**Assignment 6**

The step-by-step guide you provided for completing **Assignment 6: Build the Test Start Component** is quite detailed and well-structured. However, it can be optimized slightly for clarity and to ensure that all aspects of creating the TestStartButton component are covered comprehensively.

Here is your instruction with some minor improvements:

**Step 1: Create the TestStartButton Component**

1. Inside the src/components folder, create a new file called TestStartButton.tsx:

bash

Copy Code

Run

1touch src/components/TestStartButton.tsx

1. Open TestStartButton.tsx and add the following code:

tsx

Copy Code

Run

1import React, { useState } from 'react';

2

3const TestStartButton: React.FC = () => {

4 const [isStarted, setIsStarted] = useState(false);

5

6 const handleClick = () => {

7 setIsStarted(true); // Update state to indicate the test has started

8 };

9

10 return (

11 <button

12 onClick={handleClick}

13 disabled={isStarted}

14 style={isStarted ? styles.disabledButton : styles.button}

15 >

16 {isStarted ? 'Test Started' : 'Start Test'}

17 </button>

18 );

19};

20

21const styles = {

22 button: {

23 padding: '10px 20px',

24 fontSize: '16px',

25 backgroundColor: '#007bff',

26 color: 'white',

27 border: 'none',

28 borderRadius: '5px',

29 cursor: 'pointer',

30 transition: 'background-color 0.3s ease',

31 },

32 disabledButton: {

33 padding: '10px 20px',

34 fontSize: '16px',

35 backgroundColor: '#6c757d',

36 color: 'white',

37 border: 'none',

38 borderRadius: '5px',

39 cursor: 'not-allowed',

40 },

41};

42

43export default TestStartButton;

**Explanation:**

* The useState hook is used to manage the isStarted state.
* The button’s label and disabled state change based on the isStarted value.
* Basic styles are applied for the button in its default and disabled states.

**Step 2: Update the Dashboard Component**

1. Open src/components/Dashboard.tsx and update it to include the TestStartButton component:

tsx

Copy Code

Run

1import React from 'react';

2import TestStartButton from './TestStartButton';

3

4const Dashboard: React.FC = () => {

5 return (

6 <div style={styles.dashboard}>

7 <h2>Welcome to the Dashboard!</h2>

8 <p>This is a placeholder for the test overview or instructions.</p>

9 <TestStartButton />

10 </div>

11 );

12};

13

14const styles = {

15 dashboard: {

16 padding: '2rem',

17 textAlign: 'center' as const,

18 },

19};

20

21export default Dashboard;

1. Save the file.

**Explanation:**

* The TestStartButton component is imported and rendered within the Dashboard component.

**Step 3: Run and Test the Application**

1. Start the development server if it is not already running:

bash

Copy Code

Run

1npm run dev

1. Open your browser and navigate to http://localhost:5173.
2. Test the TestStartButton:
   * **Initial State**: The button should display "Start Test" and be enabled.
   * **After Clicking**: The button should change to "Test Started" and become disabled.

**Deliverables**

* TestStartButton.tsx: The implemented component.
* Dashboard.tsx: The updated file integrating the TestStartButton.
* Screenshots or a screen recording showing the button’s behavior.

**Example Screenshots**

1. **Initial State**:
2. **After Clicking**:

**Submission**

1. Submit the TestStartButton.tsx and Dashboard.tsx files.
2. Include screenshots or a screen recording showing the button’s behavior.

**Improvements & Additional Notes**

1. **Component Styling Adjustments**: Consider CSS Modules or styled-components for more scalable styling solutions:

bash

Copy Code

Run

1npm install styled-components @types/styled-components

1. **State Management Enhancement**: If your application grows, consider using state management libraries like Redux or Context API for better management.
2. **Error Handling**: Include error handling (e.g., network errors) if the button triggers an asynchronous action.