Hands-on Lab: Creating a Year Validation Component in React and Using Bootstrap Styling

Estimated Time: 1 hour

Introduction

In this practice exercise, you'll create a React component for year validation. The component will allow users to enter a year and check if it falls within the range of 2000 to 2023. Bootstrap is used for styling, and the validation result is displayed using a Bootstrap Alert component.

Objective:

This practice exercise aims to create a functional React component for year validation. This exercise reinforces key concepts related to React state management, event handling, and conditional rendering. By completing this exercise, you will gain hands-on experience building a simple yet interactive user interface for validating a user's input against predefined criteria.
By the end of this exercise, you should be able to:

- Understand how to create a React component that captures and manages user input, specifically a year value entered by the user.
- Implement data validation logic to check whether the entered year falls within a specified range (2000 to 2023 in this case) and respond accordingly.
- Gain proficiency in using React's `useState` hook to manage and update the state of variables like `year` `alertType`, and `message`.
- Implement a user-friendly feedback mechanism by displaying Bootstrap Alert components with appropriate messages and styles, informing the user whether the input is valid.
- Create an interactive user interface that provides real-time feedback as the user interacts with the input field and validation button.
- Apply Bootstrap styling to enhance the visual appeal and responsiveness of the year validation component, ensuring a polished user experience.

Prerequisites

1. React Proficiency: Understanding of React components, state management, event handling, and conditional rendering.
2. HTML and CSS Basics: Fundamental knowledge of HTML for content structure and CSS for styling.
3. Bootstrap Familiarity: Knowing Bootstrap and utility classes.
4. JavaScript Skills: Proficiency in JavaScript for event handling and data validation.
5. React's useState Hook: Experience using React's useState hook for state management.

These prerequisites will prepare you for the exercise of creating a React component for year validation and reinforcing key React concepts.

Task 1: Getting Started

Let's start with cloning the repository.

Clone the Git repository using the below command.

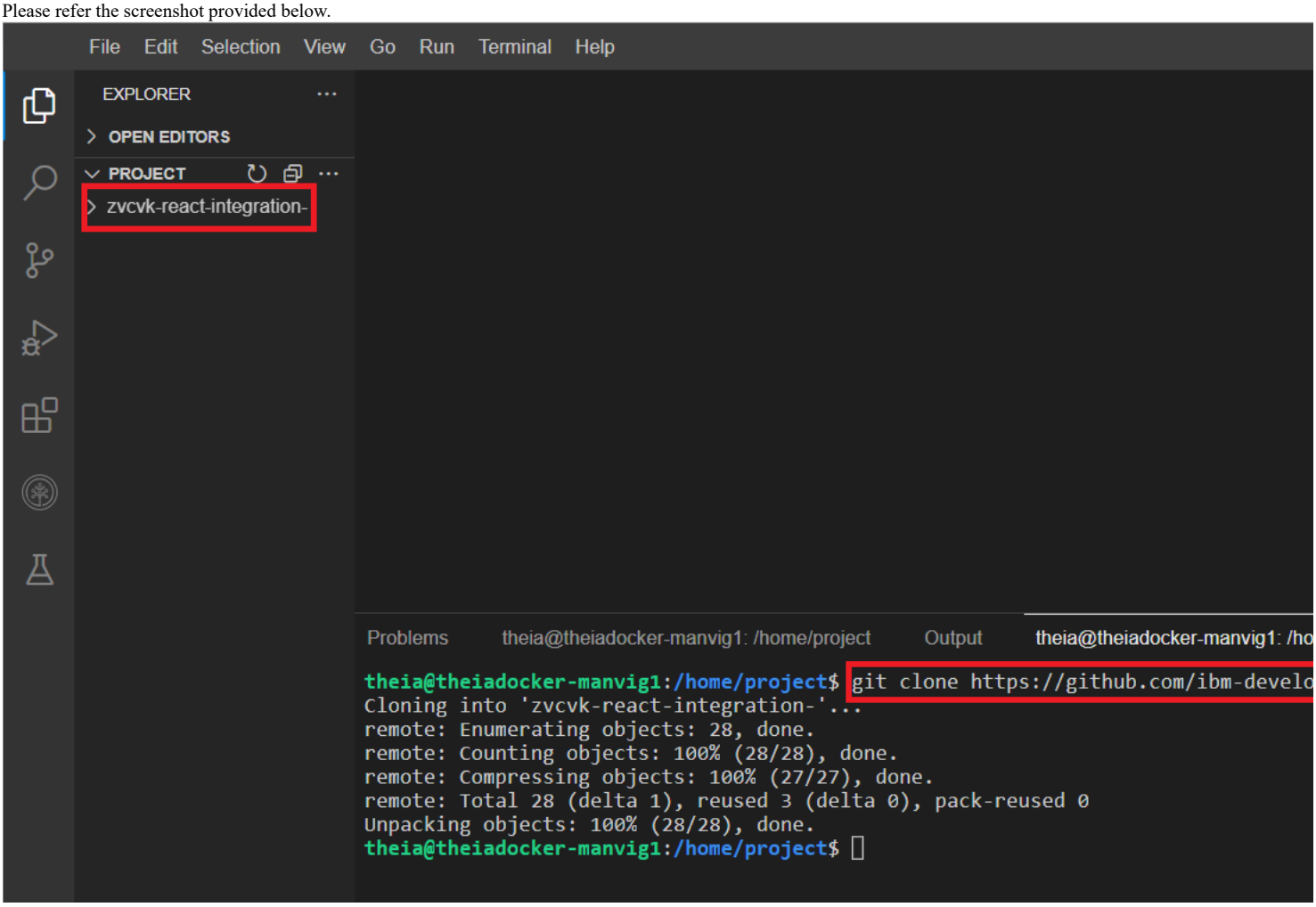
1. Open the Terminal > New Terminal and enter the following command to clone the repository:

```
1. 1
```

```
1. git clone https://github.com/ibm-developer-skills-network/zvcvk-react-integration-
```

