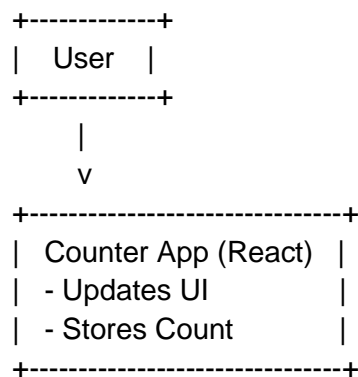


Data Flow Diagram (DFD) for Counter App

This Counter App consists of a single React component (App.js) that maintains a counter using the useState hook. The user clicks a button, which updates and displays the counter value.

Level 0: Context Diagram

At the highest level, the user interacts with the Counter App, which updates the state and displays the count.

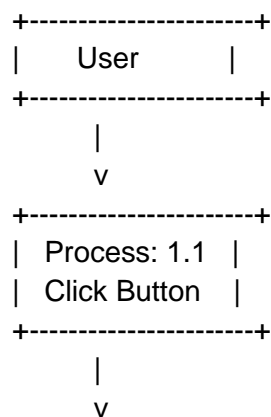


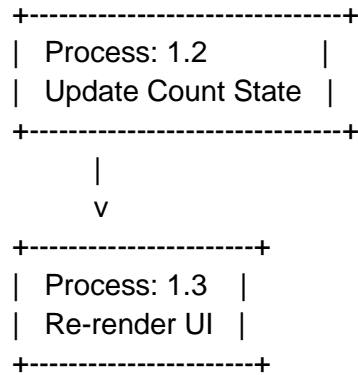
Explanation:

- **User:** Clicks the button to increase the count.
- **Counter App:** Processes the click, updates state, and re-renders the UI.

Level 1 DFD (Decomposition of Process)

Breaking down the **Counter App** into core processes:





Explanation:

1. Process 1.1 - Click Button

- The user clicks the "Click Me" button.

2. Process 1.2 - Update Count State

- `setCount(count + 1)` updates the counter state.

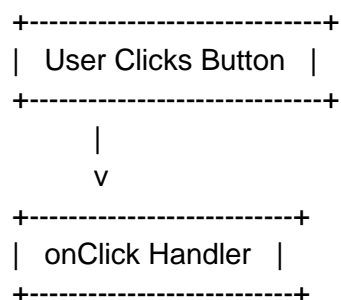
3. Process 1.3 - Re-render UI

- React re-renders the component, displaying the updated count.

Level 2

Now, let's further break down each process.

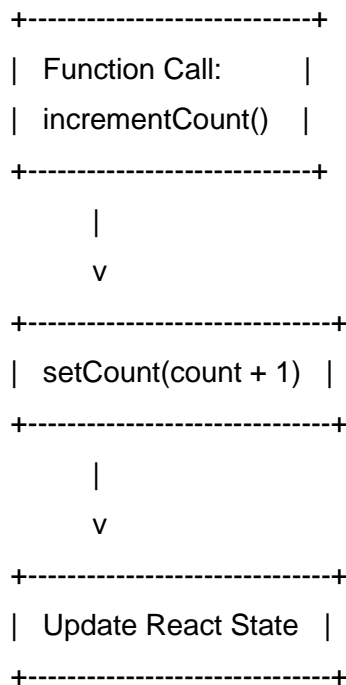
1.1 Click Button



Explanation:

- When the button is clicked, the `onClick` event triggers `incrementCount()`.

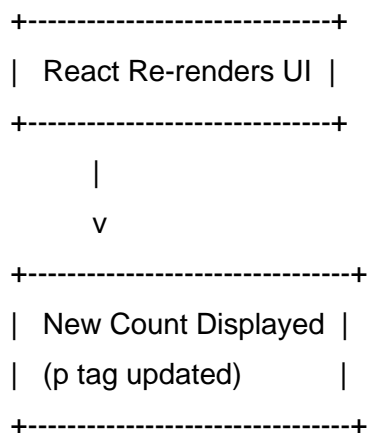
1.2 Update Count State



Explanation:

- `setCount(count + 1)` updates the state variable **count**.
- React stores the new value in memory.

1.3 Re-render UI



Explanation:

- The `<p>` tag displaying "You clicked {count} times" is **updated with the new count**.
- React automatically updates the UI without refreshing the page.

Data Flow:

Entities:

1. **User** – Clicks the button.
2. **Counter App (React Component)** – Manages state and updates the UI.

Processes:

1. **Click Button** – Triggers the function to update count.
2. **Update Count State** – Increments count in React state.
3. **Re-render UI** – Displays the updated count.

Data Stores:

1. **React State (count)** – Stores the counter value.

Additional Notes & Improvements:

1. Optimizations:

- If multiple clicks happen quickly, use functional updates:
- `setCount(prevCount => prevCount + 1);`

2. Enhancements:

- Add a decrement button to decrease count.
- Store count in local storage to persist across refreshes.
- Implement a reset button to reset count to zero.