

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 11/17/24 | Chase Knoblock | Recommendations for The Gaming Room Software  Updated classes and added Entity Class |
| 1.1 | 11/30/24 | Chase Knoblock | Updated Evaluation and Recommendation sections.  Made changes based on newfound/learned information |
| 1.2 | 12/14/24 | Chase Knoblock | Updated Recommendation section, providing more information where necessary |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has an application currently available on Android called Draw It or Lose It. They would like to develop a web-based game that can be used on multiple platforms such as Windows, Mac, Linux, and IOS, that is based on the same game. For the application to work, each game and team name must be unique. Each game must allow one or more teams to play, and each team must allow multiple players. Only one instance of the game should exist in memory at any given time.

## [Design Constraints](#_2et92p0)

Android, IOS, and the web are all different, so cross-compatibility will be needed as the application is intended to run on any system or browser.

Safe coding practices and security should be prioritized. Using the RESTful approach would be good regarding user login and authentication.

The API should allow one or more teams from any platform.

Each game and team name should be unique.

There should be an alert that a team name is already in use and be prompted to choose another.

Unique IDs should exist for each instance of a game, team, and player, limiting each instance to one.

Language being used will be Java.

## [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)

The ProgramDriver class contains the main method and creates a Singleton instance of the game in memory. The SingletonTester class, in conjunction with the ProgramDriver class, is used to show/prove that only one instance of the game does exist, using the Singleton patter to create the static instance of GameService. The GameService class uses a public getInstance method to get the singleton private instance, using encapsulation to prevent changes to instance data.

Polymorphism is used by the GameService because of the use of the getGame function, as it checks method signatures and invokes the correct method. The addGame method in GameService utilizes an iterator pattern, allowing access to an encapsulated Game instance without knowing the list object data type.

GameService can have anywhere between zero to many Game child classes, demonstrating composition as it uses a List<Game> type static variable. Every team and player will have a unique name, created by the addPlayer and addTeam methods using an iterator pattern. The Entity class will hold attributes and behaviors that the Player, Team, and Game classes inherit, demonstrating abstraction.

****

## [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 must 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** | Macs can be used as a server, although the licensing is expensive, and you must have Mac’s to develop.  Easily upgradable and they have a GUI OS that is simple and strong.  Many options for web hosting.  Very good security, specifically against malware programs. | Linux is well equipped for a web-based hosting situation. It is the most popular of these and is free as far as licensing goes.  Uses a very flexible terminal and is secure.  Less likely to be victim of cyber threats.  Very customizable as it is an open-source format and is popular for web hosting.  Not as compatible as Windows when it comes to games and there are not many Linux pre-built machines as there are Mac and Windows. | The OS is very straightforward and there is a ton of support regarding software.  Very compatible.  Susceptible to viruses due to the many updates.  Windows can be expensive as licenses are not cheap up front but is less than Apple’s. | Mobile phones are used by nearly everyone and are very user-friendly.  Less likely to attacked by viruses and malware.  Hardware cannot be upgraded and there isn’t much than can be done if a virus is contracted. |
| **Client Side** | Expertise and time will be required. Integration between different Apple devices is very easy and straightforward. Updates are done automatically, and less frequently. Setup cost is expensive, and some software is charged per month. Some hardware updates take a very long to be released. | A lot of expertise and time will be needed for Linux. Not many applications are available with Linux OSs, so Linux can be difficult to setup. Likely to be the most secure, behind Apple.  Uses open-source format so it is cheaper. Lacks in debugging causing it to be more prone to debugging and lacks technical support. | There is a very large selection of Window based PC and there is a wide price range for them. Many applications are compatible and makes it easier to run games.  Forced updates take up time and are inconvenient. There is some inconsistency in functionality and security could be an issue. | Cost-effective and is very user-friendly.  OS runs smoothly and is very fast, using flexibility.  Could cost a bit more to develop applications for mobile devices depending on the device and updates.  Android and Apple use their own languages so expertise in specific mobile OSs will be needed. |
| **Development Tools** | Languages that can be used are HTML, CSS, and JavaScript. Terminal is very useful on Mac’s compared to Windows. IDE’s that can be used are Java, Python, PHP, and Ruby. Mac OS has some tools such as PyCharm, Eclipse, Visual Studios, Notepad++, and some other online tools. | Linux can utilize almost all of the same IDEs as Mac but they are unsupported. Python is also popularly used on Linux systems. | Windows is easier to use than Linux and can run the same IDE’s. Windows and Linux can be run at the same time. Very many tools and resources for aid in Windows. | Android and Swift would be most helpful in creating software. The software would run on any of the other machines. |

## Recommendations

1. **Operating Platform**: I highly recommend using a Windows OS, considering the necessary expertise and the availability of software. Windows is great for gaming and the cost is lower. There are many IDEs that can be used to develop.
2. **Operating Systems Architectures**: Windows is layered and consists of a user and “kernel” mode. The “kernal” will manage memory allocation among other things while the layer above it, “Executive Services” will service the system. There is also a “Hardware Abstraction Layer” that will take the isolates the OS from the hardware so that the system is compatible with multiple machines. There are services used by all Windows applications, such as file management and the GUI.
3. **Storage Management**: Cloud storage can be utilized as it is a relatively small file size and there exists an ability to modify the image library quickly without extra physical hardware. Windows server OS allows for memory management. Cloud storage is also accessed quickly with fast load times.
4. **Memory Management**: Windows OS has its own physical and virtual address spaces where memory can be addressed. A 64-bit architecture will give additional virtual address space, and this will allow for more image files to be accessed quickly. This should not affect the program’s performance.
5. **Distributed Systems and Networks**: A cross-platform development tool would be good to cover all platforms, no matter what OS is being used. Some tools like Unity or Unreal Engines would be good options. Unity is cost-friendly and supports Windows, Linux, IOS, and Android. A server that is dedicated to gaming would be beneficial to high traffic and the application would be given more bandwidth, but at a much cheaper cost than some other options.
6. **Security**: Windows server OS does allow for user settings to be modified regarding security and data that both enters and exits the system. The OS uses memory allocation and virtual addresses that contain private processes that are unable to be accessed by other processes, unless those processes are shared. Using the VPN service would protect the client’s data and accounts. Windows also uses an anti-spyware program to fight malware and viruses from the client’s system. McAfee and Norton are also great options for extra security to be used on Windows.