Copied!

Please refer the screenshot provided below.

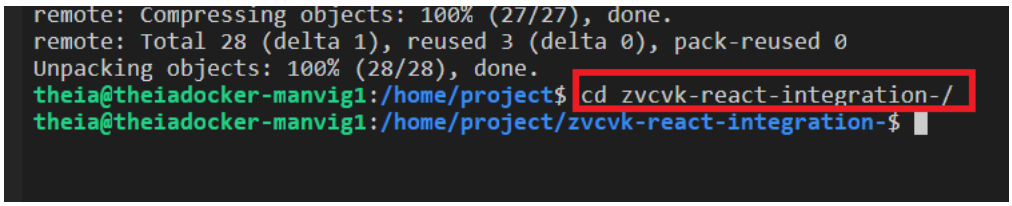


2. Use the command below to change the directory to 'zvcvk-react-integration-'

```
1. 1
```

```
1. cd zvcvk-react-integration-
```

Copied!



3. Run the following command to install dependencies for the project.

```
1. 1
```

```
1. npm install
```

Copied!

```
theia@theiadocker-manvig1:/home/project$ cd zvcvk-react-integration-/
theia@theiadocker-manvig1:/home/project/zvcvk-react-integration-$ npm install

added 1312 packages, and audited 1313 packages in 38s

247 packages are looking for funding
  run `npm fund` for details

6 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 8.1.0 -> 10.1.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.1.0
npm notice Run npm install -g npm@10.1.0 to update!
npm notice
theia@theiadocker-manvig1:/home/project/zvcvk-react-integration-$
```

