

<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 | 05/20/23 | Nicholle Caudy | Recommendations for The Gaming Room software include updating existing classes and adding the Entity class. |

**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 client The Gaming Room is looking to expand their game Draw It or Lose It. The current version of their game is only an Android application. Currently, the staff at The Gaming Room does not know how to set up the environment and The Gaming Room needs help developing a web-based version of their gaming application. The web-based version needs to have the ability to have one or more teams involved and can have multiple players on each team. The game and teams must have unique names with the ability to check whether the names already exist. Using the iterator design pattern will solve this problem because it grants access to an object’s members without exposing the encapsulated data structures. Only one instance of the game can exist in the memory at any given time. The best solution in this case is to implement the singleton design pattern, this will allow the unique identifiers to be allocated properly and allow for only one instance of the game to run at any given time.

## Requirements

Design a web-based gaming application for the client The Gaming Room that can have one or more teams involved. The game and team names must be unique to allow the user to check whether a name already exists (iterator design pattern). Another requirement for this web-based gaming application is that only one instance of the game can exist in memory at any given time (singleton design pattern)

## [Design Constraints](#_2et92p0)

There are some design constraints with this web-based gaming application that include creating a game that runs on multiple platforms, The Gaming Room’s Draw It or Lose It game is only available for Android currently. This web-based game is a multi-user platform with the need to have unique names for each game and team that allows the user to check whether that name already exists, as the game grows this may cause an overloading issue. Another constraint is there should only be one instance of the game at a time, this may slow down the servers.

## [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 UML class diagram has 7 classes ProgramDriver, SingletonTester, Entity, GameService, Game, Team, and Player. The ProgramDriver class, which contains our main() method, uses the SingletonTester class to test for a single occurrence of the game running at a time. The UML diagram represents inheritance between the GameService, Game, Team, Player, and Entity classes, we can see the cardinality that is expressed in terms of 0…\* which means zero to many. The SingletonTester is part of the abstraction that allows us to ensure the functionality of the program and that it is running correctly. The singleton design pattern also demonstrates the OOP feature of encapsulation, for this instance we are protecting the instance itself and treating it as an individual to run the program efficiently without any other instance interacting or interfering with another. The combination of different itemized instances such as the different teams, games, and player ids demonstrates polymorphism because they are an object that is stored in a specific place that will be assigned to these players.

**"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** | Some advantages of the Mac OS are the powerful GUI OS and its good use of CLI. It has flexible terminal commands to configure the server, access, or make changes, and is very secure. Mac has great anti-malware programs that protect against spyware, worms, and adware. Some disadvantages are that Mac is not open source and can be expensive. It has limited hardware upgrade options. | The advantages of Linux are that it is open-sourced, and its stability. Linux has great security and is customizable. The disadvantages of Linux include compatibility issues along with limited software availability. There is also limited hardware support and a heavy learning curve. | Windows has many advantages that include the availability of user-friendly software programs. Windows provides great support through updates, patches, and version iterations. This helps with any security issues and improves software stability. It is also great for gaming and has an extensive library. Disadvantages of Windows include some security issues because it is a popular target. Windows consumes a large amount of system resources. It can also be expensive. | Some advantages of mobile devices include that they are mobile web servers. You develop apps for both websites and phones interchangeably. They are user-friendly and have a very large user base. Some of the disadvantages include the inability to upgrade hardware. There are times when they cannot connect to the internet. They may struggle to handle large amounts of traffic. |
| **Client Side** | It is easy to develop on Mac and great for Mac users. But knowledge of Mac is necessary. The systems can be expensive and aren’t known for gaming. Charges monthly for some software. | Linux is open-source and easy to use. Can customize distribution specifically for development. Linux needs expertise and due to the lack of applications available may be difficult to set up. Can be prone to bugs because of a lack of tech support. | There is a large selection of Windows-based PCs in a multitude of price ranges. Windows would make for easy server development and administration. Windows is not open source. There can be forced updates and can be time-consuming. | Mobile devices are very cost-effective and user-friendly. They provide flexibility and accessibility on the go. They don't have the amount of configuration as a computer. Different devices use different languages and the development can take longer and require more expertise and time to complete. |
| **Development Tools** | Languages consist of but are not limited to HTML, CSS, and Java. Mac is also better suited to do more from the terminal compared to Windows. IDEs can be Java, Python, PHO, and Ruby. Development tools within Mac OS can be Pycharm, eclipse visual studios, and online tools. Swift and Xcode are specifically made for Mac. May have to translate languages into Mac language for application development. | Linux has a large variety of development tools. Can run SeaMonkey, Quanta, and Sublime Text. Linux works with all the similar IDE used with Mac all the software is not supported. | Windows runs the same IDE as Linux and is easier to use. It is, however, possible to run both Windows and Linux at the same time. Windows has many tools and resources that can prove helpful. Not specifically made for usability when developing Visual Studio, Eclipse, C++, Java, Python, or website development. | Mobile devices have specific libraries for applications that have been proven to be easy and secure. Offers tools for growing your application business-wise. Android has Xamarin and Kotlin and Apple has new M1 chips. |

## 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 recommend the client The Gaming Room use the Windows OS. I make this recommendation because of the level of expertise required and the software available. Windows has a strong gaming platform, and the cost is relatively low. Windows OS also offers a large number of helpful IDEs that can make code writing and development faster and more efficient.
2. **Operating Systems Architectures**: Windows allows users to develop and set up the environment specifically for the needs of the client. It offers GUI (Graphical User Interface) that is easy to use and is well known. They also offer great technical support as well as large amounts of documentation online to help solve any issues that may arise. We can use the OS to split up the computer resources easily to run dual operating systems, shall this be needed for test cases of applications on a different OS line Mac, and Linux.
3. **Storage Management**: Windows OS offers many different options to use that include OneDrive and Dropbox. We also have the option to use company-coded databases in addition to SQL.
4. **Memory Management**: Windows allows the ability to specify the amount of memory that you use for applications within a designated virtual instance during runtime. Windows has a physical address space and a virtual address space that enables the addressing of memory.
5. **Distributed Systems and Networks**: The Cloud technology would allow us to run servers with the company databases that would allow users to call the application on whatever platform they happen to be using. The ability to configure the type of requests and respond with different instances of the game with multiple types of virtual instances dependent on the OS is incomparable. This is because they communicate easily and efficiently, especially when it comes to company infrastructure and local networks. Unity would also be a good option because it is cost-effective and can support multiple platforms like Windows, Linux, ios, and Android.
6. **Security**: Using Windows allows us to create personalized databases, and infrastructure to implement firewalls, and encryption. Windows offers great support on all accounts. Windows OS also uses anti-spyware as a built-in feature to help keep malware software or viruses from getting into the system. Other anti-virus programs can be purchased to help protect the system.