

Draw It-or-Lose 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 | 05/24/2023 | Reon Johnson | All sections were revised to meet assignment objectives |

**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 Draw it - or – Lose It game will require a number of things to be operational. The current code only performs creation of players, teams, and the Game Service. We still need to worry about the graphical interface that will need to be applied to allow users to view images and enter their player and team information. Hosting is another issue as this game must be made public to users to access over various platforms.

## Requirements

Draw It or Lose It is like the 1980’s television game show *Win, Lose or Draw*. Teams will compete to guess what is being drawn, so it is required that multiple Teams can be saved relative to their game created. Each team must be able to have multiple players. Rather than a player drawing images on an easel to help team members guess the puzzle (a phrase, title, or thing), the application will render images from a large library of stock drawings as clues. Game and team names must also be unique as well, allowing users to check if a name already exist in the game. Only one instance can exist in memory at a time so a singleton method must be created to perform this action.

## [Design Constraints](#_2et92p0)

A design constraint to consider is the graphical display that has yet to be implemented. Users need to be engaged visually for this design to be successful. In a web-based distributed environment, many types of systems will access the application. This means that the game should run on all OS platforms. This will require more time to prepare the application as it will require different languages. Another constraint is understanding how the application will be hosted. It could be hosted on company owned hardware or through the cloud to obtain more functionality.

## [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 outlines the way the Gaming Room package (com.gamingroom) is structured. We have the ProgramDriver class which contains the public main() method, and serves as the primary driver for our program. This ProgramDriver class uses the SingletonTester class, which tests if a single instance exists in memory at the time of execution with the testSingleton() method. Then we have the Entity class, which will serve as the parent class and provide all the general attributes to be inherited by the Player, Team, and Game classes. All three child classes contain inherited public constructors with id and name parameters which allows for relative object or instances to be created. The Game and Team classes also contain private list that hold multiple instances of game and team objects. Lastly is the Game Service which holds methods and attributes to call and add games, teams, and players.

**"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 OS server offers many advantages to other macs in the environment. However to other OS’s like windows and Linux it may have issues depending of what is being served to clients. Graphical designs are superior on Mac devices and be used for hosting content or media served to applications. | Linux servers offer a variety of services. It’s highly customizable and can support many different design architectures. It can start to lack functionality when attempting to perform graphical functions. The Linux OS is great for building and designing tasks, workloads, and other various services, or processes in web applications. | Windows has a complete OS’s dedicated to server side offerings. Windows server allows you to manage entire environments. This could include both physical devices and remote desktop/VDI environments. | While mobile OS’s such as Apple iOS and Android can perform basic server-side operations, they do not retain the bandwidth, memory, and space to host full server-side capabilities. |
| **Client Side** | The MacOS will utilize the Safari client to access the developed game and it will assist by delivering speed and great graphical displays. The downfall is that this would require separate resources to be dedicated to the development of game for safari because of Mac’s proprietary OS build | Because Linux is open source there are many clients that can come with a single distribution. This is a benefit to versatility but for the development of the game for a particular client is useless on this operating system | Windows utilizes internet explorer and Microsoft edge as its primary client because these clients are so adaptable to running various forms of frameworks, packages, and scripts. This would help us with saving money. Time would not need to be extended for developing | Special development must surround developing for mobile clients. This would involve spending extra time packaging the code to operate in mobile devices, and extra money to pay for the labor required. |
| **Development Tools** | All developments languages can be utilized to develop overtop of the macOS. | All developments languages can be utilized to develop overtop of the Linux OS. | All developments languages can be utilized to develop overtop of the Windows OS. | Select languages and tools can be utilized to develop in mobile standards. |

## 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 windows server OS will be the best option being that it offers solutions to meet all requirements explained by the client.
2. **Operating Systems Architectures**: The architecture will consist of a filesystem that is stored on a hard disk. It will also contain a driver controller and a I/O controller to allow for devices to communicate with the OS.
3. **Storage Management**: A good storage solution would be to host the data in the cloud. However, if it is to be done on-prem, then SSD HD should be utilized for faster storing and retrieving times. This would assist with slow loading screen and buffering of images.
4. **Memory Management**: Windows allows you to use tools such as Windows Memory Diagnostic to perform memory checks and diagnose issues. For windows, the memory is primarily in the form of random access memory (RAM). The OS, performs allocation, and deallocation to the memory dedicated.
5. **Distributed Systems and Networks**: The efficiency of the game can be achieved by distributing systems like the web servers, and storage of the games photos in the cloud API calls can be made to retrieve the images from storage. The game should be broken down into endpoints that can be queried using REST and
6. **Security**: Authentication is necessary for our security implementation. Users need to create usernames and set passwords. This would help control who could access the game. Any endpoints listed for the game should have permissions applied to implement least privilege.