4. Run the following command to start the React application.

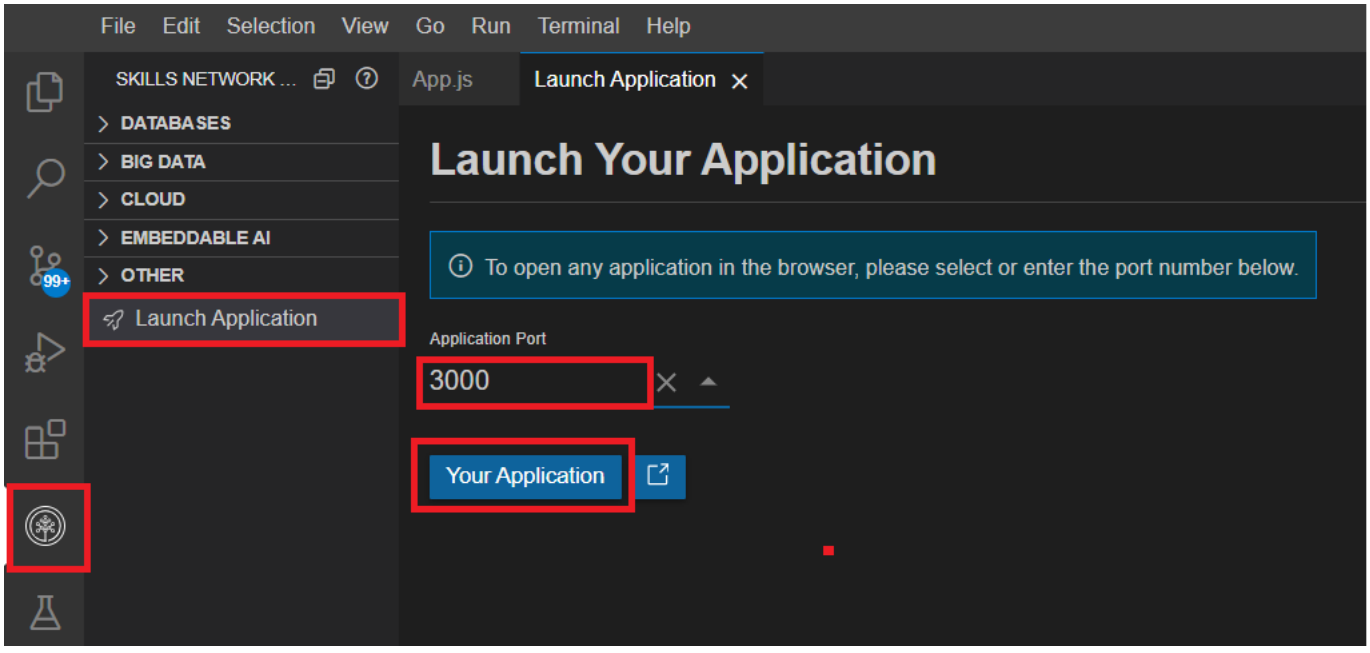
```
1. 1
```

```
1. npm start
```

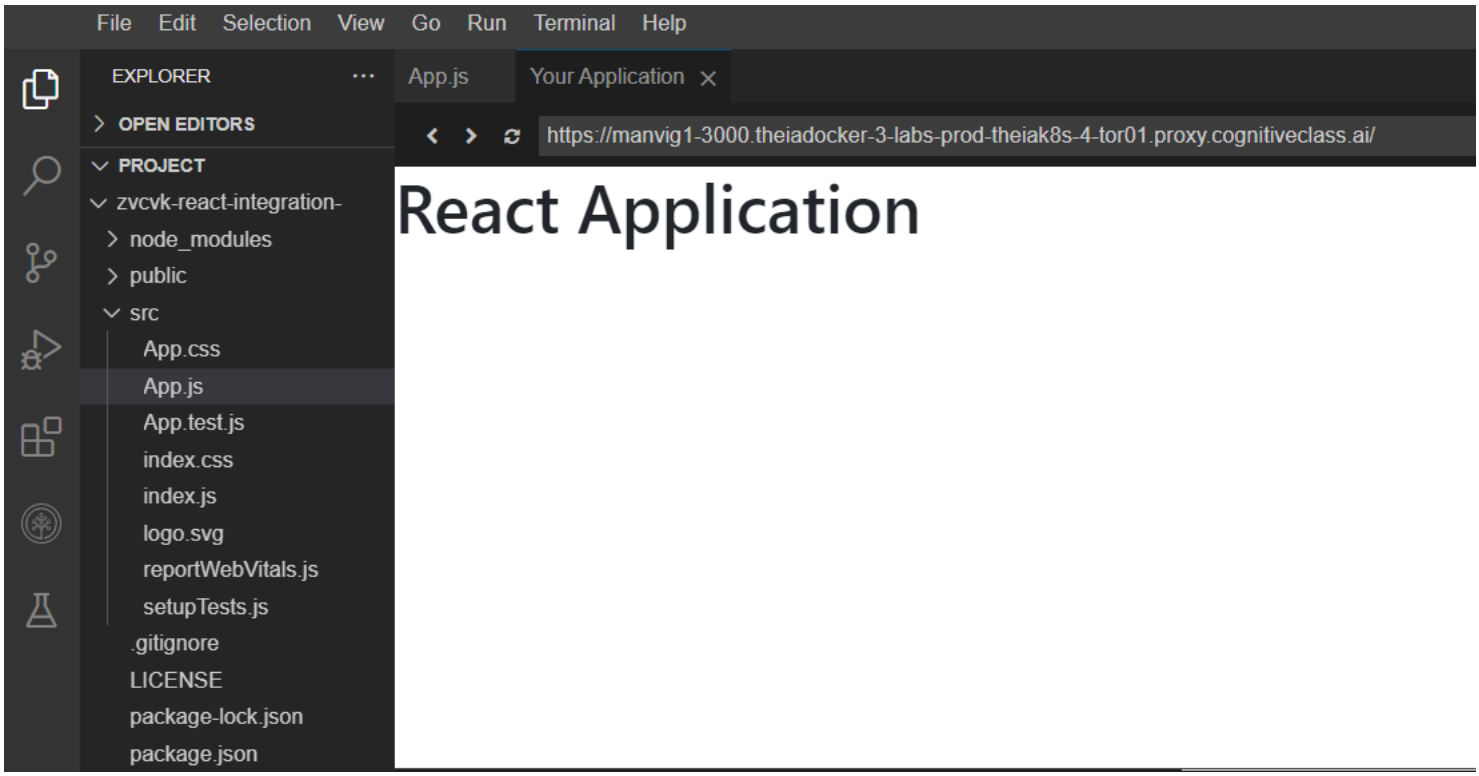
Copied!

```
Problems   theia@theiadocker-manvig1: /home/project   Output   theia@theiadocker-manvig1: /home/project/zvcvk-react-integration-

Compiled successfully!

You can now view yearvalidation in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://172.17.59.85:3000

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

webpack compiled successfully
```

5. Click the Skills Network button on the left to open the "Skills Network Toolbox". Click "Other" then "Launch Application". From there, enter the port number as 3000 and launch your application.



