REACT DEBUGGING

Problem 1: useCallback Hook Warning

Issue: In `src/bugs/HookSideEffects.js`, there was a warning related to the `useCallback` hook:

```
React Hook useCallback does nothing when called with only one argument. Did you
forget to pass an array of dependencies? react-hooks/exhaustive-deps
```

The `useCallback` hook was used without a **dependency array**, which caused the warning. Without the second argument, `useCallback` cannot optimize the function's performance because it doesn’t know when to recreate the memoized function.

Code with Issue:

```
javascript
Copy code
const track = useCallback((analyticsEvent) => {
    setEventBatch((batch) => [...batch, analyticsEvent]);
});
```

Solution: We solved this by adding a **dependency array** as the second argument to `useCallback`. In this case, we used `[setEventBatch]` to ensure the function is only recreated when the `setEventBatch` function changes. This resolves the warning and ensures the `track` function is properly memoized.

Fixed Code:

```
javascript
Copy code
const track = useCallback((analyticsEvent) => {
    setEventBatch((batch) => [...batch, analyticsEvent]);
}, [setEventBatch]);
```

Problem 2: useEffect Unused Variable Warning

Issue: In `src/bugs/PassingProps.js`, there was a warning because `useEffect` was imported but never used:

```
'useEffect' is defined but never used no-unused-vars
```

This happens when a variable (in this case, the `useEffect` hook) is imported but not used in the component, causing ESLint to flag it as an unused variable.

Code with Issue:

```
javascript
Copy code
import { useEffect, useState } from "react";
```

Solution: We resolved the issue by **removing the unused** `useEffect` import since it wasn't needed in the component. This cleaned up the code and removed the warning.

Fixed Code:

```
javascript
Copy code
import { useState } from "react";
```