Apply recognised principles to the building of high-quality software components, analysing the difference between safety and quality, and show how mechanisms that improve quality may be used to partially underpin a safety argument.

I applied principles to React components, such as modularity, ensuring that components have well-defined function and clear responsibility. I also handled API errors using try/catch and managed independent state like loading by react useState hook. I used async/await approaches for asynchronous data handling. (Kloda, 2025a) My team practiced pull requests and code reviews, which helped identify potential issues early, improve code quality. (Kloda, 2025b) I created Jest test, which reduce the probability of software faults. (Kloda, 2025c) However, I noticed I could refactor it, creating reusable hook and improving data validation to prevent runtime errors. I also plan to add automated testing.  
Analysing the difference between safety and quality, I realised that quality could exist without safety. By focusing on user experience and developing requirements I was able to provide good quality, but it didn’t ensure safety code. It could still be vulnerable to crashes or unexpected behaviours.   
Applying quality principles supports the safety by reducing the risk of systematic failures, increasing confidence in software behaviour, and enabling more robust threat mitigation.

# References

Kloda (2025). [online] github. Available at: https://github.com/Jkloda/movie\_recommendation\_system/blob/main/client/src/components/SemanticSearchBar.js [Accessed 2 May 2025].

Kloda (2025). [online] github. Available at: https://github.com/Jkloda/LOJuliaKloda/blob/main/Assets/PRs.png [Accessed 2 May 2025].

Kloda (2025). [online] github. Available at: https://github.com/Jkloda/movie\_recommendation\_system/blob/main/client/src/tests/SemanticSearchBar.test.js [Accessed 9 May 2025].