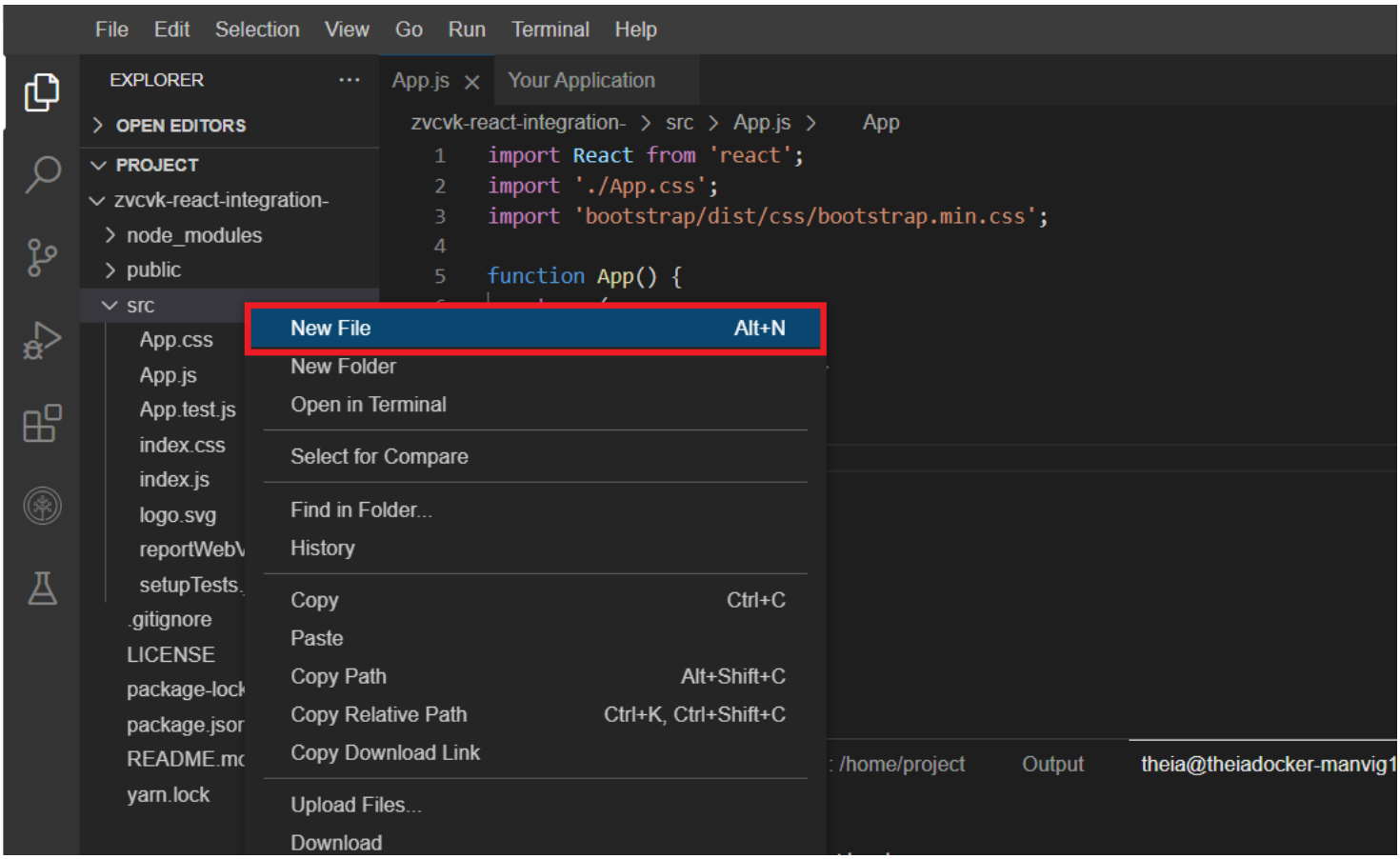
The UI will appear on the browser as seen in the image below.



Task 2: Create the Year Validation Component

1. Inside the "src" folder, create a new file (e.g., `YearValidation.js`) to define the `YearValidation` component.

Please refer to the screenshot provided below.



2. In the `YearValidation.js` file, start by importing the necessary dependencies. You'll need React, `useState` for managing state, and Bootstrap for styling. Add the following imports at the top of your file:

You may refer to the below solution code for the same:

```
1
2
3
```

```
import React, { useState } from 'react';
import { Container, Form, Button, Alert } from 'react-bootstrap';
import 'bootstrap/dist/css/bootstrap.min.css';
```

Copied!

