styles for the container element */
.search-by-name-container {
    margin: 20px auto;
    width: 80%;
    max-width: 800px;
    border: 1px solid #ccc;
    border-radius: 5px;
    padding: 20px;
    background-color: #f9f9f9;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

/* Defines styles for the header element */
.search-by-name-container h2 {
    font-size: 24px;
    margin-bottom: 20px;
    text-align: center;
}

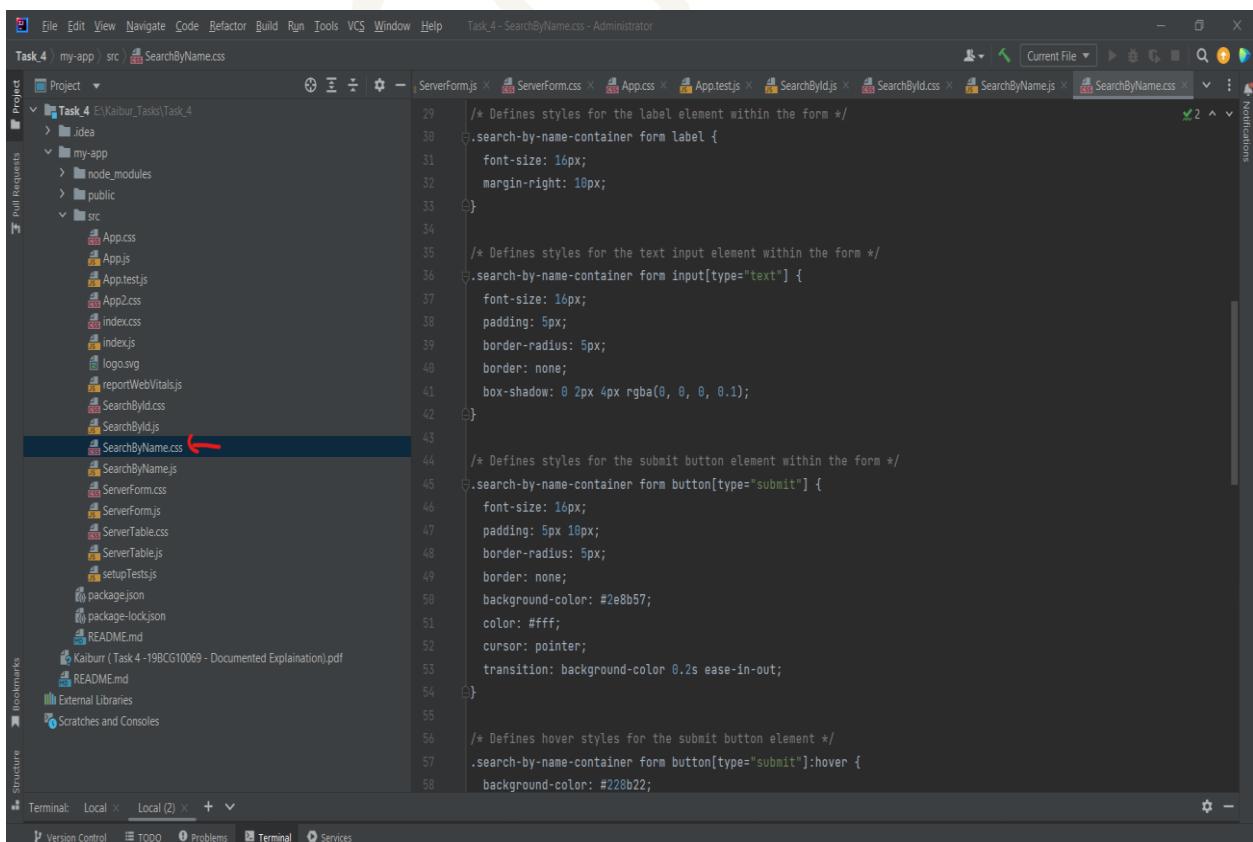
/* Defines styles for the form element */
.search-by-name-container form {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    align-items: center;
    margin-bottom: 20px;
}

/* Defines styles for the label element within the form */
.search-by-name-container form label {
    font-size: 16px;
    margin-right: 10px;
}

/* Defines styles for the text input element within the form */
.search-by-name-container form input[type="text"] {
    font-size: 16px;
    padding: 5px;
    border-radius: 5px;
    border: none;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

/* Defines styles for the submit button element within the form */
.search-by-name-container form button[type="submit"] {
    font-size: 16px;
    padding: 5px 10px;
    border-radius: 5px;
    border: none;
    background-color: #2e8b51;
    color: #fff;
    cursor: pointer;
    transition: background-color 0.2s ease-in-out;
}

