

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_gjdgxs)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_3znysh7)2

[**Executive Summary**](#_2et92p0)3

[**Design Constraints**](#_tyjcwt)3

[**System Architecture View**](#_3dy6vkm)3

[**Domain Model**](#_1t3h5sf)3

[**Evaluation**](#_2s8eyo1)3

[**Recommendations**](#_3rdcrjn)5

## [Document Revision History](#_3znysh7)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <09/19/21> | Zac Blocker | Add team play abilities, and allow it to check user names to tell if it is already in use or not. |

**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](#_2et92p0)

Our client is wanting to expand their gaming application to allow user access on multiple different operating platforms. They would also like to have a multi team gameplay with unique usernames for each player playing. To achieve these we would like to compare the operating systems in order to find out the advantages and disadvantages running this gaming program. We will also be implementing the features asked for with multi team gameplay and unique usernames by running a quick check to make sure the username is not taken before allowing it to be used.

## [Design Constraints](#_tyjcwt)

A couple constraints would consist of having to build the game application that will run on multiple different platforms. Also our client wants an authenticator implemented for security purposes.

## [System Architecture View](#_3dy6vkm)

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](#_1t3h5sf)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**"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](#_2s8eyo1)

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** | When it comes to Mac we have more flexible terminal commands that would allow configuring the server and making changes a bit easier. | Linux a lot like Mac with flexibility in commands but also more cost friendly than Mac. | Windows has more software available to use compared to any other platform in question. | Mobile devices would work best if our server is immobile. The specifications are a better fit for other platforms compared to mobile. |
| **Client Side** | Mac will require moderate time and expertise to support clients. The cost will be similar to the cost of using a Windows platform. | Linux will require the highest level of expertise and time to support. The cost is lower than any other platform though. | Windows would require the least experience and time to upkeep clients. Cost of using windows will be roughly around the same as using a Mac platform. | Mobile is the most flexible on the client side as well as even the developer side. It would be more difficult to implement rather than using another platform. |
| **Development Tools** | Development languages can mostly be used for all OS therefore we have multiple options when it comes to a language or tool we want to work with. | Some of the popular languages consist of CSS,HTML,JavaScript and of course the supporting libraries to go with these languages. | Also there are more languages used for general purpose which would be Python, Java, php, Ruby on Rails. | Tools we can use in our project PyCharm, Visual Studio, Github, Eclipse, and notepad. |

## 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**: I would recommend windows. The game could run on other platforms as well if needed and each platform has their own pros and cons. On windows I believe we could develop this game to its maximum quality over the other platforms.
2. **Operating Systems Architectures**: The preferred operating system would be x86. This would be very powerful and easy to develop a game with hardware compatibility and support. 64 bit is commonly used nowadays in most development.
3. **Storage Management**: Storage is recommended to use an ssd drive due to the fact it is a lot faster when it comes to loading files off it. HDD is another but it would be the older method and a bit slower than what is more commonly used today which would be the ssd.
4. **Memory Management**: Since we recommended windows each process using 64 bit windows has a virtual address space of 8 tbs. All the threads can access its virtual address space but cannot access memory that a different process is using. This protects each process from becoming corrupted by another process.
5. **Distributed Systems and Networks**: Network based multi user interaction systems such as network games typically include a database shared among the players that are physically distributed and interact with one another over the network. Artery is used to support network games by optimizing the network performance.
6. **Security**: We will implement data protection using data controllers to minimize data collection and processing. This will reduce risk of personal data or even financial risk when it comes to the game.