**Instructions**

Close

**Lab instructions**

0 of 1 completed

Random Number Cards: A Hands-On React App for Beginners

​

**Introduction:**

The lab will focus on creating reusable components, passing data using props, and applying conditional rendering in React. By implementing a random number generator with implicit arrow functions, learners will understand how to work with functions and components in a React application. This lab involves creating two components, App and Card, to build an interactive and dynamic application.

**Goal:**

The goal of this lab is to build a React application that dynamically generates random numbers, passes them to child components using props, and displays different messages based on the generated number using conditional rendering.

**Objectives:**

By the end of this lab, learners will:

1. Learn to create reusable React components.
2. Understand how to use arrow functions with implicit returns.
3. Pass data between components using props.
4. Implement conditional rendering using a ternary operator.
5. Style components with CSS to improve presentation.

**Problem Statement**

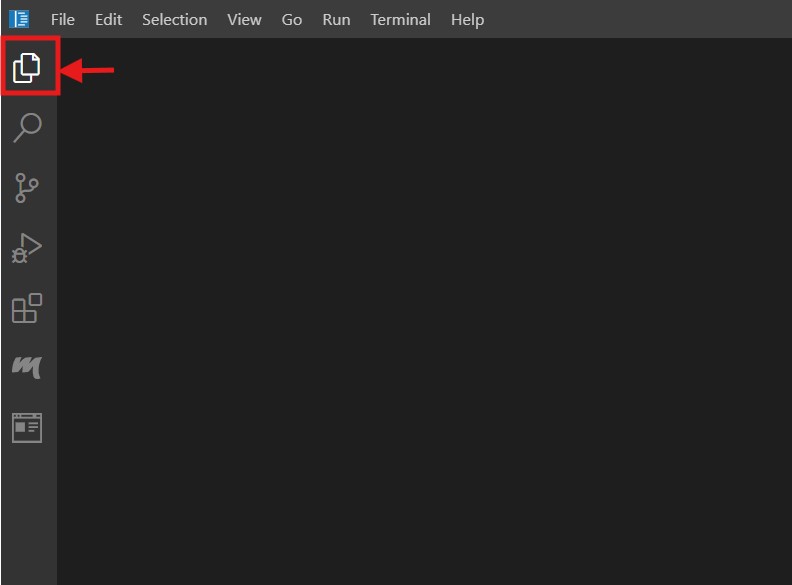
1. An **App** component that generates random numbers using an arrow function and passes these numbers as props to child components.
2. A **Card** component that receives a number as a prop and renders text based on the number's value using a ternary operator.
3. The app will display three Card components, each with a random number and text indicating whether the number is "High" or "Low."

**Instructions:**

**Step 1:** Create the React App

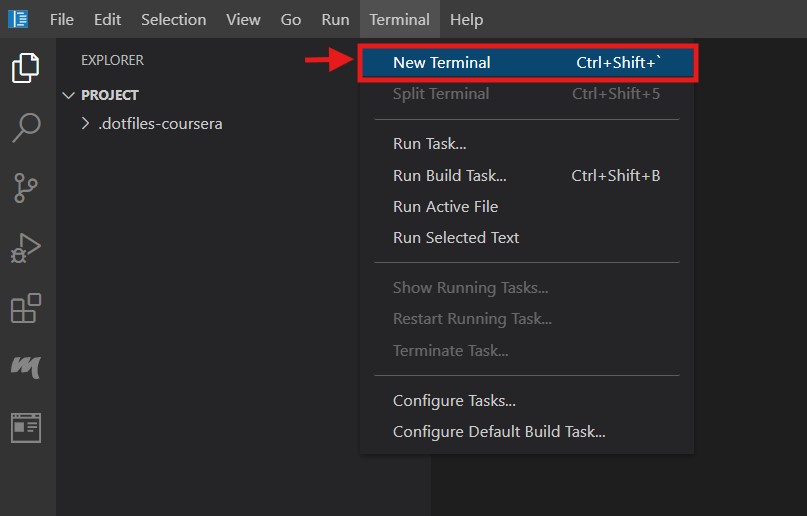
1. Open the Project Folder

* Click on the **Explorer** icon displayed below.
* Select **Open Folder** from the options.
* Navigate to and click on your **Project Folder**.

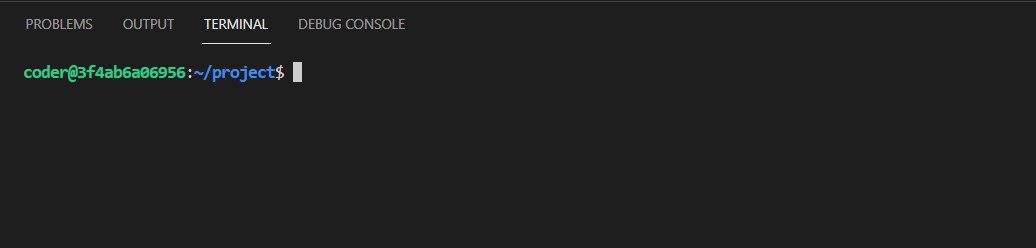


2. Opening a new terminal

* Click on the **Terminal** menu.
* From the dropdown, select **New Terminal**.



