

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

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

[Table of Contents](#bookmark1)**2**

[Document Revision History](#bookmark2)**2**

[Executive Summary](#bookmark3)**3**

[Requirements](#bookmark4)**3**

[Design Constraints](#bookmark5)**3**

[System Architecture View](#bookmark6)**3**

[Domain Model](#bookmark7)**3**

[Evaluation](#bookmark8)**4**

[Recommendations](#bookmark9)**5**

## [Document Revision History](#bookmark10)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/26/2024 | Robert Robbins | Completed Executive Summary, Design Constraints, and Domain Model sections. |

**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](#bookmark11)

The Gaming Room wants to expand their game Draw It or Lose It to reach a wider audience. This game is currently only available for Android devices and The Gaming Room wants to create a web based version. The web app needs to be able to have multiple teams per game and each team needs to be able to have multiple players. When creating new games or teams the new names will be compared to existing games and teams to prevent duplicates from being created. There will only be one game instance allowed at a time. The games, teams and players will be stored as objects and the will all be based on the same parent class. The game object will keep track of all of the teams, and the team object will keep track of the players.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client*’*s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#bookmark12)

This will be a web based game where players submit guess and are sent pictures which means a lot of exchanged data. This means the program will require a backend that receives and sends information based on game play. For example if a player guesses what a picture is the guess needs to be communicated to every player in the game. The game will also require a database to store player and game information as well as all of the pictures and past game data.

## [System Architecture View](#bookmark13)

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](#bookmark14)

The UML diagram below shows the structure of the game. There will be a driver Class that runs the program and as is shown in the diagram uses the SingletonTester class. The Program driver class will also use the GameService class to create a single game instance. The GameService class is used to ensures that no duplicate games are created by keeping a list of all active games. The Game, Team and Player class are all based on the Entity class. This is an example of polymorphism because they are all instances of the Entity class but have different attributes and methods. These class also show inheritance because they all have the same base attributes and methods that the Entity class has. All of the attributes in these classes are marked as private which means that users can only access and modify them by using the class methods which is an example of encapsulation.

**"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](#bookmark15)

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 OS is relatively simple to use and easy to learn. There aren’t drastic changes from each version. There are less viruses that target Mac. This os isn’t as common so less people are familiar with it. | There are many distributions of linux so there are a lot of options. There are distributions that are build specifically to be used as servers. This os isn’t as common so less people are familiar with it. | This is the most widely used os and most people are familiar with it. Most software is compatible with it. There are often major changes between versions. | There are two major operating system for mobile devices android and iOS. There is a huge variety of devices that use android so more thought has to go into making the app work well across them all. iOS has less devices so its simpler to build a app. |
| **Client Side** | The gaming room is going to be a web game so only one website needs to be built. When building it the developers need to make sure it works on all web browsers and the most common one for Mac is safari. | Most distributions of linux come with Mozilla firefox as the default web browser so the website must work on that. | Windows comes with Microsoft Edge as the default browser the website needs to be compatible with that. | The gaming room is already an android app so a iOS version needs to be built. This will be built using iOS languages and will be very similar to the android app. |
| **Development Tools** | Xcode is a popular ide for developing on Mac. Eclipse could also be useful especially since it will be built using java. Java will be a good choice of language for each platform because it can run on everything. | There are many ides that could be used on linux. VScode is a really popular one or intellij would be a good choice for working with java. | Visual studio is a popular IDE for windows or eclipse could be used. | Xcode would be a good choice for developing a iOS app. Xcode is a apple product so it works well with other apple products. A new iOS app would most likely be built using swift. |

## 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**: The Gaming Room should use a Linux server system for the Draw it or Lose it game. The Linux OS is one of the most used operating systems for servers which means there is a lot of documentation available. Because it is so widely used it will be easier to find developers to hire and there are a lot of hosting services available to build on. One option would be to use Amazon Web Services which has a large range of services available which could help simplify the development of the app.
2. **Operating Systems Architectures**: The Linux operation system architecture has several levels which are the user mode, kernel and hardware. They are user portion where the user interacts with the system through a graphical interface or a shell command line. The kernel is most important part of the operating system and this is what makes system calls to the hardware and manages resources. The hardware is the physical parts of the computer like the memory, hard drive and CPU which are resources.
3. **Storage Management**: The linux file system should be used to store all of the images for the game. The images can be stored in a folder on the linux server and when the site needs to pull one up it will point to that images memory location. A database could be used but the images would be store as binary strings that would be very long and take up a large amount of resources.
4. **Memory Management**: The linux system manages memory by having the kernel make system calls to allocate or deallocate something to memory.
5. **Distributed Systems and Networks**: The Gaming Room should use a rest api structure for their website to communicate between the server and the different players. The players would receive data from the server and they would make requests when submitting guesses. The server would then communicate between all the different players in the game.
6. **Security**: The gaming room would need to do its best to secure players login credentials. They could do this by encrypting sensitive data using algorithms to hash users passwords and userid’s so that if there was a data breach it would be very difficult to get any meaningful data. It would be best for The Gaming Room to use a service to manage users and security so that more attention could be placed on developing the game.