Task 3: Written Comparison (15 points) Write a short report (250–350 words) that:

1. **Explains the difference between the imperative (jQuery) and declarative (React) implementations.**

The difference between imperative (jQuery) and declarative (React) approaches is about how the interface is managed. With the imperative approach, a developer writes a sequence of instructions to the browser, such as finding an element and manually adding a class. This often results in tightly linked code that is difficult to maintain and change as the application grows or changes are necessary. In contrast, React uses a declarative approach, where the developer describes how the UI should look in various states, and the library takes responsibility for updating the display.

2. **Describes how React's approach (state + re-render) avoids manual DOM steps.**

React's state-based, re-rendering approach completely eliminates the need for manual DOM manipulation. This is achieved by using a virtual DOM—an in-memory representation of real elements. When the state changes, React compares the new state with the previous one, also stated as "diffing", and automatically performs the minimum necessary updates (patching) to the real DOM. This makes updates much more efficient.

3. **References Ch. 1, "Declarative UI structures" and "Data changes over time".**

Chapter 1, in the section "Declarative UI Structures," explains that a declarative approach using JSX avoids manual and complex DOM manipulation. Instead of performing a chain of steps to change content, the developer simply returns a JSX structure. The section "Data Changes Over Time" emphasizes that each component rendering is a snapshot of the interface at a specific point in time, based on the current data.

4. **Discusses why the declarative approach scales better (e.g. more elements, more complex UI).**

The declarative approach scales significantly better when building complex interfaces. As was stated before, while imperative code becomes a mess as the number of elements grows, declarative components remain predictable and easy to understand. Using components as building blocks allows for complex UIs to be organized into hierarchical structures that are easier to manage and provide high performance even with frequent data changes.