/* Defines hover styles for the submit button element */
.search-by-name-container form button[type="submit"]:hover {
    background-color: #228b22;
}
```

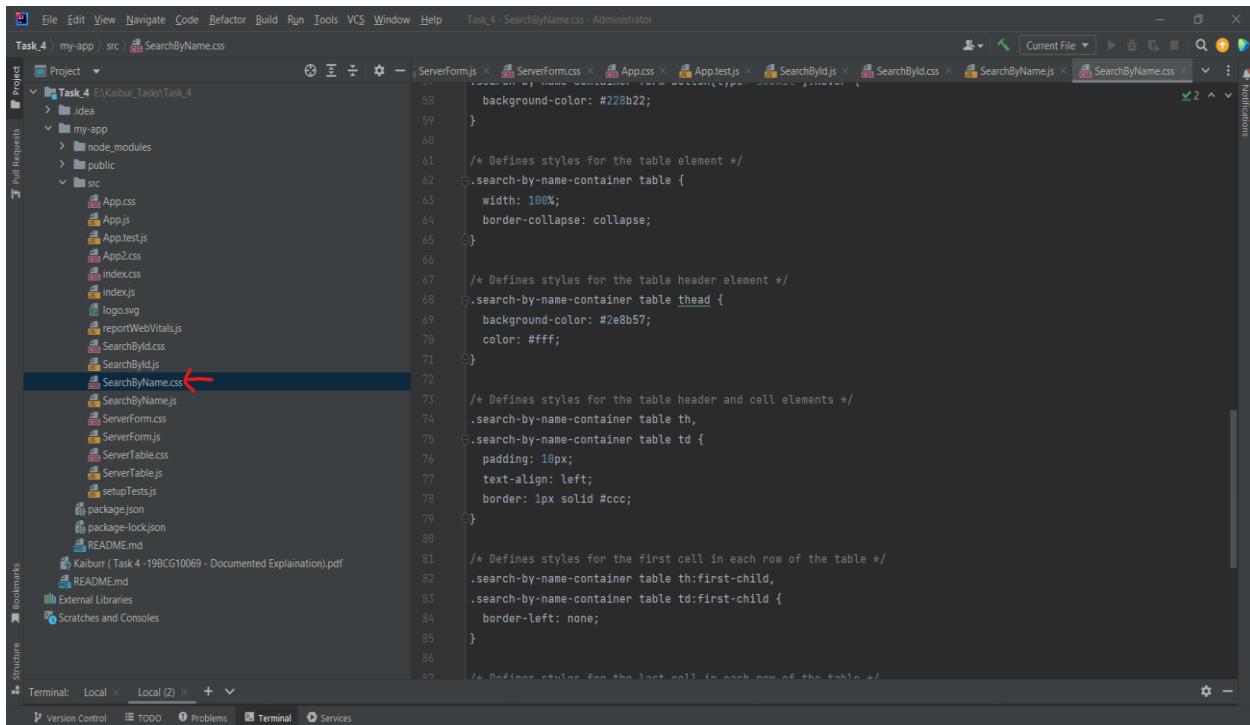


```
/* Defines styles for the label element within the form */
.search-by-name-container form label {
    font-size: 16px;
    margin-right: 10px;
}

/* Defines styles for the text input element within the form */
.search-by-name-container form input[type="text"] {
    font-size: 16px;
    padding: 5px;
    border-radius: 5px;
    border: none;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

/* Defines styles for the submit button element within the form */
.search-by-name-container form button[type="submit"] {
    font-size: 16px;
    padding: 5px 10px;
    border-radius: 5px;
    border: none;
    background-color: #2e8b51;
    color: #fff;
    cursor: pointer;
    transition: background-color 0.2s ease-in-out;
}