3. Define the `YearValidation` component. Inside the component, set up the following state variables using `useState`:
    - `year`: Initialize this variable as an empty string. It will store the year entered by the user.
    - `alertType`: Also initialize this variable as an empty string. It will be used to determine the type of Bootstrap alert (success or error) to display messages.
    - `message`: Initialize this variable as an empty string as well. It will store the actual success or error message based on the validation result.

You may refer to the below solution code for the same:

```
1
2
3
4
5
```

```
const YearValidation = () => {
  const [year, setYear] = useState('');
  const [alertType, setAlertType] = useState('');
  const [message, setMessage] = useState('');
};
```

Copied!

Task 3: Handle Year Input and Year Validation

1. Create an event handler function called `handleYearChange` to update the `year` state based on user input. This function should be triggered when the user types in the input field.

```
1
2
3
```

```
const handleYearChange = (e) => {
  setYear(e.target.value);
};
```

Copied!

2. Implement the `handleValidation` function, which will be called when the user clicks a validation button or submits a form (not shown in this code). This function will validate whether the entered year is within the range of 2000-2023 and set the appropriate alert type and message.
3. Inside the `handleValidation` function, start by parsing the `year` state variable into an integer using the `parseInt` function. This ensures that you have a numeric representation of the entered year.
4. Check if the parsed year falls within the valid range of 2000-2023.
5. If the parsed year is within the valid range (2000-2023), proceed as follows:
    - Set the `alertType` state variable to 'success'.
    - Set the `message` state variable to a success message indicating that the entered year is within the valid range.
6. If the parsed year is outside the valid range, follow these steps:
    - Set the `alertType` state variable to 'danger'.
    - Set the `message` state variable to an error message indicating that the entered year is outside the valid range, and the user should enter a year between 2000-2023 only.

You may refer to the below solution code for the same:

```
1
2
3
4
5
6
7
8
9
10
```

```
const handleValidation = () => {
  const parsedYear = parseInt(year);
  if (parsedYear >= 2000 && parsedYear <= 2023) {
    setAlertType('success');
    setMessage('Success! Year entered is within the valid range.');
  } else {
    setAlertType('danger');
```

```
    setMessage('Error! Year is outside the valid range, please enter a year between 2000-2023 only.');
  }
};
```

Copied!

Task 4: Render the Component

1. Inside the `YearValidationForm` component, return JSX elements as follows:
    - Use a `Container` with a top margin (`mt-5`) to provide spacing at the top.
    - Include an `<h1>` element with the text "Year Validation"

```
1
2
3
4
5
```

```
return (
  <Container className="mt-5">
    <h1>Year Validation</h1>
  </Container>
);
```

Copied!

2. Create a `Form` element inside `Container` to wrap the following components:
    - Inside the `Form`, add a `Form.Group` component to group related form controls together.
    - Inside the `Form.Group`, insert a `Form.Label` element with the text "Please enter a valid year between 2000-2023" to provide user instructions.

```
1
2
3
4
5
```

```
 <Form>
   {/* Year input field */}
   <Form.Group>
     <Form.Label>Please enter a valid year between 2000-2023</Form.Label>
 </Form>
```

Copied!

3. After the label, add a `Form.Control` element with the following attributes:
    - Type: "number”
    - Placeholder: "Enter year”
    - Value: Controlled by a state variable named `year`.
    - Change event (`onChange`): Controlled by a function named `handleYearChange`.

```
1
2
3
4
5
6
```

```
    <Form.Control
       type="number"
       placeholder="Enter year"
       value={year}
       onChange={handleYearChange}
     />
```

Copied!

4. Below the `Form.Control`, include an `<input>` element with the following attributes:
    - Type: "range"
    - Minimum: 2000
    - Maximum: 2023
    - Value: Controlled by the `year` state variable.
    - Change event (`onChange`): Controlled by the `handleYearChange` function.
    - Apply Bootstrap styling classes to the `<input>` for width and height.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

```
     {/* Range input for year selection */}
```

```
      <input
        type="range"
        min="2000"
        max="2023"
        value={year}
        onChange={handleYearChange}
        className="form-range" // Apply Bootstrap styling classes
        style={{
          width: '80%',
          height: '5px',
        }}
      />
```

Copied!

5. Insert a `Button` component with the text "Validate" This button should trigger the `handleValidation` function when clicked.

```
1
2
3
4
```

```
{/* Validation button */}
      <Button className="mt-3" variant="primary" onClick={handleValidation}>
        Validate
      </Button>
```

Copied!

6. Finally, conditionally render an `Alert` component:

    - Display the `Alert` only when both `alertType` and `message` are truthy values.
    - Set the `variant` prop of the `Alert` component based on the `alertType`.
    - Display the `message` inside the `Alert`.

```
1
2
3
4
5
6
{/* Display the validation result as an alert */}
 {alertType && message && (
   <Alert variant={alertType} className="mt-3">
     {message}
   </Alert>
 )}
```

Copied!

