

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/22/2023 | Matt Smith | Updated Executive Summary, Design Constraints, and Domain Model. |
| 1.1 | 02/03/2023 | Matt Smith | Updated Evaluation. |
| 1.2 | 02/13/2023 | Matt Smith | Updated 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)

The Gaming Room wants to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose it. It’s currently only available on the Android App store so it will need to able to run the same on different operating systems as it does the current app.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

The first constraint is that the game must have the ability to have one or more teams involved. This will require implementing an ability to add teams to a game.

The second constraint is that the game must have the ability to assign multiple players to a team. This will require implementing an ability to add players to specific teams.

The third constraint is that the game must have unique game names and team names. This will require implementing code that checks to see if the current game or team names are in use.

The final constraint is that the game can only have one instance exist in memory at a given time. This will require assigning identifiers to game, team, or player instances.

## [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 is what initializes the program when the program is opened and it used the SingletonTester class to test for a single instance. The classes GameService, Game, Team, and Player are all associated with each other. These classes all use encapsulation by making certain attributes and methods private. The Game, Team, and Player class also inherit from the Entity class. This shows inheritance from the arrow connecting them to the Entity class. With this, they can use methods from the Entity class within their own class.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [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** | macOS is a Unix operating system that runs on all Apple computers. It can be installed on any computer if it meets the minimum hardware requirements.  An advantage is that it’s perhaps the most user-friendly operating system compared to others. This would make setting up as macOS server easier compared to a Linux based system.  A weakness of macOS is that it’s a closed system so some things you might want to tweak may be impossible. macOS also needs Intel based CPUs so you can’t save as much money if you wanted to custom build the server. If you buy a Mac, there is limited to no hardware customization so you would have to ensure the machine is “future proofed” for your needs. | Linux is a free, open-source operating system that can be installed on any computer, even computers with old and outdated hardware.  An advantage is that it’s compatible with frequently used software and hardware. Since Linux is open source, it’s easier to customize to suit specific needs.  One weakness would be the level of expertise required to configure a Linux based server. Another would be that Windows applications aren’t compatible with Linux so if any Windows based software is needed, it will be a major problem. | Windows is a closed-source operating system that can be installed on any computer as long as the computer meets the minimum requirements for the specific OS.  An advantage is that it’s a user-friendly operating system. It supports third party applications and system issues can be resolved with system restore if needed.  A weakness is that it’s vulnerable to more malware and viruses. It also is resource intensive and can have issues because of user error. It also has high licensing costs. | Mobile devices can be carried anywhere, making it convenient to bring the server anywhere. They all have touch screens which makes it easier for users to interact with.  An advantage is that you could bring the server with you instead of going to it, in case of an event where you’d have to be physically in front of it.  Weaknesses would include the limited processing power of mobile devices compared to other systems. They also have smaller screens which can make it hard to interact with. It also relies on just a battery so it either needs to be plugged in constantly or watched when on the go. |
| **Client Side** | For multiple types of clients, it could require more investment in hardware, software, and licensing fees which will increase the overall cost.  It will be time consuming to develop software for multiple types of clients because it must be optimized for each type of client. This will require more effort to ensure compatibility.  It will require a team with a high level of expertise of macOS. It may require hiring developers that are specialized. | For multiple types of clients, it’s more cost effective considering Linux is open-sourced. Costs could increase if specialized applications or libraries are needed.  It will be time consuming to develop software for multiple types of clients because it must be optimized for each type of client.  It will require a team that understands different Linux distros. The team also needs knowledge in developing software for Linux. | For multiple types of clients, it could require more investment in hardware, software, and licensing fees. It may all require other tools or libraries that will increase the overall cost of development.  It will be time consuming to develop software for multiple types of clients because it must be optimized for each type of client. The team will also need to spend more time ensuring it’s compatible with different version of windows.  It will require a team that understands different versions of Windows. It also requires a team with knowledge of developing software for Windows. | For multiple types of clients, it will require multiple versions of the software to support different systems. It will require investing mobile device to test and debug, which will drive up costs.  It will be time consuming to develop software for multiple types of clients because each version of software must be optimized for each mobile system. This also requires a team or teams to spend more time ensuring its compatibility with each type of mobile device.  It will require a team with a high level of expertise. The team will need to understand mobile-specific technologies and have knowledge of developing software for multiple mobile systems. |
| **Development Tools** | The programming languages that are relevant for macOS are Swift, Objective-C, C++, Python, and JavaScript.  The relevant IDE is Xcode because it’s from Apple. Visual Studio Code can also be used as well.  Other tools include Homebrew, CocoaPods, and Git. Homebrew is a package manager, CocoaPods is a dependency manager, and Git is a version control system. | The programing languages that are relevant for Linux are C, C++, Python, Java, and Ruby.  The relevant IDEs are Eclipse, Visual Studio Code, IntelliJ IDEA, and PyCharm.  Other tools include make, GCC, and Git.  Make is a tool that automates building and deploying software. GCC is used to compile multiple programming languages. Git is a version control system. | The programming languages that are relevant for Windows are C#, .NET, Java, Python, and C++.  The relevant IDEs are Visual Studio, Eclipse, IntelliJ IDEA, and PyCharm.  Other tools include MSBuild, Nuget, and Git.  MSBuild is a tool that automates building and deploying software. Nuget is a package manager for .NET. Git is a version control system. | The programming language that is relevant for Android is Java. For iOS, the language is Swift. For both Android and iOS, it would be React Native.  The relevant IDE for Android devices is Android Studio, for iOS it’s Xcode, and for React Native it’s Visual Studio Code.  Other tools include Gradle, CocoaPods, and Git. Gradle is for Android, it automates building and deploying software. CocoaPods is for iOS, it’s a dependency manager. Git is a version control system. |

## 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**: I would recommend a Linux server to allow the Gaming Room to expand Draw It or Lose it to other computer environments. Linux is a stable, open-source, and is a popular choice for servers. It also has support for multiple programming languages which makes it easier to integrate with various software components.
2. **Operating Systems Architectures**: Linux uses the Unix-based architecture. It has a hierarchical file system and it supports multitasking, which allows multiple processes to run simultaneously on the server. It’s also highly configurable and can be customized to meet the specific needs of the client.
3. **Storage Management**: A Network-Attached Storage system would be the most appropriate storage management system for Linux. It allows the client to store data on a central server and access it from any device on the network, which makes it the ideal solution.
4. **Memory Management**: Linux uses a virtual memory management technique to manage memory. Virtual memory lets the operating system temporarily transfer data from RAM to disk storage when the physical memory is full. This ensures that the software continues to run smoothly even if it’s under a heavy load.
5. **Distributed Systems and Networks**: A distributed system that can be used is a Microservices architecture. This splits the software into smaller, independent components that communicate through APIs. The network that connects the devices should be designed to support high availability and fault tolerance to minimize the impact of outages or connectivity issues.
6. **Security**: To protect user information, I would recommend encryption for all sensitive information. I’d also recommend implementing user authentication and authorization through a secure token-based system such as OAuth. The recommended cloud-based platform also provides various security features such as firewalls, intrusion detection, and data encryption. This ensures that the game and its users are protected.