/* Defines hover styles for the submit button element */
.search-by-name-container form button[type="submit"]:hover {
    background-color: #228b22;
}
```

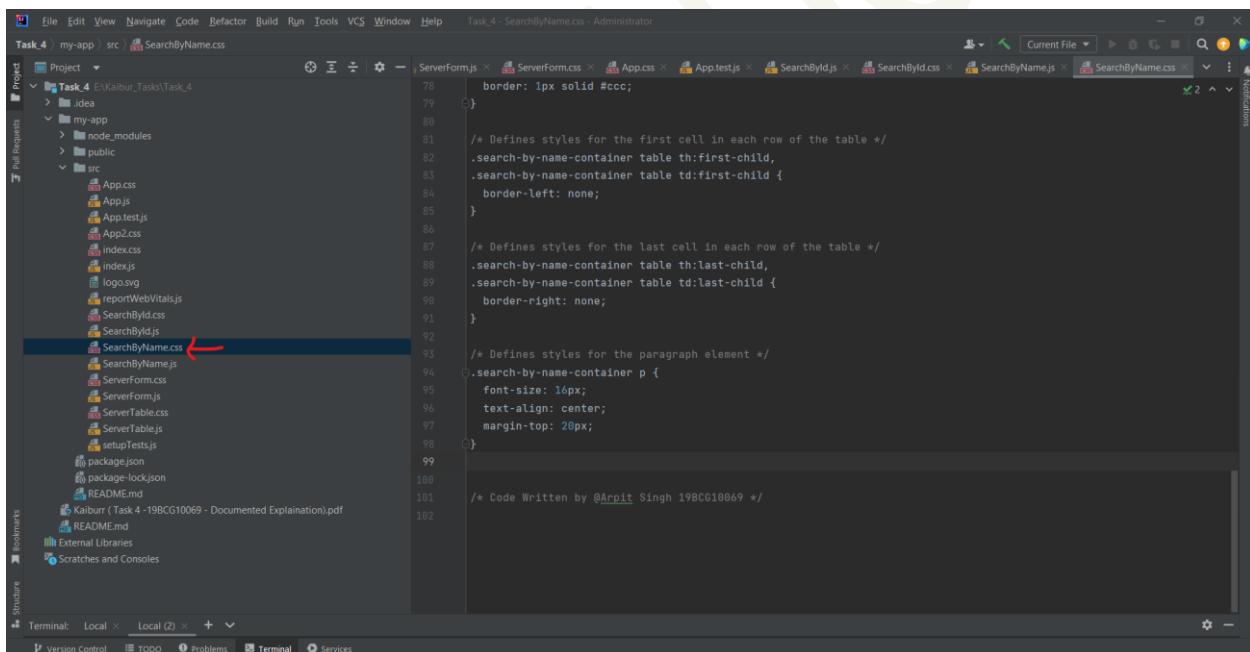


```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - SearchByName.css - Administrator

Project Task_4 my-app / src / SearchByName.css
  -> idea
    -> my-app
      -> node_modules
      -> public
      -> src
        -> App.css
        -> App.js
        -> App.test.js
        -> App2.css
        -> index.css
        -> index.js
        -> logo.svg
        -> reportWebVitals.js
        -> SearchByld.css
        -> SearchByld.js
        -> SearchByName.css
        -> SearchByName.js
        -> ServerForm.css
        -> ServerForm.js
        -> ServerTable.css
        -> ServerTable.js
        -> setupTests.js
        package.json
        package-lock.json
        README.md
        README.md
        External Libraries
        Scratches and Consoles

SearchByName.css
  .search-by-name-container {
    background-color: #228b22;
  }
  .search-by-name-container table {
    width: 100%;
    border-collapse: collapse;
  }
  .search-by-name-container table thead {
    background-color: #2e8b57;
    color: #fff;
  }
  .search-by-name-container table th,
  .search-by-name-container table td {
    padding: 10px;
    text-align: left;
    border: 1px solid #ccc;
  }
  .search-by-name-container table th:first-child,
  .search-by-name-container table td:first-child {
    border-left: none;
  }
  .search-by-name-container table th:last-child,
  .search-by-name-container table td:last-child {
    border-right: none;
  }
  .search-by-name-container p {
    font-size: 16px;
    text-align: center;
    margin-top: 20px;
  }
  /* Code Written by @Arpit Singh 19BCG10069 */

Terminal: Local < Local (2) < + Version Control TODO Problems Terminal Services
```



```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - SearchByName.css - Administrator

Project Task_4 my-app / src / SearchByName.css
  -> idea
    -> my-app
      -> node_modules
      -> public
      -> src
        -> App.css
        -> App.js
        -> App.test.js
        -> App2.css
        -> index.css
        -> index.js
        -> logo.svg
        -> reportWebVitals.js
        -> SearchByld.css
        -> SearchByld.js
        -> SearchByName.css
        -> SearchByName.js
        -> ServerForm.css
        -> ServerForm.js
        -> ServerTable.css
        -> ServerTable.js
        -> setupTests.js
        package.json
        package-lock.json
        README.md
        README.md
        External Libraries
        Scratches and Consoles

