

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.1

## 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 | 05/22/2022 | Luke Lauck | Wrote the initial paper |
| 1.1 | 06/12/2022 | Luke Lauck | Edited Evaluation |
| 1.2 | 06/19/2022 | Luke Lauck | Edited Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room needs to set up website that allows their game “Draw It or Lose It” to be played. We have created a UML and an evaluation on what operating platforms to provide service to.

## [Design Constraints](#_2et92p0)

A game will have the ability to have one or more teams involved. Teams will need unique id’s to not overlap one another.

Each team will have multiple players assigned to it. The client will have to send signals to ensure they are still playing the game.

Game and team names must be unique to allow users to check whether a name is in use when choosing a team name. Users cannot have the same name so searching names on the server is required to confirm a username.

Only one instance of the game can exist in memory at any given time. This will have to be ran server sided with unique ids to ensure the same game isn’t running twice.

## [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 ProgramDriver and SingletonTester are used for testing. The GameService uses a singleton pattern which contains a Game list. A Game has a list of Team, and each Team has a list of Player. Game, Team, and Player all inherit an Entity for a unique id/name.

**"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 does not have much server-side support and all the server equipment must be purchased by Apple which can get expensive. | Linux is a minimum running operating platform that is all about performance. It also comes with strong security with user’s acting as domains on the server. | Windows is known for viruses and shutting down to update. It does have a lot of support for server hosting, but not at a large scale. | Mobile Devices are terrible for hosting servers if they are “Mobile” since they would not be tethered to the internet. |
| **Client Side** | Mac has a lot of tools meant for only Mac and can be implemented but takes time. It is a lot better if the tools being used has Mac compatibility. | The client side of Linux isn’t well provided and may take a lot of time to get everything working. | Windows is where the computer audience is at. As the go to operating system, most tools have windows compatibility. | Android runs on the Linux kernel and iOS runs on the Mac kernel. A lot of time and the right tools need to be applied to get it working on all mobile devices. |
| **Development Tools** | The IDEs for Mac are stylish like AppCode and CLion. There is not many variants and they are costly. | Open-source programs like VIM and EMACS are free to use but can be difficult to learn. It requires employees’ expertise. | Most IDEs are for Windows with Visual Studio probably being the biggest one. Everyone knows how to use Windows and development tools for paid programs isn’t a lot since I have seen a lot of business licenses for a discount. | It depends on the team choosing a specific mobile device and the same tools since different mobile devices have different tools. |

## 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 11 is the newest of the Windows series and is used by most of the computer market. Its popularity grants it Visual Studio, an IDE that most programmers are familiar with making it easy to find new employees skilled in this field. Draw It or Lose It is aiming for the web-based game market, which is dominated by Chromium, a software usually built on Windows.

**Operating Systems Architectures**: Windows NT is the kernel that runs Windows 11. It is a paid kernel which runs the hardware in a way that the operating platform can work with. Instead of old monolithic designs, Windows NT runs on two main components, the user mode and kernel mode, which allows for multitasking while the operating platforms sends more processes to run.

1. **Storage Management**: Windows has a file system, storing data in a directory that uses folders in folders. By going from a root all the way down the branches, the so-called “Tree-Structured Directory” is the more common way to store information without security risks of cycling through the system. A way to set up the folders for Draw It or Lose It would have folders named for what comes beneath it such as: lib (third-part libraries), src (source code), img (images).
2. **Memory Management**: Windows manages its memory with a virtual address space. This space can store functions, heaps, and mapping to speed up read, write, and optimizations through loops.
3. **Distributed Systems and Networks**: Games can be hosted peer-to-peer, but as that isn’t as user friendly as a server, I would recommend Windows Azure. It allows any device to connect to it, has good connection on it’s given region, and is supported by Microsoft the creator of Windows. Azure notifies its users if any outages has occurred and has a record of every server that has every gone down. It has been in many projects like this and is very well documented.
4. **Security**: Security on web service games can be provided by encrypting the data with https. The server will be protecting with Azure’s security features which includes Microsoft Defender for unintended programs and Microsoft’s Intelligent Security Graph that searches incoming data. The user will be contacting only with the server and never directly peer-to-peer which prevents any other user from sending user’s malicious code.