7. Export the `YearValidation` component as the default export.

```
1
```

```
export default YearValidation;
```

Copied!

You may refer to the below solution code for the same:

```
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
```

```
return (
  <Container className="mt-5">
    <h1>Year Validation</h1>
    <Form>
      {/* Year input field */}
      <Form.Group>
        <Form.Label>Please enter a valid year between 2000-2023</Form.Label>
        <Form.Control
          type="number"
          placeholder="Enter year"
          value={year}
          onChange={handleYearChange}
        />
        {/* Range input for year selection */}
        <input
          type="range"
          min="2000"
          max="2023"
          value={year}
          onChange={handleYearChange}
          className="form-range" // Apply Bootstrap styling classes
          style={{
            width: '80%',
            height: '5px',
          }}
        />
      </Form.Group>
      {/* Validation button */}
      <Button className="mt-3" variant="primary" onClick={handleValidation}>
        Validate
      </Button>
    </Form>
    {/* Display the validation result as an alert */}
    {alertType && message && (
      <Alert variant={alertType} className="mt-3">
        {message}
      </Alert>
    )}
  </Container>
);

export default YearValidation;
```

Copied!

▶ Complete solution code for "YearValidation.js"

Task 5: Create the App Component

1. Importing Dependencies:
    - All the necessary react dependencies are imported at the beginning of your file.
    - The Bootstrap CSS file apply Bootstrap styling to your components.
    - Import the `YearValidation` component, which you'll render within the `App` component.

```
1
2
3
4
```

```
import React from 'react';
import './App.css';
import YearValidation from './YearValidation';
import 'bootstrap/dist/css/bootstrap.min.css';
```

Copied!

2. A functional component named `App` is defined. This component represents the main structure of your application.
3. Return JSX Elements:
    - Inside the `App` component, return JSX elements that represent the structure of your application.
    - You should have a top-level `div` element with the class name 'App'.
    - Inside this `div`, render the `YearValidation` component. This is where the year validation interface will be displayed.

```
1
2
3
4
5
6
7
8
9
```

```
function App() {
  return (
```

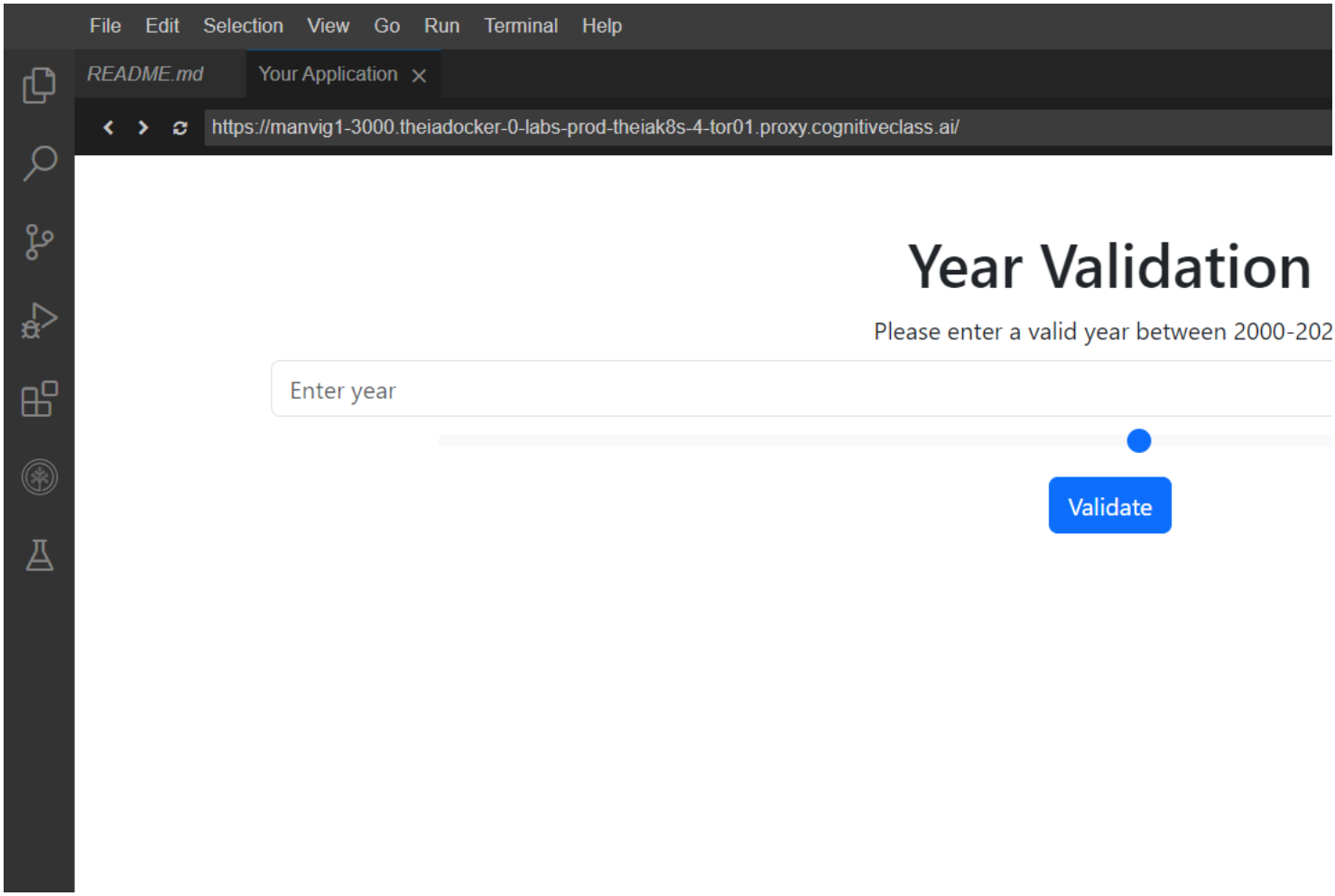
```
    <div className="App">
      <YearValidation />
    </div>
  );
}

export default App;
```

