

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 | 04/17/22 | Jessica McAlum | Introducing Executive Summary, design constraints, domain model, evaluations and recommendations of operating systems. |

**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 game, Draw It or Lose it, is currently only available in an Android app. The client would like to develop a version of this game that is web-based and serves multiple platforms. The client requires the game to be able to have one or more teams involved with multiple players assigned to each team. The software must ensure that each game and team name is unique and allow users to check whether a team name is already in use when choosing a team name. The software must also ensure that only one instance of the game exists in memory at any given time. A solution for ensuring that each game and team name is unique, and only one instance of the game exists in memory at any given time, is by using a singleton design pattern when developing the game.

## [Design Constraints](#_2et92p0)

The game must be web-based and able to run on multiple platforms. This means that we must develop software that is able to work on all web browsers.

## [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 concept of inheritance is illustrated in this UML class diagram by showing that the Game class, Team class, and Player class are to inherit all the attributes and methods from the Entity class. The use of inheritance will allow for the program to be easier to build upon and minimize repetitive code. Encapsulation is also illustrated in this UML class diagram with plus and minus signs that represent which attributes and methods can be accessed by other classes. The lines between the GameService class, Game class, Team class, and Player class represents an association relationship between those classes and there is no dependency between them. The multiplicity values between those classes also mean that there may be zero to many instances.

**"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** | Mac would be a very secure option for hosting a web-based software application. The hardware and software for Mac hosting is very reliable. However, Mac hosting and licensing costs may be pricey. | One advantage of Linux for hosting a web-based software application is that it’s the most cost friendly. Another advantage is that Linux is open source. However, the learning curve is steep when it comes to Linux, so more expertise is required to operate. | Windows is a very strong choice in terms of supporting gaming. Windows would also be a strong choice in terms of usability and compatibility. Windows devices and licensing may become pricey. | An advantage would be the ability to have real-time updates. A weakness would be that the web-based software application would need to be downloaded to access it each time. |
| **Client Side** | Mac products would be an expensive option. However, they are reliable and relatively easy to use. | Linux is easy and free to install. It’s cost friendly because it doesn’t require higher end hardware. However, Linux is a relatively more difficult operating system to use and requires more expertise. | Windows may be costly when it comes to hardware and licensing. Windows is user-friendly and provides the most customization choices. | Mobile devices are portable, which makes them a convenient option. However, service connection might be a factor to consider. Mobile devices are user friendly. |
| **Development Tools** | Xcode is a popular IDE that allows programming in Java. They provide a multitude of development tools for developing software for deploying on Mac. | Visual Studio Code is an IDE that supports all the popular languages (C++, Java, Python, etc.). | Visual Studio IDE is available for Windows and supports web, mobile, and app game development. They also support C++, Python, ASP.NET, Node.js, and Unity. | Xcode is an IDE that allows building apps for Apple mobile devices and supports Swift. Android Studio is an IDE for Android that only supports Java as a programming language. |

## 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**: Windows would be the most appropriate operating platform that would allow The Gaming Room to expand Draw It or Lose It to other computing environments. This would be the more user-friendly and compatible option. It’s not as complicated as Linux to set up. There’s a lot of tech support that comes with Windows, and Windows 11 offers much more security support than previous versions.
2. **Operating Systems Architectures**: Windows processors switch between a user mode and a kernel mode. User mode allows for applications to run in isolation, so crashes are limited to only the application and does not affect the operating system.
3. **Storage Management**: Adding more storage to windows systems are relatively easy. Cloud storage is also an option. Windows includes a Storage menu with a Storage Sense feature that can help users to manage and optimize their disk storage space. It can clear up disk space by deleting unnecessary files or allow files to be made online-only files and saved in cloud storage. These storage management settings are completely optional and may be configured to the user’s liking.
4. **Memory Management**: Windows includes a virtual memory feature that uses the hard drive to essentially free up RAM. This will enhance overall performance and expands the systems’ data handling capabilities. To ensure that memory is effectively managed, must ensure that we have adequate RAM that would allow the application to render and display pictures at a fixed rapid rate.
5. **Distributed Systems and Networks**: Windows can run almost any software and is client/server operating system. An advantage of having a client/server network is that it can be accessed across multiple platforms. We would also be able to utilize cross-platform development tools, which would allow us to develop the game for use across multiple platforms.
6. **Security**: Windows comes with built in firewalls as well as many other great security tools like Windows Defender. They also offer Windows Sandbox which allows users to run applications in isolation to protect the system or platform from malicious software.