

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | <11/03/2023> | Charles Matthews | This is the first edition. |

**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)

## Requirements

The Gaming Room, hereby referred to as the Client, is looking to implement their game “Draw It or Lose It” such that the architecture recognizes individual players, teams, and games. The system would need to also identify players by a username and restrict other players from using identical usernames. Additionally, the Client needs the game to be able to create multiple teams, usually one or more, with more than one user in a team.

## [Design Constraints](#_2et92p0)

The game code will need an iterator to identify individual games, teams, and even players running on the server and reject duplicates of each should there be conflicting IDs. It will also need to identify the user’s operating system in order to accurately produce the images required for the game to run, and allow for each system’s graphics to create the image for each round of each individual game, and likewise allow for each player to be able to submit the solution to each round’s clue

## [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 program model displayed below shows that many of the items that are required for the program to work properly. The SingletonTester, which the ProgramDriver uses, is what tests to ensure that each Game, Team, and Player have their own unique ID, identified by each class’ ID variable. Each Game, Team, and Player item with their own unique ID are all a piece of the GameService, which uses the singleton iterator found in SingletonTester to ensure each ID is unique to each Game, Team, and Player. The GameService will help to assign Players to Teams, and those Teams to the appropriate Game. This is then fed into the Entity class, where the game starts to play out in accordance to the Client’s rules. The GameService, Team, Game, and Player classes are all associated with each other, while the entire Entity inherits elements from each of the Team, Game, and Player classes, and the ProgramDriver is used by the SingletonTester.

**"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** | Unix based Mac has a direct compatibility with many Linux systems, is very user friendly, considered highly secure, and hosts a smooth performance between hardware and software, with additional use of developers tools and a wealth of multimedia capabilities. The issues arise with the lack of ease when attempting to upgrade, not as cost effective or as much of a knowledge base with hosting. | Linux is Open Source, has a more robust security model and permission system, flexibility, and, with a large community, has a wealth of resources to support the performance and stability of the program. However, this comes at the expense of requiring a decent learning curve, occasional hardware compatibility issues, and may not be fully compatable with other systems. | Windows is not as secure as other OS, nor as stable, and may require the use of a command line, licensing costs, and may utilize too much of the available hardware resources. There is, however, numerous resources available, with lots of support for user management, including remote desktop services and a .NET framework, making Windows Easier to use. | While mobile and easy to access at all times, with touch interactions and a slew of features that can be used for programs, Mobile Devices require access to a network pretty much nonstop, and have limited resources, including power. Security is also a concern, as well as trying to scale-up, as using multiple devices would be impractical, and the fact that there are limitation and compatibility differences between each mobile operating system. |
| **Client Side** | Constant support is required to ensure that everything adheres to Apple’s guidelines, both with user interface and within the Apple store itself, while also being compatible with the macOS versions and their respective security systems, and properly scaling to the appropriate size on the various models. This would require some experienced developers and adequate time to update as the operating system and security updates as well, while requiring the proper licensing. | To work with Linux, the program will have to know which distribution it will be running in. The development will have to work with the various hardware limitations on various machines, and a preference towards being open-source. Creating such a program will take time, as it will also need able to utilize command line information as well as standard GUI outputs, and also adhere to the proper security information requiring an expert hand in the development of the program. | When using Windows, one must take into account the various hardware constraints each system has, in addition to the actual operating system version the computer is running. The program would need to be able to work with the accessibility framework, and be heavily security minded as threats are much more prevalent in the Windows systems than many other operating systems. It would take a lot of time and effort to ensure the security of the program and its users. | In order to run on a Mobile Device, the product must be able to work cross-platform, while also being mindful of the connectivity, software limitations, and hardware limitations. Optimizing things like battery life and the memory system are prudent, as would ensuring that everything follows proper guidelines in their appropriate downloading application and the various security structures associated with the platforms. It would require developers time and expertise in the major mobile operating systems to get each product to run smoothly. |
| **Development Tools** | Some useful tools include Xcode, which supports the Apple-exclusive Swift language, as well as Javascript, Python, Java, and C/C++ coding IDE of Visual Studio Code, IntelliJ IDEA, and Eclipse. Another useful program language to use is Ruby, which is useful for web development. | Useful tools found in Linux include Visual Studio Code, Eclipse, Atom, and Geany. These IDE can support useful languages like Python, Java, C++, PHP and Ruby, many of which are useful in the environment for the type of development necessary. | Window uses many IDE tools, including Visual Studio/Code, Eclipse, and PyCharm to utilize every language from Python and C++, to HTML and Javascript in creating the required code structure. | Depending on the system in use, one could use Java with Android Studio, or Swift with Xcode, but the two big platforms can use Visual Studio for C# or PhoneGap/Cordova to work with HTML or Javascript to build with web technologies. |

## 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**: After much consideration, the recommended operating platform for the Client should be a Linux system.
2. **Operating Systems Architectures**: Linux has a wide variety of platforms within itself, like CentOS, Fedora, Ubuntu, and the like. Everything is contained in a monolithic kernel architecture, and follows a standard system file hierarchy. It incorporates various security features, package management systems, comprehensive network support, and a solid file support system.
3. **Storage Management**: To properly manage the storage of the program and its code, Linux can swap files to make room for the files needed to operate the software, which plays into the standard file system hierarchy that drives the system under operation.
4. **Memory Management**: Linux uses a memory mapping algorithm to properly get processes to share memory, and also utilizes features to reallocate memory from unnecessary processes to the process that requires it, making it easier to ensure that the software doesn’t throw a code due to insufficient memory.
5. **Distributed Systems and Networks**: Due to a robust networking capability, its nature as being open source, and the ability for scalability, Linux systems provide the best available environment to create and communicate to users hoping to use the Draw It or Lose it software.
6. **Security**: User information can be protected using Pluggable Authentication Modules to allow flexible authentication mechanisms, as well as two-factor authentication, as the need arises. Packets are also encrypted in transit, as well as a powerful firewall for added protections. This allows the data pertaining to user information to be secure, and Linux systems audit systems within the OS to determine any unusual activity.