Reflection and Traceability Report on RwaveDetection

Junwei Lin

1 Changes in Response to Feedback

The following subsections organize the feedback received throughout the project according to the corresponding deliverables: the Software Requirements Specification (SRS) and Hazard Analysis, the Design and Design Documentation, and the Verification and Validation (VnV) Plan and Report.

1.1 SRS and Hazard Analysis

This section summarizes the modifications made to the SRS document in response to the feedback received. The revisions primarily focused on improving document structure and enhancing content clarity, as recommended by Dr. Smith in issue #13.

As shown in Table 1, the following changes were made in response to the feedback received.

Feedback Description	**PR Number**
Refine the description of intended readers to better reflect their academic and technical background.	#35
Clarify the scope of requirements by explicitly defining what is included and excluded from the project.	#35
Improve clarity in the description of system constraints, specifically the requirement for low-performance embedded devices.	#35
Clarify the terminology and definitions, such as for QRS complex and R-wave.	#35
Refactor functional and nonfunctional requirements to ensure consistency and clarity.	#35

Table 1: Summary of Changes for SRS and Hazard Analysis

1.2 Design and Design Documentation

This section summarizes the modifications made to the MG and MIS document in response to the feedback received. The revisions primarily focused on improving the clarity of the module hierarchy and enhancing the explanation of each module's functionality, as Dr. Smith recommended in issue #21. In addition, Domain Expert Baptiste raised an issue related to document hyperlinks in #25., which made the document easier to read.

As shown in Table 2, the following changes were made in response to the feedback received according to the MG Document.

Feedback Description	**PR Number**
Refactor the module hierarchy to improve clarity and structure.	#35
Clarify the services provided by the different modules, especially the Rwave Detect and Math modules.	#35
Improve the description of implementation methods and algorithms used in the Pan-Tompkins Algorithm and Specified IIR Filter modules.	#35
Refine the explanations of the Input Output Processing and Error Handler modules.	#35

Table 2: Summary of Changes for MG Document

As shown in Table 3, the following changes were made in response to the feedback received according to the MIS Document.

Feedback Description	**PR Number**
Add previously missing Environment Variables for certain modules.	#37
Correct the transition semantics in the Access Routine section.	#37
Add several hyperlinks to improve navigation between modules.	#26

Table 3: Summary of Changes for MIS Document

1.3 VnV Plan and Report

2 Challenge Level and Extras

2.1 Challenge Level

[State the challenge level (advanced, general, basic) for your project. Your challenge level should exactly match what is included in your problem state-

ment. This should be the challenge level agreed on between you and the course instructor. —TPLT]

2.2 Extras

[Summarize the extras (if any) that were tackled by this project. Extras can include usability testing, code walkthroughs, user documentation, formal proof, GenderMag personas, Design Thinking, etc. Extras should have already been approved by the course instructor as included in your problem statement. —TPLT]

3 Design Iteration (LO11 (PrototypeIterate))

The final design of the R-peak detection system retained the core structure of the Pan-Tompkins algorithm without modifications to its detection logic. The primary design iteration focused on handling abnormal or invalid input cases more effectively. During early testing, issues were identified when the system encountered empty, non-numeric, or improperly formatted ECG data, which could lead to runtime errors or undefined behavior. To address this, input validation and exception handling mechanisms were introduced, ensuring that the system could safely detect and report invalid inputs without interrupting execution. These adjustments improved the robustness and user safety of the application, aligning the implementation with practical usage scenarios where unexpected or corrupted input data might occur.

4 Design Decisions (LO12)

[Reflect and justify your design decisions. How did limitations, assumptions, and constraints influence your decisions? Discuss each of these separately. —TPLT]

5 Economic Considerations (LO23)

This project is a reimplementation of the Pan-Tompkins algorithm for R-peak detection in ECG signals and is intended as an open-source tool rather than a commercial product. There are no associated production costs or pricing considerations. The project will be shared on a public repository platform with clear documentation and example usage to attract developers, students, and researchers in biomedical signal processing. Its value lies in providing a reliable, accessible, and well-documented implementation for educational use and integration into other health-related applications.

6 Reflection on Project Management (LO24)

6.1 How Does Your Project Management Compare to Your Development Plan

The project closely followed the initial development plan. The technologies outlined in the plan, such as Docker for containerization, Google Test for unit testing, GitHub Actions for continuous integration, `cpplint` for code style enforcement, and `cppcheck` for static code analysis, were all implemented as intended. These tools facilitated a streamlined development process and ensured code quality and consistency.

6.2 What Went Well?

Several aspects of the project management were successful:

- **Tool Integration:** The integration of Docker, Google Test, GitHub Actions, `cpplint`, and `cppcheck` into the workflow enhanced development efficiency and code reliability.
- **Continuous Integration:** GitHub Actions provided automated testing and analysis, allowing for immediate feedback on code changes and reducing the likelihood of introducing errors.

6.3 What Went Wrong?

One challenge encountered was with the implementation of Git pre-commit hooks. Initially, pre-commit hooks were planned to automatically run tests before each commit. However, issues related to file read/write permissions prevented certain tests, especially those involving file operations, from executing correctly in the pre-commit context. Since the continuous integration pipeline via GitHub Actions already performed comprehensive testing, maintaining the pre-commit hooks was deemed redundant and subsequently removed from the workflow.

6.4 What Would You Do Differently Next Time?

In future projects, the following adjustments would be made:

- **Evaluate Tool Redundancy:** Assess the necessity and potential overlap of tools like pre-commit hooks and continuous integration pipelines to avoid redundant efforts.
- **Focus on Effective Tools:** Prioritize tools that offer the most value and align with the workflow, ensuring that each tool serves a distinct and necessary purpose.

- **Early Testing of Tools:** Conduct thorough testing of all planned tools in the early stages of the project to identify and address any compatibility or functionality issues.

7 Reflection on Capstone

This section summarizes the relevant academic courses and additional technical skills that contributed to the successful completion of the R-wave detection project.

7.1 Which Courses Were Relevant

The capstone project for R-peak detection primarily drew on knowledge from several relevant courses. The *C++ Programming* course provided essential skills for system implementation, including memory management, modular programming, and testing frameworks. Additionally, the *Digital Signal Processing* and *Signals and Systems* courses were directly applicable, as they covered the theoretical foundation of ECG signal processing, filtering techniques, and time-domain analysis, which are critical components of the Pan-Tompkins algorithm.

7.2 Knowledge/Skills Outside of Courses

Beyond coursework, several additional skills were required for this project. Experience with continuous integration tools, such as GitHub Actions, was necessary to establish automated testing and analysis pipelines. Knowledge of Docker was also acquired to manage a consistent development environment. Furthermore, familiarity with code analysis tools like `cpplint` and `cppcheck` was developed to ensure code quality and compliance with C++ standards. These tools and practices, although not covered in previous courses, were essential for maintaining a reliable and maintainable codebase in a modern software development workflow.