Project Overview

The goal of this project was to develop a web-based application that converts numbers between roman and natural numerals.

Additionally, the interface has been inspired by the Roman Empire and instances of Roman soldiers when the user writes a number on the web.

Tools & Technologies Used

- **HTML**: To structure the webpage and its components.
- **CSS**: To style the interface, including colors, backgrounds, and layout improvements.
- **JavaScript**: To implement the conversion logic and dynamically generate Roman soldier images based on user input.
- **ChatGPT:** To write the necessary code in HTML, CSS and javaSCript for the correct operativity of the web.

1. Requirements-Based Quality

This refers to how well the software meets explicit, standard, and implicit requirements.

- **Explicit requirements**: The converter correctly allows users to input a number (Roman or Natural) and provides a conversion.
- **Standard requirements**: The application follows basic web development standards (HTML, CSS, and JavaScript separation).
- **Implicit requirements**: The UI is intuitive, but additional usability improvements, such as real-time validation, could be included.

2. Human Perspective-Based Quality

This focuses on user experience and expectations.

- The interface has a **thematic design** with Roman Empire elements, improving engagement.
- The probability-based **soldier image generation** adds a dynamic effect.
- However, the background image may not always scale correctly, affecting readability.

3. Internal vs. External Quality

- **External Quality**: The application functions correctly from the user's perspective. However, **error messages are minimal**, which could lead to confusion.
- **Internal Quality**: The JavaScript functions are well-structured, but the lack of unit tests makes the code harder to maintain and validate.

4. Based Quality (Maintainability, Usability, Reliability, Efficiency, Portability)

- **Maintainability**: The code is structured but lacks documentation. Comments on key functions would improve it.
- **Usability**: The UI is visually appealing but could benefit from real-time conversion instead of requiring a button click.
- **Reliability**: Works consistently, but validation for inputs is missing.
- **Efficiency**: The code is efficient, but adding performance optimizations (e.g., debouncing input events) could improve responsiveness.
- **Portability**: Runs well on different browsers, but mobile responsiveness should be tested.