

The Gaming Room

# **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 | 09/16/2023 | John Schmidt | Revised application to use the singleton pattern so only one instance of the GameService class can exist at any given point. Created single Entity class that is inherited by game, team, and player classes to reduce repeated code. |

**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 Company wishes to make a web-based version of their gaming application. To do this, software will have to be created that allows the game to function as intended, while being hosted on an online server. We will need to create a product that runs the game seamlessly online with multiple teams and players being able to play at once. We will also need to determine which operating system best suits the development of this web-based game.

## Requirements

The software requirements for the game application are:

* The game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

The design of this application has a few constraints. First off, we must create an application that allows multiple unique teams to use the service at one time. These teams need to be able to be identified based on a unique name that isn’t allowed to be used by a different team. Each team will also need to be able to have multiple players assigned to it. We must also ensure that only one instance of the game may exist in memory at any given point. To do this, we will have to create unique identifiers for each instance of a game, team, or player. Finally, we have to complete this project within the company’s budget, and within a timeframe agreed upon by both parties.

## [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)

Looking at the UML diagram below, we can identify a number of relationships. The first one to point out is that the classes game, team, and player all inherit from the super class, or parent class, entity. This means that all of the attributes from entity are carried on through to these classes as well. We can also see that the game service, game, team, and player classes are all associated with each other. We can go a step further and see that game service may contain 0 to many games, games may contain 0 to many teams, and teams may contain 0 to many players. This just highlights a bit more about the relationships between these classes. We can see the OOP principle of encapsulation through the use of private member functions and attributes in the different classes to make sure that data within these classes isn’t accessed accidentally by outside code. We also see inheritance through the use of the parent class Entity, and all of the classes that extend it.

**"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** | The Unix-based mac operating system is known to be developer friendly, allow easy integration with popular technologies, and have high graphics and multimedia capabilities. Its downsides are that it can cost more than other operating systems, it has limited hardware options, and might not be the best for scaling large projects. | Linux has several strengths when it comes to software development. Some of them include being open source and available for free, being known for reliability, having high security, and being able to scale to accommodate growing demands. Some of Linux’s weaknesses include a high learning curve, hardware compatibility options, and concerns about server optimization. | Windows has many advantages when it comes to hosting a web-based software application. For starters, Windows OS has a large amount for SQL servers and databases. Windows also has a user-friendly interface that makes hosting simple. It also has well put together integration with Azure Cloud, which makes hosting easier. Its weaknesses include being more costly than open-source options, as well as higher server requirements in terms of memory and CPU usage. | Since mobile smart phones are common, any program developed for them can reach a wide audience. The user experience with smart phones is also usually easy to use. Some weaknesses include uncertain connectivity because smart phones must function in a wide variety of areas. Smartphones also have the issue of constant security concerns. |
| **Client Side** | Creating a program for macOS may require a higher cost and more time. It would require sufficient testing on macOS devices, which can cost more, as well as an in-depth knowledge of how programs function on macOS. These factor into being able to make a program that is desirable to mac clients. | Being open source, many upfront costs are avoided with Linux. However, there would have to be quite a bit of testing done to make sure that it is compatible with the future client’s hardware. Also, due to Linux being quite different than some other operating systems, it may take more expertise or time to successfully develop a program that clients will enjoy. | Windows OS development can require specific tools, adding to upfront cost. This combined with licensing costs for servers can make the OS a more expensive option. Also, making the program run well on multiple Windows versions may also increase the time needed. Most people are familiar with the Windows platform and wouldn’t need much additional training with it. | Using the tools to make the application cross-platform may increase development costs. Also, third-party services are often needed for apps to run successfully, which may lead to a higher cost as well. It will also take people that are experienced with mobile operating systems to be able to create a application that is optimized for smartphone across all different devices. |
| **Development Tools** | The primary programming language for macOS is swift. Another popular option would be objective-C. Xcode is the main IDE used for Mac application development. It supports both languages mentioned above. A popular framework to use for macOS is Cocoa and Cocoa Tough, both of which are used as application programming interfaces. | Many languages are relevant for Linux, including C, C++, Python, Java, and Rust. Also, many popular IDEs are used in Linux including Visual Studio, Eclipse, PyCharm, and Atom. This makes developing in Linux rather diverse because it supports many different options. Qt is a popular framework that uses C++ and enables well-made graphical interfaces for Linux users. | Windows uses a variety of common programming languages, including C#, .NET, C++, and JavaScript. Some tools and frameworks include the .NET framework, which allows a large variety of languages to be used for developing Windows applications, as well as Universal Windows Platform, which allows building apps that run across different Windows devices. Some common IDEs include Visual Studio and JetBrains Rider. | Android and apple vary in their common programming languages. iOS devices utilize Swift often, while android utilizes Kotlin and Java frequently. Dart is a common language used for cross-platform development. Flutter is a popular toolkit for building an application for cross-platform use using the same code. Xcode is the main IDE for iOS app development, while Android Studio is a popular choice for application development on Androids. |

## 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 using Windows for the development of this application. Windows is the most common operating system used every day, and comes with a wide variety of different IDEs, programming languages, and tools to help us develop this software.
2. **Operating Systems Architectures**: The Windows OS architecture is a well-developed model that is continually improved upon over time. The Windows OS has a hardware abstraction layer which allows the kernel to isolate from differences in the hardware, allowing Windows to run well on very different hardware configurations. This allows Windows to be used very widely, as it runs well on most machines. The Win32 system provides the graphical user interface and allows user applications to work with the operating system.
3. **Storage Management**: Based on the desired function of this game, I would recommend the new technology file system (NTFS). This is the most used and accessible file system in Windows. It is highly efficient and reliable, and without the need for massive amounts of data or classified information, it should be more than enough for this software.
4. **Memory Management**: Windows segments memory for different purposes which helps to organize it and helps applications run smoothly. Windows also does not allow one process to access another process’s memory, which helps increase security and stability. Windows also has many methods to optimize memory usage so that its applications run well.
5. **Distributed Systems and Networks**: The best way to have Draw It or Lose It accessible across many different systems and platforms is to design the software in frameworks that support and facilitate compatibility between different platforms. Some of these frameworks include React, Native, and Flutter. These frameworks will help us create an application that can work across many platforms.
6. **Security**: There are a few different methods to increase the security of the application. One of Window’s secure storage platforms is known as Windows Credential Locker. This is a platform-specific mechanism that helps to ensure secure user data. The software will also use HTTPS guidelines to increase secure data transmission. Another thing that the company can do is receive additional training to increase their overall knowledge of software security and how to avoid or detect possible breaches of security.