

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 11/13/2022 | Christopher Sharp | Added information to requested sections |
| 1.1 | 11/24/2022 | Christopher Sharp | Filling in Evaluation section |
| 1.2 | 12/7/2022 | Christopher Sharp | Adding 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, our client, wants to develop a web-based version of their popular Android title, Draw It or Lose it. This title must be able to perform on multiple platforms to allow a broad spectrum of players.

## [Design Constraints](#_2et92p0)

* The game will consist of one or more teams
* Each team will have multiple members
* Platform independent
* Team/Game names must be unique
* Only one instance of the game service can be run at a single time. Checking against unique identifiers.

## [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)

First, the Entity class serves as an Abstract class in which Game, Team, and Player inherit information from to cut down on redundant code being present. Next, the GameService, Game, Team, and Player classes all share an association with each other, which means they all ‘know’ of each other, allowing information to be used across the classes. Also present in the associations are the numbers that represent that zero to many instances can be present with unique identifiers. Finally, you have the ProgramDriver class and the SingletonTester class that share an association. In this case, the ProgramDriver uses the SingletonTester to test the application for single instances of the game service.

**"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** | -Newer Macs have the software built in to be ran as a remote server.  -Unix based like Linux  -Not easily scalable  -Cost prohibitive in that the server must be ran from Mac Hardware  -Lower security threshold | -Industry leader in server software  -Highly configurable  -Easily scalable  -Very high security standards  -Unix based  -Disadvantage in learning curve to run system | -Offers server software  -Is used in the Industry when certain frameworks need to be ran  -disadvantage is the cost of the licensing and any hidden cost in expansion | -It is very cost prohibitive due to different requirements of the systems involved. |
| **Client Side** | -must train in the intricacies of a UNIX based system  -Cost prohibitive in the form of hardware | -most complex in application and training  -inexpensive to setup and run | -Easiest to learn and get a handle on due to familiarity  -cost relative to equipment and software | -great flexibility  -complexity and time based upon languages used for each system |
| **Development Tools** | -IDE: XCode, Eclipse, Visual Studio Code, most of the IntelliJ IDEs  -Languages: Swift, Java, C++, and Python, JavaScript, CSS, HTML | -IDE: Visual Studio Code, Most IntelliJ IDEs  Languages: Python, C++, Rust, and JavaScript | -IDE: Eclipse, Visual Studio Code, most of the IntelliJ IDEs  -Languages: Java, C++, and Python, JavaScript, CSS, HTML | Android:  Mostly using Android Studio  Written in Kotlin, Java, C++  iOS:  XCode with the Swift 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**: For operating platform, I would recommend the use of Windows. With the use of Windows, it will require the use of a paid license, but you have the familiarity of the Windows system that is currently deployed on many users’ systems.
2. **Operating Systems Architectures**: The Windows operating platform is used extensively throughout the enterprise sector and is used by many individuals throughout the world. The Windows architecture allows for the use of the of low-level processes without directly effecting the memory and processes that allow the system to operate. This protects the system from unauthorized access.
3. **Storage Management**: While the Windows Server software has built in storage management, such as Storage Spaces Direct, that allows for the clustering of internal storage, the recommendation that I am going with is to make use of an offsite storage solution in addition to the onsite server storage. This allows for the back up of the critical data while allowing for the future expansion.
4. **Memory Management**: Memory management is handled directly by the operating platform itself. Windows Server has Windows Virtual Memory Manager as a leading feature that maps the committed regions of address space to the available physical memory. This allows for the application to run on systems that may lack the amount of memory to run the application. Also, I recommend the use of databases to allow the application to quickly access the images.
5. **Distributed Systems and Networks**: For this application, we will use a client-server distribution. This will allow us to create a client application to run the game, while depending on a single instance of the server. The connectivity between the client and the server will be paramount for without the server application the client application will cease to function. For this, redundancy will need to be planned for in case of outage, whether electrical or data connectivity.
6. **Security**: The Windows Server software has many built in security protocols that will be utilized to protect the sensitive data of the userbase and that of the developer themselves. By leveraging these features will help to secure the application from unauthorized access. On top of this, the encryption of the data will be at the forefront of this security. Also, most modern programming languages have built in processes that help with the protection of the data to help deter the hijacking of the application for nefarious means.