Quality Assurance Concept – *Think Outside the Room*

Team Collaboration

Our team organizes the project through a combination of regular in-person meetings and online coordination via our Discord channel. We meet every week to discuss progress, resolve open issues, and plan the next development steps. Task assignments are not fixed from the beginning—instead, we distribute responsibilities after each milestone, based on the current project state and the needs of the next phase.
This flexible approach helps us adapt quickly to changes while ensuring balanced participation from all team members. Continuous communication is key to our workflow: we stay in touch throughout the week using Discord, where we hold online meetings, share updates, ask questions, and collaborate in real time. During both in-person and online sessions, we also present our completed work to receive feedback and improve the overall quality of the code.

Version Control

For version control, we use GitHub. Our team follows a clear branching strategy to ensure the stability of the main branch and the traceability of all development efforts. The `main` branch contains only stable and tested code. Each team member creates their own feature branch based on the latest state of `main`, and commits changes frequently with descriptive commit messages.
Merging into the main branch is only permitted after the following criteria are fulfilled:

- The code has been tested locally.
- A code review has been completed by at least one other team member.
- The merge is discussed and approved by the group.

In order to maintain traceability and transparency, we follow a commit message convention that includes meaningful summaries of the changes.

Code Review

All changes to the main branch go through a peer review process. Each pull request must be reviewed and approved by at least one team member. The reviewer focuses on:

- Code readability and structure,
- Correct use of exception handling,
- Logical consistency and test coverage,
- Adherence to project coding standards.

In some cases, pair programming is used especially for complex parts of the project such as the GUI or networking logic. This improves code quality and ensures that more than one team member is familiar with key components of the codebase.

Coding Standards

To ensure readability, maintainability, and consistency in our codebase, we follow standard Java conventions throughout the project. We have agreed on a common style guide within

the team, including rules for naming, indentation, and structure. Our IDEs are configured to automatically format code on save, which helps us maintain clean code with minimal manual formatting effort.
We use JavaDoc for method and class documentation to clearly explain functionality.
We also maintain a comprehensive `README.md` file at the root of the repository.

Tools

The analysis was conducted using the MetricsReloaded plugin integrated into IntelliJ IDEA. This tool provides a comprehensive set of static code analysis metrics, including cognitive complexity, cyclomatic complexity, design complexity, and code documentation metrics. It allowed us to evaluate the evolution of code quality across project milestones.

Measurements

For our project, we measured the following metrics:

Number of code lines, JavaDoc lines, and comments per class/method

Class:

Milestone	Av. Comment lines of Code	Av. Javadoc lines of Code	Av. Lines of Code
MS2	4.22	4.18	15.96
MS3	3.23	2.69	15.05

Method:

Milestone	Av. Comment lines of Code	Av. Javadoc lines of Code	Av. Lines of Code
MS2	35.26	29.56	95.41
MS3	27.85	19.21	91.92

Complexity

Milestone	Cognitive (Total / Avg.)	Cyclomatic (Total / Avg.)	Design (Total / Avg.)
MS2	417 / 2.37	424 / 2.48	352 / 2.06
MS3	1,185 / 2.82	1046 / 2.52	903 / 2.18