1 Measures

In this document, we will be outlining our approach to ensuring that the final product of our project is of the highest possible quality. As a team of four, we understand that any project of this nature can be prone to errors, bugs and glitches. Therefore, it is our goal to identify and mitigate these issues early on in the development process, to ensure that we deliver a smooth and enjoyable game experience to our users. The following is an exhaustive list of the steps we are taking to ensure the final product is robust, reliable, and bug-free.

- Defining clear requirements: Before starting the development, we should define clear and specific requirements for the game. These requirements should cover all aspects of the game, from its gameplay mechanics to its visual design and user interface. Knowing what we are planning on programming in the near future will allow us to organise our code in an appropriate way: it will then be easier to refactor or modify to implement features later on.
- Conducting thorough testing: We will conduct extensive testing throughout the development process, including unit testing, integration testing and user acceptance testing (exterior opinions and suggestions). These tests will cover critical, as well as non-critical parts of our code if we see the need for it. The tests will most likely be built with JUnit tests, and the code coverage measured with Jacoco.
- If they are to arise, major bugs will be reported on either discord in a dedicated text channel or on GitHub. Each of those bug reports will contain the following information:
  - Commit hash
  - Operating System / Environment
  - Description of the problem
  - How to reproduce it
- We will be splitting up in groups of two to program the server and the client sides of the project. In each pair, both developers shall keep their colleague updated on their progress, explaining their code and what it does. After having code explained to them, team members will review the code and give feedback to improve it.
- Documenting the code: We will document our code thoroughly with Javadoc, making it easier for each and every one of us to understand how the code works and to identify any issues or bugs if they are to arise.
- Additionally, we will set some standards for our code, in order to keep it as clean, concise and efficient as possible:
  - Write short methods. If the code gets to hefty, redistribute in sub-methods that can be reused from other places in the code.
  - Use Javadoc before and within methods if needed.
  - IntelliJ allows us to see the number of method calls of each method as well as the number of usages of variables. This can be used to estimate how meaningful our methods are.
  - We will use either Jacoco or Metrics Reloaded to evaluate how well we have implemented the previous points and modify the code if they report inconsistencies or illogical choices.
- We want to use a logger library. We will try to go with Log4j2, to help us debug our code and save the outputs in log files.