3. After opening the new terminal, the terminal panel will appear as shown below, defaulting to the project directory.



4. Run the following command to create the React app named **random\_number\_app**:

1

npx create-react-app toggle\_app

5. The installation process may take some time. Once completed, navigate to the project folder using the following command:

1

cd random\_number\_app

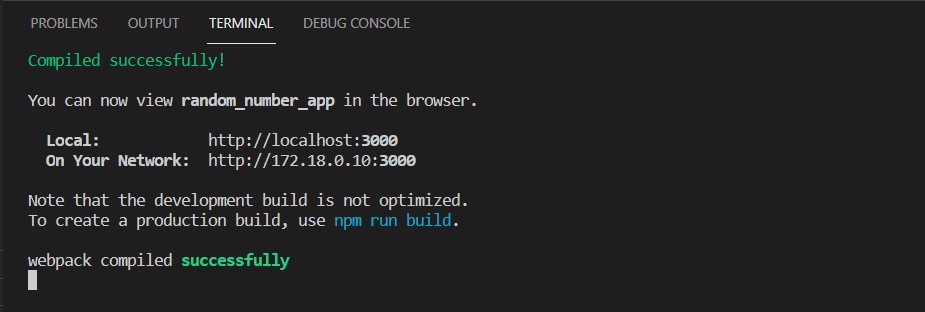
**Step 2:** Start the React Development Server

1. Use the following command to start the development server and test the app:

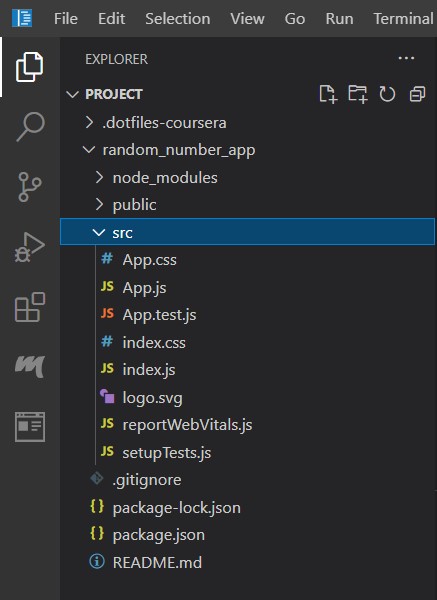
1

npm start

2. You can now view the random\_number\_app in your browser by navigating to localhost:<exposed port> (eg:3000).



3. After successfully installing the React app, the project structure will appear as shown below.



**Step 3:** Create Card.js and Add Code

**Creating Card.js**

1. Right-click on the **src** folder in your random\_number\_app directory.
2. Select **New File** from the context menu.
3. Name the file **Card.js** and press **Enter**.
4. The file is now created, and you can start writing the code.

**Objective**

* This component renders a card that displays a random number and evaluates whether the number is “High” (greater than 50) or “Low” (50 or below).

**Steps to Implement**

* Create a functional component named Card.
* Accept props and use props.num to display the card’s value.
* Add conditional logic to determine whether the card is "High" or "Low".

**code**

6

7

8

9

10

3

4

5

1

2

        </div>

    );

}

export default Card;

        <div className="card">

            <h1>This card's value is {props.num}</h1>

            <p>This card is: {props.num > 50 ? "High" : "Low"}</p>

function Card(props) {

    return (

**Explanation:**

1. **Function Definition**: The Card component is a functional React component that takes props as its argument to receive data.
2. **Container Element**: It returns a <div> element with the class card, serving as the main container for the component.
3. **Dynamic Heading**: Inside the <h1> tag, it dynamically displays the num value passed through props.
4. **Conditional Rendering**: The <p> tag uses a conditional expression to display "High" if props.num is greater than 50, or "Low" otherwise.
5. **Exporting Component**: The Card component is exported as the default export, allowing it to be imported and used in other parts of the application.

**Step 4:** Modify App.js and Add Code

The App.js file is located in the src folder. It already contains some code; delete the existing code.

**Objective**

This component generates and renders three Card elements, each with a random value.

**Steps to Implement**

* Create a functional component named App.
* Use a utility function randNum to generate random numbers.
* Render three Card components and pass random values as the num prop.

