

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 | <05/20/2023> | Marvin Carpenter | Initial design for the project |

**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 client, The Gaming Room, would like us to develop a web-based game based on their current game, Draw It or Lose it. Their current game is only available in an Android app but they would like for this to serve multiple platforms such as Windows, MacOS and Linux.

## Requirements

*The requirements for this web-based game are as follows:*

* *Must have the ability to have one or more teams involved*
* *Each team will have multiple players assigned*
* *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.*
* *Must be made available on multiple platforms*

## [Design Constraints](#_2et92p0)

The design constraints for the client, The Gaming Room are:

* Creating the game to be compatible with multiple platforms via a web application.
* Having the ability to have one or more teams involved with each team having multiple players assigned.
* Creating a similar User Interface that they have on their current Android version or updating the already existing to match the web application.
* Creating unique identifiers for each instance of a game, team, or player to allow only one instance of the game to exist in memory at any given time.

## [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 Entity class is what creates a relationship between Game, Team, and Player classes. The arrow indicates that they all will inherit or get information from the super class, Entity. With UML, we can show this with inheritance by making Entity a superclass. Within the ProgramDriver class, this is where we will create the project. Indicated by the arrow pointing towards the SingletonTester, it will use that SingletonTester to test our code. The class GameService is the backbone of the game and how it will function. Since the program requires a unique identifier for Game, Teams, and Players, each class has been designed in the UML diagram and connected with lines to show they are associated with each other. The number indicated in each line is 0 which means that they can each have zero or more games associated.

**"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** | Apple’s operating system, MacOS, has easy accessibility and server configuration. These offer advantages for MacOS clients on a network. Maintaining a MacOS server can be expensive and difficult for companies to use that utilize third party programs. The graphical user interface is easy to use. | A major upside of why many companies such as Google utilize a Linux server is due to is free and open source. Being open source, this allows for companies to customize their security as needed. If you have little to no experience using Linux, this can create a difficult learning curve to navigate the platform. Some third-party programs and applications may not be supported. | Windows is a very widely used operating system. The licensing required for Windows can be expensive. Security is constantly updated and very well respected. Windows supports a large variety of third-party applications and software which is why most companies rely on them. | Two widely known mobile server companies are Oracle and Eurocom. These mobile servers can manage things such as users, applications, and data. These offer both iOS and Android dev tool support. One downside is mobile devices specs will vary. |
| **Client Side** | Costs for client side for MacOS would be high due to not being open source. Due to Windows being a more popular operating system, there may be a learning curve to new users but not as much so as it would be for Linux. | Linux is open source which means that the costs would be extremely low. However, due to the low percentage of Linux users, the training required would be extensive due to the complexity. | Similar to MacOS, the costs would be high since it is also not open source. Approximately 69% of the population uses Windows OS so there would hardly be a learning curve due to the simple design of their operating system. | Mobile servers are fairly flexible which means that costs would be low. Although implementing would be difficult, approximately 86% of the world population owns and uses a smartphone which means the training on this would be minimal. |
| **Development Tools** | On MacOS, you are able to run both Windows and Linux by utilizing local virtual machines.  You can use apple dev program for $99 per year. All applications must be extensively reviewed and approved through Apple. While JDK and Java are not native to MacOS, you are able to set them up. Some dev tools you can utilize are Visual Studios, GitHub, Pycharm, etc. | Due to being open source, Linux has many options for development tools. While it takes more knowledge to setup, in return it will give you more flexibility and options.  You can run both MacOS and Windows through a virtual machine. | While you can not run MacOS through a virtual machine, you are able to run Linux.  You can use Visual Studio Code for Html/Javascript and Eclipse for Java. | Depending on what Mobile OS you are using would determine your applications and IDE’s to develop. |

## 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 this I would highly recommend Windows as the operating system.
2. **Operating Systems Architectures**: Windows Operating System is designed by Microsoft. It allows an efficient way to run programs, manage files, connect to the internet, and for entertainment.
3. **Storage Management**: Windows comes built with Storage Sense. Windows Storage Sense is a silent assistant that works along with OneDrive. It can automatically free up space by removing files that are no longer needed. You can also use either HDD or SSD depending on budget. SSD is more expensive but allows for quicker access to your files and bootup time.
4. **Memory Management**: Memory management can be done via the storage sense as well. You can also install more RAM space if needed.
5. **Distributed Systems and Networks**: With Windows, you can utilize a cloud service known as Azure. Their service allows an ease of access for dealing with distributed systems and networks.
6. **Security**: Windows has their own built-in security that is available to everyone. It is being constantly updated to combat attacks and is well trusted.