SearchByName.css
  .search-by-name-container p {
    font-size: 16px;
    text-align: center;
    margin-top: 20px;
  }
  /* Defines styles for the first cell in each row of the table */
  .search-by-name-container table th:first-child,
  .search-by-name-container table td:first-child {
    border-left: none;
  }
  /* Defines styles for the last cell in each row of the table */
  .search-by-name-container table th:last-child,
  .search-by-name-container table td:last-child {
    border-right: none;
  }
  /* Defines styles for the paragraph element */
  .search-by-name-container p {
    font-size: 16px;
    text-align: center;
    margin-top: 20px;
  }
  /* Code Written by @Arpit Singh 19BCG10069 */

Terminal: Local < Local (2) < + Version Control TODO Problems Terminal Services
```

- In summary, this is a CSS file that styles the **SearchByName component**. It defines the **appearance of the search form, the table that displays the search results, and the message that appears when no search results are found**. The styles include margins, padding, borders, font sizes, colors, and transitions. The CSS uses classes to target specific elements in the component, such as the container, the form, the input field, the submit button, the table, and the table headers and cells. The CSS also includes hover effects for the sub-

mit button and removes the left and right borders of the first and last table cells, respectively.

- i. **App.js**(logic – *entry point to our Application*):

i. Create a File named '**App.js**' with the ***following code***:

The screenshot shows the VS Code interface with the following details:

- Project Explorer:** Shows the project structure under "Task_4". The "src" folder contains files like App.css, App.js (highlighted with a red arrow), App.test.js, App2.css, index.css, index.js, logo.svg, reportWebVitals.js, SearchById.js, SearchByName.js, SearchByName.css, SearchById.css, ServerForm.css, ServerForm.js, ServerTable.css, ServerTable.js, and setupTests.js.
- Editor:** Displays the content of the "App.js" file. The code uses React hooks like useState and useEffect to manage server data and handle search requests.
- Terminal:** Shows the command "Local (2)".
- Bottom Bar:** Includes tabs for Version Control, TODO, Problems, Terminal, and Services.