**code**

8

9

10

11

12

13

14

15

16

17

6

7

3

4

5

1

2

        <div>

            <h1>Task: Add three Card elements</h1>

            <Card num={randNum()} />

            <Card num={randNum()} />

            <Card num={randNum()} />

        </div>

    );

}

export default App;

function App() {

    return (

const randNum = () => Math.floor(Math.random() \* 100) + 1;

import "./App.css";

import Card from "./Card";

**Explanation:**

1. **Import Statements**: The code imports the App.css file for styling and the Card component from a separate module for reuse.
2. **Random Number Generator**: A function randNum is defined to generate a random number between 1 and 100, which will be passed as a prop to the Card components.
3. **App Component**: The App function defines a React functional component that serves as the main structure of the application.
4. **JSX Structure**: Inside the App component, a div contains an h1 heading and three Card components, each receiving a unique random number via the num prop.
5. **Export**: The App component is exported as the default export, making it accessible for use in other parts of the application.

**Step 5:** Style the App

The **App.css** file is located in the **src** folder. It already has some existing code; remove it and replace it with the following code for styling and improved visual appeal.

43

44

45

46

47

48

49

50

51

52

53

.card h1 {

    font-size: 1.5em;

    color: #007bff;

}

.card p {

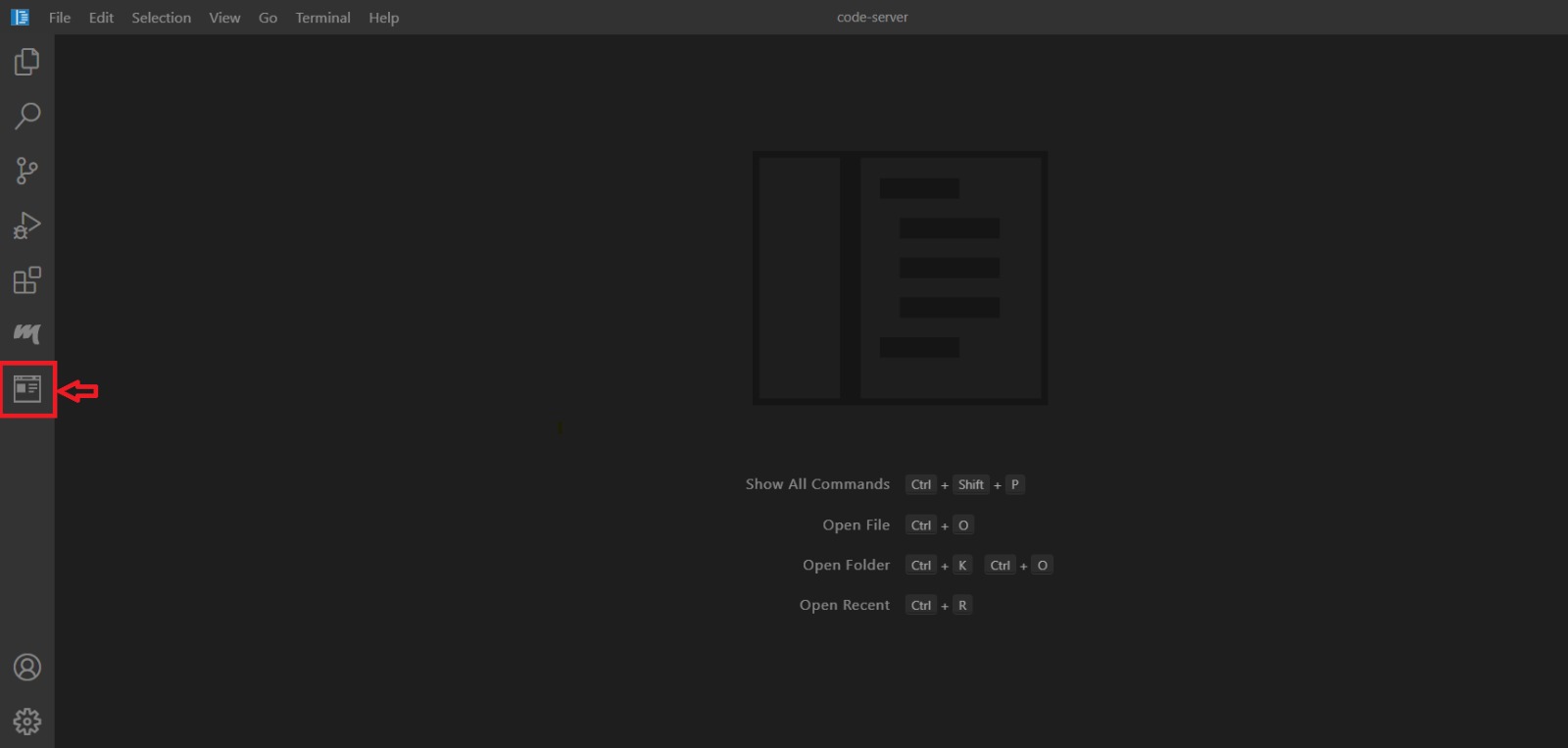
    font-size: 1em;

    color: #333;