Copied!

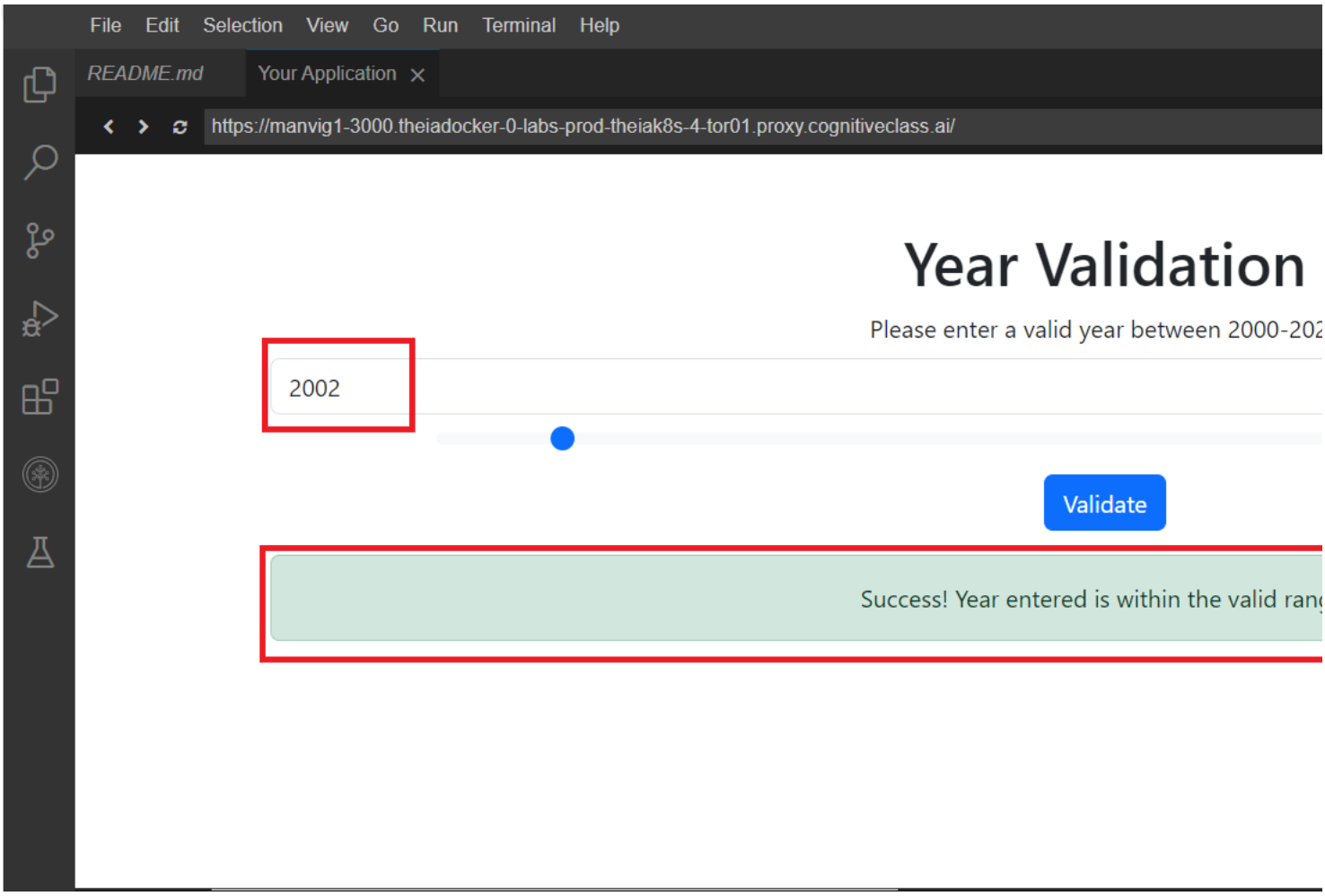
▶ Complete solution code for \"App.js\"

Note: Make sure your React development server is running to test the year validation functionality.

