

<Draw It or Lose It >

# **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 | 09/13/22 | Lenny Konesko | Initial Draft |

**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)

In order to assign multiple players to a given team we would need to create a class that we would use to create unique teams. We would also need to create a class that we would use to create unique games.

To allow for unique game and team names we will have to create a list that can be looped through to search for existing game and team names. To determine if a given team or game name exists, we would check a unique identifier assigned to each class.

Finally, the customer only wants one instance of any given game to exist in memory at a given time. We can use our singleton service to help resolve this.

## [Design Constraints](#_2et92p0)

1. The company would need a URL name.
2. If this is going to be done via web development, then it would be ideal for employees to have web development experience rather than app development experience.
3. Cross compatibility between desired platforms.

## [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 diagram shows our ProgramDriver which includes our main class. This uses the SingletonTester class to test our Singleton’s behavior. We then have a base Entity class that holds several shared variables. These variables are then inherited by the Game, Team, and Player classes. These classes are then called and used in the GameService class.

**"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** | Like Windows, Mac is closed platform. This means that as a web host you will need to pay for the latest software update provided from Microsoft for Windows or Apple for Mac. | Typically, the least resources intensive OS. Provides the most customization options via the many Linux distributions. The largest weakness would be that it is the most challenging to set up. Tends to be command line only. | Windows tends to be resource intensive. There is a lot of built-in software in their server distribution that may not be used. Some advantages could be the simplicity of use and it tends to be the most user friendly as well as having a familiar GUI. | Due to hardware limitations mobile devices tend to not be very strong for server hosting. |
| **Client Side** | Much like my answer for Windows. In order to support Mac OS, you will need individuals with Mac expertise. There is also going to be extra time needed to handle compatibility and software updates. | Depending on the distribution that the client is using different plugin/drivers may be needed. This would require someone expertise in Linux distributions which could cost extra money. | In order to support windows OS, you will need individuals with windows expertise. There is going to be extra time to handle compatibility coming from android. | Mobile web applications tend to function under separate rules from typical computer browser applications. You’re going to need individuals with expertise in mobile web development as it will likely take an entirely different mobile browser than the others. |
| **Development Tools** | To develop for Mac you will want to use high-power IDE’s available such as xCode. I would say that windows tends to be friendlier than Linux but not as much as Windows in regard to compiling and deploying code. | Open-source IDEs tend to be the standard on Linux (VSCode, Sublime, Atom, ETC)  Linux has compilers for nearly every language so whatever language the developers are comfortable with should be fine. | Essentially the same as Linux development tools. There tend to be more paid, high-power IDE’s available for windows. (Visual Studio, IntelliJ)  Windows tends to be friendlier than Linux in regard to compiling and deploying code. | With mobile your options are going to be a lot more limited. For cross platform development you are going to have to use something like XAML inside of Visual Studios to be able easily deploy to most mobile devices. |

## 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**: From my standpoint I believe for the purposes of this project. I believe The Gaming Room should choose a high availability Linux distribution (Ubuntu server, Oracle server, etc..). This would allow them to have the most customization and compatibility with other operating platform environments for future expansion.
2. **Operating Systems Architectures**: Linux has 5 basic architecture components (Kernel, System Libraries, System Utility Programs, Hardware layer, and Shell utility). The Kernel is responsible for major portions of the Linux OS. The System Libraries are used for implementing OS functionality. System Utility Programs are responsible for individual services. Hardware Layer contains hardware devices such as the Hard Drive, CPU, and RAM. Finally, the Shell is similar to a BIOS menu that allows for interaction between the Kernel and the user via command lines.
3. **Storage Management**: I would recommend using a cloud storage management system. Cloud storage tends to be used to host databases and other data sources in many modern applications. This would have to be outsourced to third party company that would typically host and provide much of the management.
4. **Memory Management**: Linux memory management is considered to be quite a complex system. However, much like Linux itself it is highly customizable. Linux memory management includes a plethora of things such as virtual memory implementation, memory allocation, mapping of files, and many more. In relation to “Draw It or Lose It” there are going to be multiple instances of games that host a number of different players. However, when the players leave said game the instance likely will still exist. This will need to be deleted from memory both front and back end. On the back end you could have Linux set up to automatically remove inactive sessions.
5. **Distributed Systems and Networks**: The company would need to start by setting up a network. The network is going to need to have open ports for incoming and exit ports for outgoing traffic. There will also need to be security protocols on the open and exit ports. There will need to be some sort of packaging handling software invested in that will read the database and send out information. (Typically done with JSON files.)
6. **Security**: Most Linux distributions tend to come with built in open SSL encryption. However, The Gaming Room should first create a policy based on what is going to be considered sensitive vs non-sensitive information. (User personal acct info vs images sent to screen, etc.)

The company would then want to make sure incoming or outgoing traffic to and from their servers are encrypted at a base of probably 128 bit minimum as it is typically the standard.