```
1 import React, { useState, useEffect } from "react";
2 import axios from "axios";
3 import ServerForm from "./ServerForm";
4 import ServerTable from "./ServerTable";
5 import SearchById from "./SearchById";
6 import SearchByName from "./SearchByName";
7 import "./App2.css"
8
9 function App() {
10   const [servers, setServers] = useState([]);
11   const [message, setMessage] = useState("");
12
13   useEffect(() => {
14     // GET request to retrieve all servers
15     axios.get("http://localhost:8080/servers")
16       .then(res) => {
17         setServers(res.data); // Update state variable with server data
18       }
19       .catch((err) => {
20         console.log(err); // Log any errors to the console
21       });
22   }, []);
23
24   const handleSearchById = (id) => {
25     // GET request to search for a server by ID
26     axios.get(`http://localhost:8080/servers/${id}`)
27       .then(res) => {
28         setServers(res.data); // Update state variable with server data
29       }
30       .catch((err) => {
```

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - App.js - Administrator

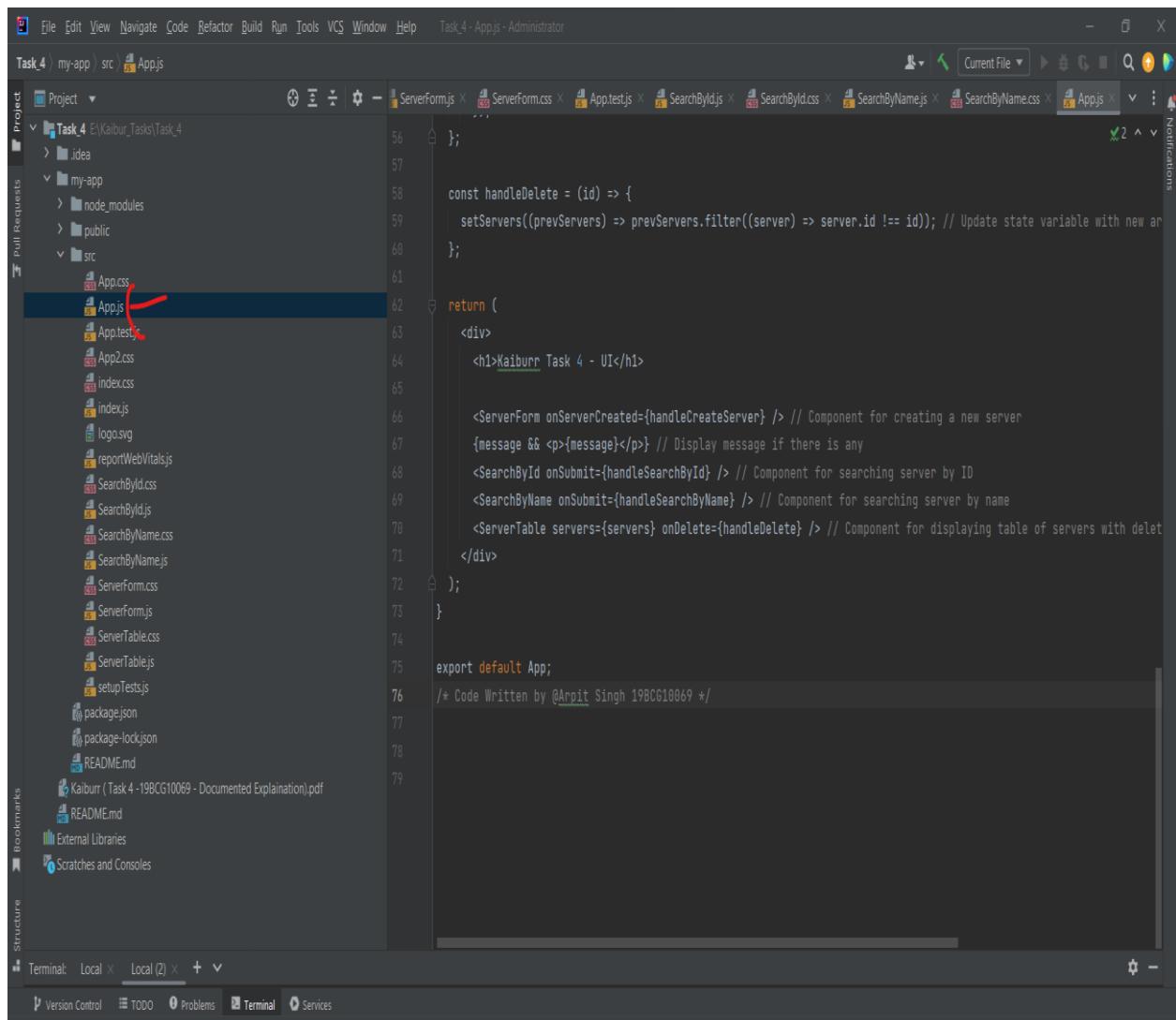
Task_4 my-app src App.js

Project ▾ Task 4 E:\KaiBurj_Tasks\Task_4
> idea
my-app
> node_modules
public
src
  App.css
  App.js
  App.test.js
  App2.css
  index.css
  index.js
  logo.svg
  reportWebVitals.js
  SearchByld.css
  SearchByld.js
  SearchByName.css
  SearchByName.js
  ServerForm.css
  ServerForm.js
  ServerTable.css
  ServerTable.js
  setupTests.js
  package.json
  package-lock.json
  README.md
  KaiBurj (Task 4 -19BCG10069 - Documented Explanation).pdf
  README.md
  External Libraries
  Scratches and Consoles

ServerForm.js x ServerForm.css x App.test.js x SearchByld.js x SearchByld.css x SearchByName.js x SearchByName.css x App.js x

  2 ^ Notifications

  29
  30      .catch((err) => {
  31          console.log(err); // Log any errors to the console
  32      });
  33  };
  34
  35  const handleSearchByName = (name) => {
  36      // GET request to search for a server by name
  37      axios.get(`http://localhost:8080/servers?name=${name}`)
  38          .then((res) => {
  39              setServers(res.data); // Update state variable with server data
  40          })
  41          .catch((err) => {
  42              console.log(err); // Log any errors to the console
  43          });
  44      };
  45
  46  const handleCreateServer = (server) => {
  47      // POST request to create a new server
  48      axios.post(`http://localhost:8080/servers`, server)
  49          .then((res) => {
  50              setServers([...servers, res.data]); // Add new server to the state variable
  51              setMessage("Server created successfully!"); // Update message state variable
  52          })
  53          .catch((err) => {
  54              console.log(err); // Log any errors to the console
  55          });
  56      };
  57
  58  const handleDelete = (id) => {
```



```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Task_4 - App.js - Administrator
Task_4 my-app src App.js
Project Task_4 E:\Kaiburr_Tasks\Task_4
my-app node_modules public src
App.css App.js App.test.js
App2.css index.css index.js logo.svg reportWebVitals.js
SearchById.css SearchById.js SearchByName.css SearchByName.js
ServerForm.css ServerForm.js ServerTable.css ServerTable.js
setupTests.js package.json package-lock.json README.md
Kaiburr (Task 4-198CG10069 - Documented Explanation).pdf README.md
External Libraries Scratches and Consoles
Current File ▾
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
};

const handleDelete = (id) => {
  setServers((prevServers) => prevServers.filter((server) => server.id !== id)); // Update state variable with new array
};

return (
  <div>
    <h1>Kaiburr Task 4 - UI</h1>

    <ServerForm onServerCreated={handleCreateServer} /> // Component for creating a new server
    {message && <p>{message}</p>} // Display message if there is any
    <SearchById onSubmit={handleSearchById} /> // Component for searching server by ID
    <SearchByName onSubmit={handleSearchByName} /> // Component for searching server by name
    <ServerTable servers={servers} onDelete={handleDelete} /> // Component for displaying table of servers with delete button
  </div>
);
}

export default App;
/* Code Written by @Arpit Singh 198CG10069 */
```

