

Draw it or Lose it!

**CS 230 Project Software Design Template**

Version 1.0

**Table of Contents**

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 04/03/2022 | Karama | Used this document so updated everything. |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

**Executive Summary**

Creative Technology Solutions (CTS) wants to build a larger client than what's available for Andriod users. The three chosen platforms to increase the clients are: Linux, Mac, and Windows. In addition, there's to be possibility for it to be ran on mobile platforms, and how long the platforms can run with recommended requirements.

**Design Constraints**

The game needs to be hosted on a server that outputs the working program to the client's screen. In addition, there needs to be a way to show the game through different playforms, which html allows for. For this project, we'd need the correct program language and APIs to work with the client's devices.

**System Architecture View**

Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

**Domain Model**

Game, Team, and Player all have an "is a" relationshp with Entity, meaning they're all a part of Entity. These three have all inherited from Entity, which is shown in this UML. Since Game, Team, and Entity have common attributes like ID and name, Entity is defined as a super class.

Team and Player have a "has a" relationship, as Team has players. Similarly, Game has Teams, and GamesService has Games, which is represented by Aggregation in the UML.



**Evaluation**

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| **Server Side** | flexible terminal commands for the server, to gain access to, and to make changes | flexible terminal commands for the server, to gain access to, and to make changes, plus more cost friendly | more software is available for this compared to the rest | best if server stays still and can be tracked |
| **Client Side** | moderate knowledge and time needed, cost about the same as windows | Takes a lot of knowledge and time, cost the least | almost no knowledge and time needed | provides flexibility for clients and developers, more difficult to implement |
| **Development Tools** | Objective C  Visual Studio  PyCharm  Eclipse  Github | C, C++, CSS, Java, JavaScript, HTML, PHP, Python, Linux (language)  same tools as Mac | C++, Java  same tools as Mac | Java  same tools as Mac |

**Recommendations**

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

* **Operating Platform**: When expanding Draw It or Lose It, the best I'd suggest for platform would be Windows, as this allows the developers to be able to maximize the platforms the games can be compatible with.
* **Operating Systems Architectures**: x86 for windows is one of the best for developments of games. This is already used massively and has the compability we're looking for.
* **Storage Management**: With The Gaming Room, we could use a data structure that promotes a store of a collection of objects and files, called a Stack.
* **Memory Management**: Windows uses its own virtual address space, which allows for it to process information from its own machine, allowing for a massive amount of its address space to be able to be used.
* **Distributed Systems and Networks**: Network games usually use a database shared with the players, which takes interactions from the players. Having a distributed software would have a high performance application for the client, as separate OS would be combined through the similar program. Connectivity would depend on the server, along with outages, as we're relying on a constantly running program with Lose it or Draw it.
* **Security**: When looking at security between various platforms, it's important to have a certification necessary before users have access to the server. The certification can be a purchase requirement or a login ID that's unique and deemed hard to crack. These choices (while one means it's a paid-to-play app) result in an application where security is important and focused on when having the game be available to other platforms. Currently, anything that's deemed as risky is given a few warnings and is even hard to install or to get to work on devices that the application has failed to pass when security of someone's private information is at stake.