}

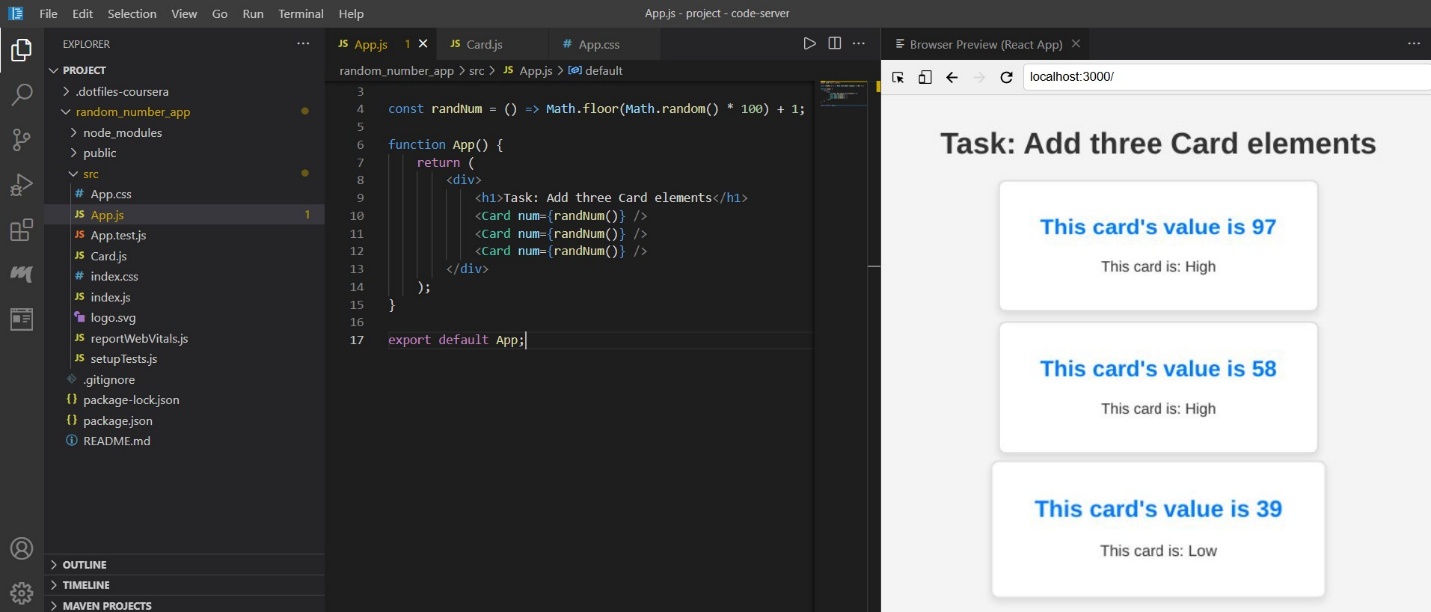
**Step 6:** view the output

1. To view the output, click on the **Browser Preview** icon located on the left panel. It is the last icon in the panel.



2. In your browser, enter the following URL format, replacing <exposed port> with the actual port number displayed in the editor when the server was initially started: http://localhost:<exposed port>.

3. After entering the URL correctly, the webpage will load, displaying a preview of the webpage as shown below.



**Step 7:** Close the **server** after completing the lab:

Once you’re done with the lab, make sure to **close the server** to free up the port:

* You can stop the server by pressing **Ctrl + C** in the terminal.

**Key Takeaways:**

**1. Understanding React Components**:

* React applications are built using components. Components are the building blocks of React apps, where each component manages its own structure and behavior.
* In this example, you created an App component and a Card component. The App component is responsible for managing the logic and passing data (via props) to the Card component.

**2. Using Props for Data Passing**:

* **Props** are used to pass data from a parent component (App) to a child component (Card).
* In this app, the App component passes a random number generated by the randNum function to the Card component as a prop. The Card component then uses this prop to display the number and decide whether it is "High" or "Low".

**3. Arrow Functions and Implicit Return**:

* The randNum function in the App component uses an arrow function with an **implicit return** to generate a random number between 1 and 100.
* Arrow functions allow for cleaner and more concise code, especially when you don't need a function body.

**4. Conditional Rendering with Ternary Operator**:

* Inside the Card component, a **ternary operator** is used to render different text based on the value of the number passed as a prop.
* If the number is greater than 50, the card displays "High". If it's 50 or less, it shows "Low". This is an example of **conditional rendering** in React.

**5. Reusable Components**:

* The Card component is reusable and can render different data depending on the props passed to it. You can pass different numbers to the Card components, and each will display its own value and the corresponding "High" or "Low" label.

**Final Step: Mark as Completed**

* Click the **Mark as Completed** button present below to mark the lab as **Completed**.