

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/14/2021 | Andrew Jones | Updated Executive Summary, Design Constraints, and Domain Model |
| 1.1 | 07/15/2021 | Andrew Jones | Started Evaluation and Recommendations |
| 1.2 | 07/31/2021 | Andrew Jones | Updated Evaluation and Recommendations |
| 1.3 | 08/13/2021 | Andrew Jones | Completed Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

Creative Technology Solutions (CTS) will create a web-based application of the existing Android-based game “Draw It or Lose It” capable of serving multiple platforms for the client, The Gaming Room. The current software requirements for the game are as follows: a game will have the ability to have one or more teams involved, each team will have multiple players assigned to it. game and team names must be unique to allow users to check whether a name is in use when choosing a team name, and only one instance of the game can exist in memory at any given time. A multi-class java application that follows the existing rules and gameplay of the existing Android app will be developed in order to best serve the client’s needs.

## [Design Constraints](#_2et92p0)

The design must be able to potentially host multiple teams, each with several players. The more teams and players the client desires to have, the more robust the application must be, along with the hosting hardware to ensure that the game runs smoothly which will result in an increase in the overall budget. The ability to host large amounts of players will also affect the timeline in which the project can be completed. Regardless of that, the Java application will be developed with portability in mind to ensure the game is able to run on multiple platforms seamlessly. The game will also have to be designed to mimic the existing game that the client has already deployed including style and gameplay.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

The Entity class exists to hold common attributes and behaviors that can be used by each of its child classes. Therefore, GameService, Game, Team, and Player all inherit from Entity in order to access commonalities including simple variables, methods, etc. Each of the child classes also has access to one another so that they may interact with each other and share information. Outside of these connected classes exists the ProgramDriver class which serves as the “kick-starter” for the application and containing the main method. The ProgramDriver class also uses the SingletonTester to ensure that there is only ever one instance of GameService in memory at one time.

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac is one of the best for anti-malware. It is the most expensive of the three as the user must purchase an Apple device . Simple yet powerful GUI based OS. Less compatible for gaming when compared to Windows or Linux. MacOS Server software from Apple makes using a Mac as a server rather easy and powerful. | The least used operating system with only 1% of users. Less prone to cyber threats though there are some known weaknesses in its architecture such as the directory structure. Linux is free, making it the most cost-effective. It is open-source, so it is fully customizable. However, it is not very user-friendly and is less compatible than Windows for gaming. Drivers for hardware are much harder to find than on other OS. | Windows is used by 90% of users worldwide. It is the most malware prone of the three. Windows is expensive up-front for licenses, but less so than Mac. User-friendly and highly customizable with extensive support. Not preferable for multi-user situations. | Simple and user-friendly. Unable to upgrade hardware, and existing hardware is not particularly robust when compared to non-mobile devices. Less prone to malware than other OS. Newer than other OS so there is less support. Large user base. |
| **Client Side** | Mac is the most expensive both in terms of hardware and software, and it has a steeper learning curve than Windows. Mac applications can only be developed on a Mac device. It is extremely easy to deploy Mac applications to Mac users, however, reducing cost. More easily secured than Windows or mobile software. | Linux has the steepest learning curve, and, therefore, will have the largest timeframe restrictions. However, it is also the cheapest option as it is open-source. It is also arguably the most secure of the options besides Mac. | The most widely used OS, therefore, the learning curve is much lower, and less expertise is needed for development. Arguably the best for gaming, it also has the widest range of supported software making it one of the best values. The latest Windows deployments are slower than their competitors which affects development time. | Often very costly to develop and maintain as mobile apps need updated and maintained more frequently. Both iOS and Android use their own native languages so development can take longer, require more expertise, and is more costly. One main advantage is mobile devices tend to be very user-friendly |
| **Development Tools** | Swift, Objective-C, C#, C++, JavaScript, Java, and Python are all common languages to develop with on Mac. FlexiHub and Xcode are popular development tools. | Just about any language can be used but most popular are Python, C++, C, Perl, and Java. SublimeText and NetBeans are common IDE’s. | C-based languages like C# and C++ are popular along with Python, Java, Javascript, and PHP. Microsoft Visual Studio, Eclipse, and PyCharm are popular development tools. | React Native is best for cross-platform development. Kotlin and Swift are used for iOS while Python and C are used for Android. Xcode, Microsoft Visual Studio, and Xamarin are popular IDE’s. |

## Recommendations

1. **Operating Platform**: Windows would be the ideal operating platform for The Gaming Room.
2. **Operating Systems Architectures**: Windows is used by 90% of users worldwide, it is less costly than Mac for development, has extensive support available, and it is arguably the best for gaming. Linux could also be considered, however, there is a steeper learning curve associated with development which could hinder development time.
3. **Storage Management**: Storage should be handled through a cloud storage solution such as AWS or Microsoft Azure. This is due to the relatively small file size (roughly 1.6 gigabytes for 200 high-resolution images) and the ability to add to or modify the existing image library quickly and easily without having to mess with physical hardware. Cloud storage also eliminates the need to estimate the amount of physical storage needed and it hinders potential growth.
4. **Memory Management**: Regardless of whether the application is 32-bit or 64-bit architecture, Windows divides the virtual address space into two equal sectors: user and kernel space. In the case of a 32-bit architecture, there is a physical limit of 232 – 1 or roughly 4 GB regardless of the amount of RAM a system has. 64-bit Windows architecture allows for much greater virtual address space: 8 TB for user space and 8 TB for kernel space. By using a 64-bit architecture for Draw It or Lose It, all of the round’s image files and their individual renderings can easily be stored in virtual memory and easily accessed without paging which results in much higher performance overall. Though with that being said, the application is small enough to be written in 32-bit architecture and still function well.
5. **Distributed Systems and Networks**: In today’s day and age, an application can be easily distributed amongst clients using a serverless architecture and it can be great as it eliminates the need to focus on the infrastructure and development can be focused entirely on the application itself which cuts costs dramatically. Serverless architecture also aids in scalability as it eliminates the need to add or provision servers on-the-go as the application grows. In the case of serverless architecture, the vendor handles all of the messy underlying infrastructure, and the developer can focus on improving the game. There are potential drawbacks to serverless architecture including cold-start latency; however, that can be minimalized.
6. **Security**: One of the main benefits of using a popular OS such as Windows is that security is already a large consideration. For instance, when it comes to memory allocation, Windows keeps the virtual address space private for processes and it cannot be accessed by other processes unless it is explicitly shared. There is also a significant amount of security features built into the Java language such as automatic garbage collection, automatic bounds checking of arrays, exception handling, etc., and as long as the developer uses the correct programming standards, then the JVM handles the rest.