- In summary, this code defines a *React functional component called “**App**”*. The component sets up a **UI for managing servers, including creating new servers, searching for servers by ID or name, and deleting servers**. The component uses **hooks to manage state, including an array of servers, a message** to display when a server is successfully created, and functions to handle server operations.
*The component makes HTTP requests to a REST API running on **http://localhost:8080/servers** using the Axios library* to retrieve, create, and delete servers. The component renders a header, a form for creating new servers, search forms for searching for servers by ID or name, a table for displaying the list of servers, and a message displaying the success of the server creation operation.

j. **App2.css(style – App.css already exists by default):**

i. Create a File named ‘**App2.css**’ with the **following code**:

The screenshot shows a code editor interface with a dark theme. The left sidebar displays a project structure for 'Task_4' containing 'my-app' and 'src' folders. Inside 'src', there are files like 'App.css', 'App.js', 'App.test.js', and 'App2.css'. The 'App2.css' file is selected and highlighted with a red arrow. The main pane shows the CSS code for 'App2.css':

```
1 /* Defines the font family for the entire webpage */
2 body {
3     font-family: 'Roboto', sans-serif;
4 }
5
6 /* Defines styles for the h1 header */
7 h1 {
8     /* Centers the header text */
9     text-align: center;
10    /* Sets the font size to 4rem */
11    font-size: 4rem;
12    /* Sets the color to grey */
13    color: #grey;
14    /* Adds a top margin of 10% */
15    margin-top: 10%;
16    /* Adds a text shadow of 2px */
17    text-shadow: 2px 2px 2px #000;
18 }
19
20 /* Media query for screens with a maximum width of 600px */
21 @media only screen and (max-width: 600px) {
22     /* Reduces the font size of the h1 header to 3rem */
23     h1 {
24         font-size: 3rem;
25     }
26 }
27
28 /* Code Written by @Arpit Singh 19B0G10069 */
```

- In summary, this code sets the **font family for the body of a webpage** to **Roboto** and defines styles for an **h1 header element**. It **centres the header text**, **sets the font size to 4rem**, **the color to grey**, **adds a top margin of 10%**, and **a text shadow of 2px**. It also includes a **media query that reduces the font size** of the header to **3rem** for screens with a **maximum width of 600px**.

- **That's it!** All the **functional components** required for the **UI have been created** along with the logic of handling/sending **requests to our REST API (Task 1)**.

- **Problem:** the application will run fine with “**npm start**” command and the UI as we defined will also be displayed **but the functionality will not work**. This Problem is caused due to **CORS policies**.

4. Starting REST API and Handling CORS policies problem:

a. Solving CORS policy:

- i. So, this problem arises because my **Rest Api is deployed at the “port: 8080” and my React app is working with “port: 3000”**.
- ii. According to CORS policy a **port cannot send request to another port** until explicitly mentioned or using a proxy.

Error: Access to XMLHttpRequest at 'http://localhost:8080/servers' from origin 'http://localhost:3000' has been blocked by CORS policy: Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource.

- iii. This **error message** indicates that **the browser is preventing the API call from being made due to a CORS policy violation**. The CORS policy is a security feature implemented by web browsers that **restricts web pages from making requests to a different domain** than the one the page was served from.
- iv. To **resolve this issue**, you need to configure your **server to allow cross-origin requests** from the domain that your React application is running on.
- v. One way to **enable CORS on your server** is to add the following **header to the response for each API request**:
Access-Control-Allow-Origin: http://localhost:3000
- vi. To allow cross-origin requests from the specified origin **http://localhost:3000**, you can add the **“@CrossOrigin” annotation** to your **controller class or individual methods**. Here's an example of how to add it to our **“ServerController”** class:

```
"C:\Users\ArpitSG\Kaiburr\Kaibur_Tasks\Task_1\restapi\src\main\java\com\example\restapi\controller\ServerController.java"
```



