

The Gaming Room’s *Draw It or Loose it*

# **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 | <4/16/2024> | Justin Schumann | Information updated |

**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)

On behalf of our new client, the Gaming Room we are assign to produce for them a multiple operating system available, web base game base on the former “Draw it or loose it”. The desire game should be multiple for different players from different locations. With separate teams guessing on which image is being form on the screen. There should be four rounds with a limit of at least one minute. The game should be accessible from a diverse range of different operating systems. MacOS, Linux, Windows Android, Apple.

## Requirements

Multiple platforms: the game should be able to be access by different platforms.

Multipliable: different players from different locations should be able top play side by side on the same team.

Web-Access: the game should be web base since it relied on the network.

Unique player, team and game IDs : the teams and players should have unique features in order to different them.

## [Design Constraints](#_2et92p0)

As stated in the fore mention, our client require this game to run on a diverse set of operating systems, have id checks, be multiplayer and web base. The current version of the game is a Android limited app, this newer version should be running on all the major plat-forms. To support more than one players playing on different teams at the same game there should be software architecture that can support such a thing. The IDs of the players should be unique, so the game’s system should be able to handler a good amount of memory.

## [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 below will be our game’s blue print for the coding creation. As seen, there is a relationship connecting the Entity class to the game, team and player classes. The Entity class will act as a “Super Class”, with the other three mention classes inheriting from it. For the code to be tested, a programDriver and a singletonTester class will be created. Those two classes connect to one another. What is being tested is whether or not if the game could exist in memory in one instance, which would be important for it to be playable. The foundation of the game itself will be within the GameService class, which will have none or more associations with the game class, which will be added one to the other classes. The Diagram will be a good guide for our project, though there may be adjustments we can easily start our work with a organized plan in place.

**"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** | Can make the type of changes that are useful. Mac OS Server is easily to get. Idea for web hosting for its Unix based. | Linux is open source so it is easily to get and low costs. Compare to Mac it have more server software options. With that in mind, some of its features are not readily available. | Experience users of windows are easily to find because of its long popularly. Though it is more costly than other services. | Though the costs will be less, mobile devices do not have the same functionality as other devices. |
| **Client Side** | Operating systems would require costs for they are not open source. How mac is set up, would limit the time and experience in using them. | Linux is not as wildly use as other services. The lack of experience people to use them will be harder to find. Though with it being open source, the costs will be lower. | The pros and cons will be exactly the same as the Mac, requiring costs and experience. There will be little significant difference. | Since the costs is low and they are easily to use, when it comes to money and experience, they have the best of both worlds. Though with more than one operating system they may require more work. |
| **Development Tools** | Ruby, CSS, HTML, JavaScript and Javal could all run on Mac. Swiftly is another good language to use for mac. Atom is the idea IDE for mac for its editing features. | IDEs that are commonly use on Linux such as Eclipse and Atom can do all of the major programming languages. | Eclipse and Visual Studio code are the major choices for Window users. Since both IDEs can use more than one language, they can be very useful. | Swift is the idea tool when using both Android and iPhones. Languages can be use on the major platforms. |

## 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**: When considering which platform to use, it is best to use the one with the most experience users and useful features. With its ability to instill all the major IDEs Windows will be the idea choice for our Gaming Room. Windows have been widely use and heavy popular, so there is no shortage of experience users for the app. Windows have built in security features, and a well establish technological support. All stated above is reason why Windows is our idea choice.
2. **Operating Systems Architectures**: Multimedia, graphics, web service apps. Windows’s architectures have what we need to make our game. Windows’s ability can be use to their best by setting up a GUI. With good access to memory, the operating system is perfect.
3. **Storage Management**: Windows have more than one service idea for Storage Management. Disk Management and Storage sense keep track of you are saving. With Storage Sense you can see how much storage is being use. Other features make it easily to kept and access files. Everyone working on this product should have a flash-drive to copy and store valuable files and information if everything goes wrong.
4. **Memory Management**: graphics and IDE files need to be kept on where you can find them. Windows have many apps that let you build a database or a liberally to save and access such things.
5. **Distributed Systems and Networks**: A strong and constantly maintain server network needs to be utilized for such a project. Since more than one player will be playing the same game from different locations, distributing system needs to be reliable.
6. **Security**: Since we using Windows, Windows Defender will be accessible. Another option is a cloud base service such as from Google Cloud. Instead of relying on our own hardware services from a cloud source could make things simpler. Since security should be a major concern for any project such as this, all data needs to be encrypted, there will personal and private information that could be use for sinister means by bad actors. To protect against bad actors and data, a firewall could be use to safe guard everything.