Non-Functional Requirements

We have decided on three non-functional requirements. They will be described in this short document by order of priority. Moreover, this document will explain how we will try to approach our code to get the requirements fulfilled.

1)Performance

The reason why we want this requirement to be on top is that in any game you need to have an optimum responsiveness. If the game lags (meaning the actions are delayed because of problems on the server or the client), the user experience decreases a lot.

To reach this requirement, we will try to have the code be optimal in the sense of having methods not running into loops too big. Or also having algorithm run in a maximum of O(nlog(n)). (This idea will also be applied to the requirement (2).

2)Scalability

We want our system to be scalable. Meaning that the server should be able to handle a big amount of user without having to invest in more servers. After looking up online the different types of scalability[1], we will mainly focus on functional scalability, and administrative scalability. Therefore, we will strive to have methods and classes able to handle new methods without much struggle for whoever implements the new functionality.

Other actions taken to insure a good scalability will be to have our algorithm run into a maximum of O(nlog(n)) if possible. And, to minimize the exchange of data between client and server so the server can handle more request at the same time.

3)Maintainability

Finally, we want our software to be maintainable. This means that if a bug arises, or an exploit was found we want whoever oversees the system. Even if it is not a member of the original development team to be able to easily navigate the code and find his way around it. Another scenario where maintainability is desirable is if someone (not necessarily from the original team) wants to modify the app to implement new features or change it. That person should be able to find his way around the code without having to spend an absurd amount of time to figure out what a method is doing. However, note that while understanding what a method does, does not mean understanding how it does it, meaning that the documentation will exists for the method, but it will not always be precisely explained how each algorithm is implemented. Although if a method is long, complicated or key to understanding the code, we will provide extra information on how the code works.

To insure this requirement, we will make sure that all classes and packages are clearly separated. That the documentation exists and is kept up to date. We will also provide a clear class diagram that explains how each class relate to each other.

[1] At : https://en.wikipedia.org/wiki/Scalability