vii. Therefore we need to make some ***changes in the REST API*** we created in **Task 1**.

```

1 package com.example.restapi;
2
3 import ...
4
5 /**
6  * TSM-ArpitSG
7  */
8 @RestController
9 @RequestMapping("/servers")
10 @CrossOrigin(origins = "http://localhost:3000") // Red arrow here
11 public class ServerController {
12
13     /**
14      * Constructor that injects the ServerService instance into the controller.
15      * @param serverService The ServerService instance to be injected.
16      */
17
18     private final ServerService serverService;
19
20     /**
21      * Handles GET requests for retrieving all servers.
22      * @return A list of all servers.
23      */
24
25     @GetMapping
26     public List<Server> getAllServers() { return serverService.getAllServers(); }
27
28     /**
29      * Handles GET requests for retrieving a server by ID.
30      * @param id The ID of the server to retrieve.
31      * @return The retrieved server.
32      */
33
34     @GetMapping("/{id}")
35     public Server getServerById(@PathVariable String id) { return serverService.getServerById(id); }
36
37 }

```

b. Starting Task 1 Rest Api:

- For the ***requests sent by user*** through the react app the ***REST API must be on***.
- Therefore, **LAUNCH** the application we created in **Task 1** (https://github.com/TSM-ArpitSG/Kaiburr/tree/main/Kaibur_Tasks/Task_1).

```

1 package com.example.restapi;
2
3 import ...
4
5 /**
6  * TSM-ArpitSG
7  */
8 @SpringBootApplication
9 @CrossOrigin(origins = "http://localhost:3000")
10 public class RestapiApplication {
11
12     /**
13      * Main entry point of the application.
14      * @param args Command line arguments.
15      */
16
17     public static void main(String[] args) { SpringApplication.run(RestapiApplication.class, args); }
18
19 }

