

# **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 | 05/26/2023 | Diana Diaz | Completed components on page 3 |
| 1.0 | 6/2/2023 | Diana Diaz | Completed Evaluation Chart |
| 1.0 | 6/17/2023 | Diana Diaz | Completed Recommendations on page 8 |

**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 Gaming Room is seeking Creative Technology Solutions (CTS) to develop and design a web-based version of their current game, Draw it or Lose it. The purpose of this new development is to make their game accessible by multiple platforms. The main conflict to overcome will be to create code that is compatible with the different operating systems in the market available to consumers to ensure that the game is accessible by anyone.

## Requirements

The software requirements that have been requested are to allow each game to have multiple teams involved, each team will also have multiple players. Each game and team must have unique names to avoid duplicate game or team names. Users need to be able to check if the name they are considering is already in use. Also, only one instance of the game can exist in the software memory at one time.

## [Design Constraints](#_2et92p0)

The main design constraint is that the code we develop must be compatible to allow any user to access and play the game regardless of the operating system of their electronic device.

Then, each game must have four rounds, each round must be one minute. The “Draw it” portion of the game must happen in the first 30-seconds. If the current team does not guess the puzzle, the remining teams have 15-seconds to provide one guess to try and win the game. The application and code must access and display images from the image library.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

Based on the UML diagram shown below we see that the Entity class will be the base class for the game application. The Entity class holds the id and name attributes which will be provided to the Game, Team, and Player classes because these classes will inherit from the Entity class. The UML shows that the code will ensure that each game, team, and player has an identifier and name, this will help check for duplicate names and ensure duplicates are not created.

We also see the ProgramDriver and SingletonTester classes will use eachother. The GameService class is strictly associated with the Game 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** | Mac is beneficial for stability, performance, and easy navigation. One of its main disadvantages is the high cost for servers. This is usually a deal-breaker when choosing which operating platform to operate.  This platform does offer a server-based deployment method. | This platform offers reliability, stability and security. The open- source operating system makes it a server favorite, along with cost-effective pricing. | Windows is a widely popular operating platform for servers however, it is a closed source system that also requires other Windows compatible paid software making it a bit pricey if considering all the additional Microsoft programs to purchase. Offers unique software for servers: Windows Server. | Application Programming Interface (API) compatibility between website and mobile device software.  Hardware limitations to prevent desktop functions be to available and executed.  Must include code for touch screen usage and compatibility. |
| **Client Side** | Mac operating system only runs on Mac computers which are significantly more expensive than other operating system devices. | Free to use by anyone making it easily accessible and cost effective for users/clients. | Windows is a user favorite platform, it is easy to use and navigate.  Requires additional Microsoft purchased software | As it might be known- not everything that can be easily accessed on a website can be accessed on a mobile device. When developing software for mobile devices, it is important to consider what features the user should be able to access on mobile device. Layout and design is important since mobile devices have a smaller screen than laptops or desktops. Touchscreen capabilities and ease of navigation when it comes to touch sensitivity would be important to consider as well. |
| **Development Tools** | Swift- Programming Language created by Mac to be used with Mac products.  XCode- IDE designed and created for Mac products. | Extremely versatile, able to work with any programming language, except, those created for Mac products specifically.  Many options for different IDEs to allow servers and developers use whichever IDE is most comfortable for them. | Also offers several options for programming language; making Windows a good option for developers to code in whichever language they feel most comfortable.  Many options in terms of IDEs as well. I’ve personally used Microsoft Visual Studio, Eclipse and PyCharm. All worked well. | HTML and JavaScript are the base programming languages for mobile devices. jQuery is a derivate of HTML and Sencha Touch is a derivate of JavaScript. |

## 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**:

My recommendation for an operating platform that will allow The Game Room to expand Draw It or Lose It would be to begin with Linux. This operating system seems to be widely accessible by any user who might be using any other operating system.

1. **Operating Systems Architectures**:

As shown on the section above, Linux seems to be a favorite by programmers and users. It is compatible with other operating systems, easily accessible and cost effective for both programmers and users. It offers reliability, stability and security. It’s also very versatile and is able to work with any programming language, except, those created for Mac products specifically.

1. **Storage Management**:

Linux utilizes XFS, which is a well-known high-performance file system for storage. It has a high capacity for data and can grow in capacity as long as there are unallocated blocks(extents).

1. **Memory Management**:

Linux uses a virtual memory technique called Paging System, this technique divides memory into smaller chunks, called pages to allocate and deallocate memory space as needed. This allows a computer or device to use more memory by creating an extension of the original memory size.

1. **Distributed Systems and Networks**:

One of the ways in which The Game Room can ensure the Draw It or Lose It game is accessible in different platforms is of course, creating a code that is compatible with different operating systems platforms. Additionally, I found that direct virtual memory access (DVMA) would be beneficial as well. The benefit of this is that it allows virtual addresses to undergo translation to physical addresses, it allows transfers between two-memory mapped devices without the use of a CPU or use of main memory.

1. **Security**:

The Game Room should ensure that as the game application is being created, the code is developed with safety in mind, there should be a clear understanding of what information the users will have access to and what information the developers/administrators can have access to. This will help so users cannot edit or alter the game. The game application should also have administrator log-in credentials, this would be beneficial in case of a data breach or a hacker attempting to access the code of the game. Ideally, this would prevent any unauthorized users accessing the code of the game and make unauthorized changes to the game structure.