

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)4

[**Recommendations**](#_m8aleynsvzvc)6

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 2.0 | 08/21/2021 | Jason Arendt | Completed recommendations |

**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](#_sbfa50wo7nsh)

Our new client, The Gaming Room, has an Android based game titled “Draw It or Lose It” that they would like to use as a foundation for a multi-platform, web-based game. The game is a multiplayer game based on a 1980’s tv game show. The game is played by teams of people with the goal of answering a puzzle by using stock images from a stored library. Each team has a 1-minute timer to guess based on the slowly rendering image over the course of 30 seconds. If they cannot guess the answer it is then passed to the other team who then have 15 seconds to guess. The team with the most correct guesses after 4 rounds is the winner.

While the team at The Gaming Room has experience building the game in Android, they are not as versed in building up web-based games. Our task is to help them stream-line the process of adapting from Android to web-based and ensuring the game works on Android, IOS, Windows, Linux, and any other mainstream platform that might be used.

## [Design Constraints](#_2et92p0)

* Use the design of the Android version of the game as a template to create a similar gaming experience in the web-based version as the Android version.
* Build up the web-based version of the game in a way that takes into account the difference in capabilities between the Android interface and the web-based interface.
* Being that JavaScript is so engrained in web-based programming, it would make for an excellent choice in building the web-based Draw it or Lose it.

## [System Architecture View](#_ilbxbyevv6b6)

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](#_8h2ehzxfam4o)

Entity the parent for Game, Team and Player. So, Game is an Entity, Team is an Entity, Player is an Entity. They not only have their own instructions, but they also inherited the instruction set from Entity. An example of this is for Player class which is created with its base tools, but since Player extends Entity, it also has access to all the tools within Entity. This also applies to Team and Game, giving each class an extended list of tools with which to function.

Game, Team and Player are interconnected in that Game must reference the Team class and the Team class must reference the Player class. GameService creates the instances of Game, which there can be many different copies of. Each Game is different, due to the singleton, and each game has multiple Team and Player classes. Game can create new Teams, Team can create new Players and each time there is a check to make sure there are not duplicates. In each Game there can only be multiple Teams and Players, but each Team and Player must be unique.

Each step of the creation process checks for duplicate classes to ensure the game only has a single instance of any of them at a time. By designing the program in this manor, it ensures the game only has a single instance running of each game, multiple players and teams can be created, but no duplicates can be created in a single game.

****

## [Evaluation](#_2o15spng8stw)

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** | Options for configuration and access are strong. Although Macs are not preferred for this web hosting, they are used and there are options that make them viable as web servers. | Linux systems are one of the more preferred options for web hosting as they are secure and open source. Being open source creates a strong foundation for bug fixes and troubleshooting knowledge. The downside to Linux is that it is less user friendly and requires a higher degree of knowledge to use effectively, increasing cost of use. This breaks even with the fact that the OS is open source, so it balances the cost a little. | Windows server has been more widely used, so there is a wealth of software and knowledge currently available, although this comes at a premium cost. Being a preferred platform, the Windows server is robust and more user friendly. The downside to the user-friendly aspect is it takes away some of the power from the user, making the system more susceptible to attacks. | While current mobile technology is powerful, mobile device server side would struggle. This is due to the reliance on mobile data rates, limited power availability and lack of overall use of mobile servers. Server systems are better suited as stationary systems that allow other devices to be mobile. |
| **Client Side** | Mac is very similar to Windows in that development on it is user friendly, which keeps cost and time down. Availability of documentation, software tools and knowledge are readily available. | With Linux being open source, the cost for the OS and software tools are a minimum. The cost for this OS comes from the advanced platform architecture and knowledge needed to develop on it. | Similar to Mac, windows development is user friendly with documentation and software tools available for reasonable costs. Windows, being a more common OS makes it less costly and complex to develop for. | Cross platform compatibility is more complicated with mobile devices, which drives up cost. Also, the knowledge needed to use and develop on a mobile device is high. The upside to mobile is the ability to develop and update anywhere the device gets data signal. |
| **Development Tools** | Dev tools for Mac include Eclipse, Visual Studio, Notepad++ and PyCharm, to name a few. Mac can run any general-purpose language such as Python, Java, C/C++, HTML, CSS, JS  Objective-C is the primary programming language of Mac | Dev tools for Linux include Eclipse, Visual Studio, Notepad++ and PyCharm, to name a few. Linux can run any general-purpose language such as Python, Java, C/C++, HTML, CSS, JS | Dev tools for Windows include Eclipse, Visual Studio, Notepad++ and PyCharm, to name a few. Windows can run any general-purpose language such as Python, Java, C/C++, HTML, CSS, JS. The main language used in creation of windows is C++, which is a very powerful and widely used language with OOP in mind. | For mobile devices, apps can be created using Android or Swift. These languages can be run on mobile devices as well as desktops. Mobile devices can run general-purpose languages such as Python, Java, C/C++, HTML, CSS, JS.  Java is the most common Android app dev language and Swift is the preferred language of IOS. |

Recommendations

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

1. **Operating Platform**: Being one of the most used operating systems currently around, Windows would be a good base to build the game on. There are plenty of resources to utilize and a vast knowledge base for any potential issues. It is user friendly and familiar to a large portion of the population.
2. **Operating Systems Architectures**: Windows gives developers the ability to program in basically whatever environment they choose with minimal effort. Along with that, the access to system resources, file management, memory access, web services and graphical output, with little effort.
3. **Storage Management**: Storage management is handled by the OS in that when storage is needed, the amount of space is calculated and allocated. The information to be saved is then moved to that location. On mechanical drives this is done in sectors and wherever there is space on the drive closest to the center, which is done to minimize access times. On newer SSD drives, the information can be stored together and shifted around together to keep files contiguous. This eliminates the need for the system to have long searches for data.
4. **Memory Management**: Windows memory management takes care of tracking, moderating and controlling memory within the system. It monitors and calculates out how much memory will be needed and figures out if it is available. Based on priority levels of the program, it can then allocate or take back memory if needed for other, higher priority applications. It also keeps track of what is no longer using memory and reallocates that to where it is needed most.
5. **Distributed Systems and Networks**: Being that this game will be multi-player and the players will all be interacting with each other and a central computation system, having a server setup to handle the games would be the route to take. The server can host any image libraries, chat services, game save information and anything else that is needed to run the game. From here the players can all connect to a central system, which would be handling data routing, game computations and a central time system for monitoring game timers.
6. **Security**: Part of the security for a game like this would come in the form of protecting the user’s personal info such as name, address, etc. This can be done by keeping only the necessary information for the players, keeping it behind a firewall and keeping it encrypted. The other part of the security comes in the form of monitoring for cheaters. Keeping data logs of times, matching times against a central clock, masking user IP addresses and actually enforcing bans for cheating are some ways in which to keep the users happy, secure and wanting to come back to the game.