```

- Now we are good to go! We can *start our React App by running “**npm start**”* 😊.

```
PS E:\Kaibur_Tasks\Task_4\my-app> npm start
> my-app@0.1.0 start
> react-scripts start
  Local:          http://localhost:3000
  On Your Network: http://192.168.0.100:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

5. **UI Testing & Output:**

a. **Create Server:**

Kaiburr Task 4 - UI

Create a Server Object

Server ID:
1

Server Name:
Arpit Singh

Language:
c#

Framework:
spring boot

Create

Server created successfully!



The screenshot shows a browser window with multiple tabs open. The active tab displays a success message: "Server created successfully!". Below this, there is a "Create" button and another message: "Server created successfully!". A "Search by ID" section follows, containing a "Server ID:" input field and a "Search" button. The result of the search is shown in a box with the message "No server found." Below these sections is a table with columns: ID, NAME, LANGUAGE, FRAMEWORK, and ACTION. The table contains one row with data: ID 1, Name Arpit Singh, Language c#, Framework spring boot, and two "Delete" buttons in the ACTION column. A red arrow points to the "Delete" button in the first row.

ID	NAME	LANGUAGE	FRAMEWORK	ACTION
1	Arpit Singh	c#	spring boot	Delete Delete

b. **Search By Id:**

The screenshot shows a browser window with multiple tabs open. The active tab displays a success message: "Server created successfully!". Below this, there is a "Search by ID" section with a "Server ID:" input field containing the value "1" and a "Search" button. The result of the search is shown in a table with columns: ID, Name, Language, and Framework. The table contains one row with data: ID 1, Name Arpit Singh, Language c#, and Framework spring boot. A red arrow points to the "Search" button in the "Search by ID" section, and another red arrow points to the "Delete" button in the first row of the table.

ID	Name	Language	Framework
1	Arpit Singh	c#	spring boot

Below the "Search by ID" section is a "Search by Name" section, which is currently empty with the message "No servers found."

At the bottom of the page is a table with columns: ID, NAME, LANGUAGE, FRAMEWORK, and ACTION. The table contains three rows with data: ID 1, Name Arpit Singh, Language c#, Framework spring boot; ID 2, Name my centos, Language java, Framework spring boot; and ID 3, Name END, Language ggg, Framework best. Each row has a "Delete" button in the ACTION column. A red arrow points to the "Delete" button in the third row.

ID	NAME	LANGUAGE	FRAMEWORK	ACTION
1	Arpit Singh	c#	spring boot	Delete
2	my centos	java	spring boot	Delete
3	END	ggg	best	Delete

c. Search By Name:

Server created successfully!

Search by ID

Server ID: 1

ID	Name	Language	Framework
1	Arpit Singh	c#	spring boot

Search by Name

Server Name: end

ID	NAME	LANGUAGE	FRAMEWORK
3	END	999	best

ID	NAME	LANGUAGE	FRAMEWORK	ACTION
1	Arpit Singh	c#	spring boot	<input type="button" value="Delete"/>
2	my centos	java	spring boot	<input type="button" value="Delete"/>
3	END	999	best	<input type="button" value="Delete"/>

d. Delete:

Server created successfully!

Search by ID

Server ID: 1

ID	Name	Language	Framework
1	Arpit Singh	c#	spring boot

Search by Name

Server Name: end

ID	NAME	LANGUAGE	FRAMEWORK
3	END	999	best

ID	NAME	LANGUAGE	FRAMEWORK	ACTION
1	Arpit Singh	c#	spring boot	<input type="button" value="Delete"/>
2	my centos	java	spring boot	<input type="button" value="Delete"/>
3	END	999	best	<input type="button" value="Delete"/>

Server created successfully!

Search by ID

Server ID: 1

ID	Name	Language	Framework
1	Arpit Singh	c#	spring boot

Search by Name

Server Name: end

ID	NAME	LANGUAGE	FRAMEWORK
3	END	ggg	best

ID	NAME	LANGUAGE	FRAMEWORK	ACTION
1	Arpit Singh	c#	spring boot	<input type="button" value="Delete"/>
3	END	ggg	best	<input type="button" value="Delete"/>

Summary of the steps to create a UI for REST API (Task 4):

1. Choose a UI framework (e.g. ReactJS).
2. Set up a new React project using `create-react-app`.
3. Install necessary packages, such as `axios` for HTTP requests and `react-router-dom` for client-side routing.
4. Create UI components, such as a form component to create new records, a table component to display records, and a delete button component to delete records.
5. Make API requests using `axios` to interact with the REST API.
6. Implement client-side routing using `react-router-dom` to navigate between different pages in the application.
7. Test the application by running `npm start` in the terminal to start the development server and open the application in a web browser.

Explanation on Tools/Technology(s) Used:

❖ NodeJs:

- a. Node.js is used in **this project** as a runtime environment for building the backend of the REST API. Node.js allows developers to use JavaScript to write server-side code, which can handle incoming HTTP requests, interact with databases, and respond to those requests with data or HTML pages. In this case, Node.js is used to create a RESTful API that provides access to the application's data, which can be consumed by the frontend application built with React. Node.js provides a lightweight and efficient way to build scalable and fast server-side applications, making it an ideal choice for building the backend of a REST API



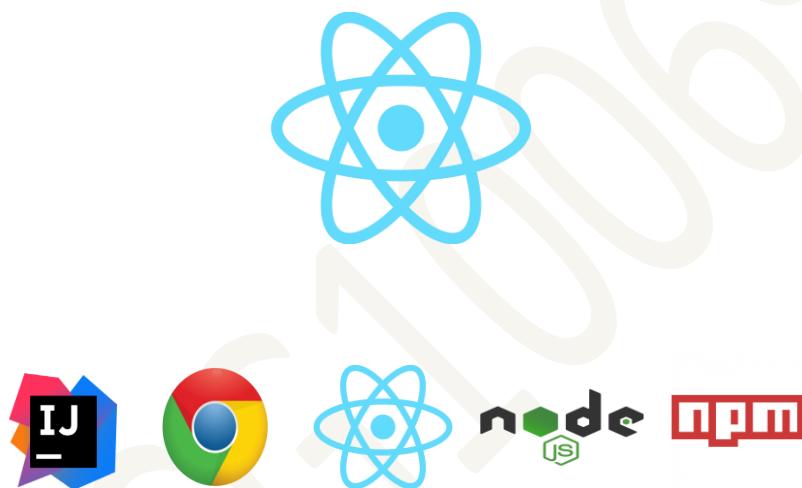
❖ Npm:

- a. In **this project**, npm (Node Package Manager) is used to manage and install the dependencies required for the project. Dependencies are external packages or libraries that are required to run the project. npm allows developers to easily download and manage these dependencies, which can include things like React, Axios, and React Router.
- b. By using npm, developers can simply specify the dependencies they need in the package.json file, and then run a single command (npm install) to automatically download and install all the required packages. This makes it much easier to get started with a new project, since you don't need to manually download and configure all the dependencies yourself.



❖ **ReactJS:**

- a. ReactJS is a popular front-end library used for building user interfaces. In ***this project***, ReactJS is used to create the user interface components of the web application that interacts with the REST API. With React, developers can create reusable UI components that can be used across multiple pages and can easily handle the state of the application. React also allows for easy implementation of client-side routing, which enables the user to navigate between different pages of the application without a full page refresh. ReactJS provides an efficient and easy-to-use way to build complex and responsive web applications with minimal code.



Thank You!
