

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 07/28/2022 | Raymond Plassio | Initial |

**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)

Creative Technology Solutions has recently taken on a new client, The Gaming Room. The Gaming Room wants to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is currently available in an Android app only.

## [Design Constraints](#_2et92p0)

Check for unique team and game names  
Must be singleton (one instance of the game allowed at a time)  
Must work on multiple platforms

## [System Architecture View](#_ilbxbyevv6b6)

Nothing required here.

## [Domain Model](#_8h2ehzxfam4o)

Looking at the UML diagram you can tell the Entity class is the super class and Game, Team, and Player all inherit from it. One game should go on at a time with the players and teams coming from a list. ProgramDriver has main() in it so that is where the program will be executed. SingletonTester uses ProgramDriver. GameService has a reference to Game which has reference to Team which has reference to Player. That seems to be done through aggregation.

**"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 is not as popular as Windows or Linux. macOS server has been discontinued and future versions of macOS are not supported. This is a bad choice for server. | Linux can do all the things you would need it to do for the server side. It is cheap and open source so those two things are good, however, not many people are familiar with Linux so you would need to find someone who is good at it. | Windows is probably the best choice for the server because it is the most widely used and it would be easy to find people to maintain it, but it is the most expensive. | I have never heard of anyone running a server using a mobile device and I’m not entirely sure it would be possible to do everything you needed to do using a mobile device to run a server. The good thing about running a server using a mobile device is the cost would be very low. |
| **Client Side** | The cost of Mac is higher than Windows for comparable hardware so it may not be the best choice. Once setup though it would run as you expected. Less people seem to use Mac than others so development might be harder. | There isn’t a cost for Linux because it is free and open source. A high amount of expertise and time would be needed because it is not as widely used as Mac or Windows. | This is probably the best choice for client side because it is the most widely used and easiest to find people who are good at it. The cost would be high, but the time and expertise needed would be low. | Cost would be low with mobile devices. Expertise would also not really be as much needed because a lot of people now are very savvy with mobile devices, and they are easy to work on. A lot of time would be needed just due to how many different devices exist there would need to be a lot of work to make sure they all worked. |
| **Development Tools** | The programming language used to develop on Mac is Swift. IDE to use would be AppCode, CodeRunner, Atom, SublimeText just to name a few. My favorite one is VSCode and that’s the one I would use. | Programming languages used to develop on Linux would be Java, C++, and Lua for games. I would use VSCode to do these since it seems the easiest for me. | The programming languages and tools for Windows would be the same as Linux, so using C++, Java, Lua for games. The IDE I would use, again, is VSCode for this. | To develop on iPhone, which, is the most popular mobile device I would use swift. I would use VSCode to do this. But now a real problem is like I said before you would need to develop it for Android and other Mobile platforms since there are so many so C++, Java, and Lua might come into play. |

**Recommendations**

1. **Operating Platform:** Recommend an appropriate operating (server) platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.  
     
   I think that windows is the most appropriate operating platform for the server. It is the most widely used and therefore easiest to find people to maintain the server. Although it is most likely the most expensive, there is a reason for that and that is that the tools available for it are the best and most widely used.
2. **Operating Systems Architectures:** Describe the details of the chosen operating platform architectures.

With windows there are two main layers one is kernel mode and one is user mode. The kernel mode is good because you can use that and not affect anything else that you don’t really want to. Kernel mode is like the very bottom Windows layer, where registration file is so when you are doing something in that it doesn’t affect the top layers.

1. **Storage Management:** Identify an appropriate storage management system to be used with the recommended operating platform.

Windows has many storage management capabilities like Storage Sense, Disk Management, and Disc Cleanup. These are two types of tools. The one, Disk Management is used for stuff like setting up a new drive. The other ones just clean up what already exists on those drives, deleting temp files and such.

1. **Memory Management:** Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.

Memory Management is built into the operating platform. The best technique to use would be to make a database where the game’s files are located and then the application can just access them.

1. **Distributed Systems and Networks:** Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).  
     
   You would use the client-server model and the client would just get all of the information it needs from the server to play the game. A consideration is how good the server is because if that has downtime, the game would stop working.
2. **Security:** Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.

To protect user information between the various platforms, the data between the client and server would need to be encrypted. What is good about windows is it has Windows Defender built into it which contains a whole suite of security applications such as firewall and antivirus. Windows Defender may not be the best choice here though, since it seems sometimes too intrusive to the user experience. There may be better freeware options on websites such as https://www.majorgeeks.com/.