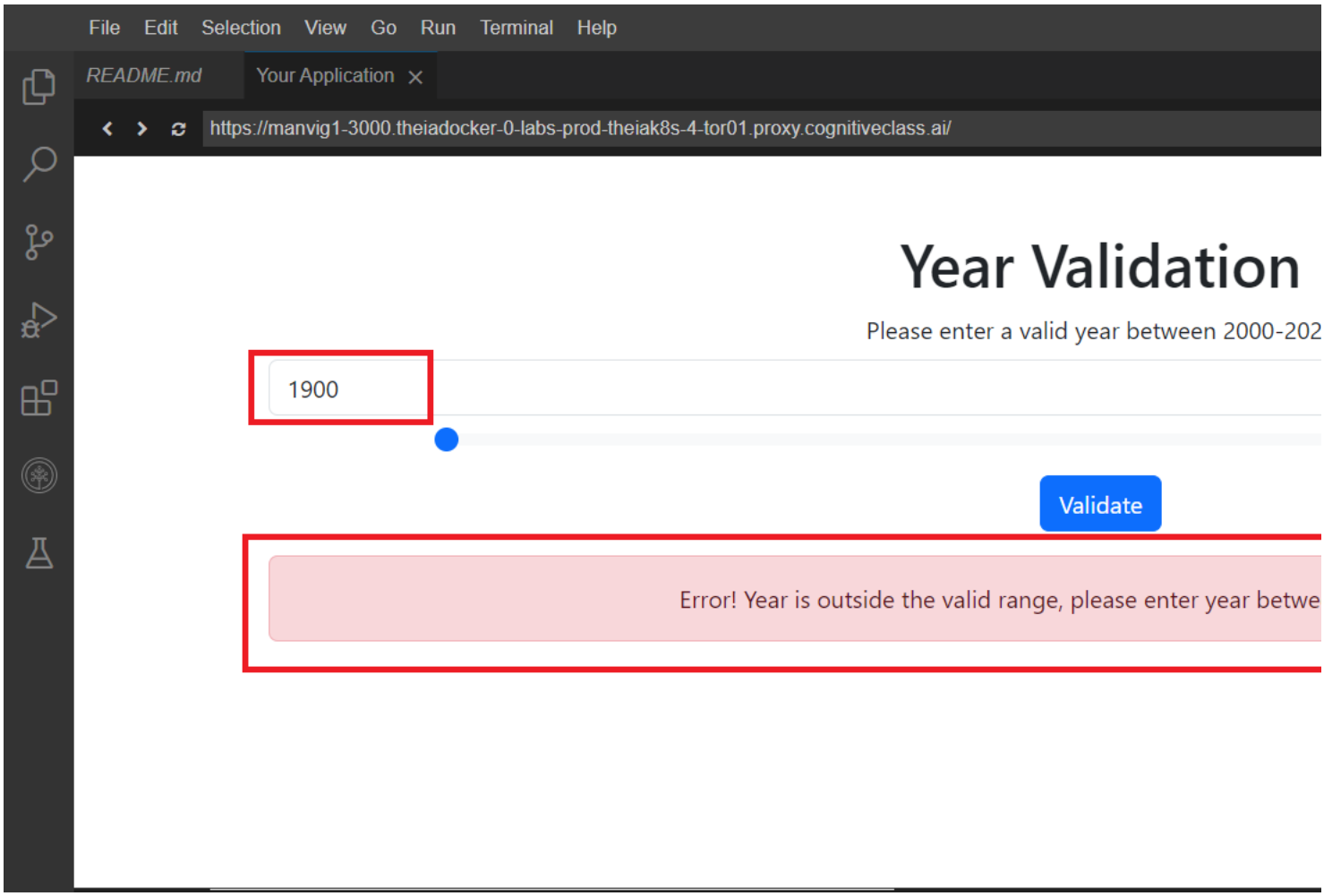
Your final output should look similar to this:



Validation Success



Validation Failed



Note: Bootstrap classes and styles have already been applied to the components for you. You can further customize the styling as needed.

That's it! You've created a React component for year validation with Bootstrap styling. You can further extend or customize this component as needed in your React application.

Congratulations! You have successfully created a React component for year validation with Bootstrap styling.

Author(s)

Rajashree Patil
Manvi Gupta

Changelog

Date	Version	Changed by	Change Description
2023-09-15	0.1	Rajashree, Manvi	Initial version created
2023-09-21	0.2	Mercedes Schneider	QA Pass w/Edits