

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

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

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03.18.2022 | Jessica Megaro | Initial Design Implemented |
| 2.0 | 04.03.2022 | Jessica Megaro | Evaluation added pg. 3 |
| 3.0 | 04.17.2022 | Jessica Megaro | Recommendation added pg. 5 |

**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)

CTS is being tasked with the development of a gaming application, “Draw It or Lose It”. The client The Gaming Room is not equipped to set up the environment. CTS will prepare the design documentation and begin development to achieve a streamlined workflow for the staff at The Gaming Room to maintain. Currently the game exists only as an Android based app. CTS will develop the web-based app to serve multiple platforms.

The Game

Similar to 1980’s TV show Win, Lose or Draw.

Teams compete to guess at what is being drawn.

App renders images from a large stock library as clues.

4 rounds of 1 minute each.

Drawings are rendered at a steady pace, finishing at the 30 second mark.

If not guessed, opposing teams may offer one guess to solve within a 15 second limit.

## [Design Constraints](#_2et92p0)

Must be available on multiple platforms, currently an android app exists. The game can either be translated to be able to use universally or updated and released with a new appearance, but still feel familiar to current users.

The game must be able to have one or more teams playing, with each team having multiple players.

Game and team names will be unique. When choosing, must be warned if a duplicate name is attempted.

One instance of the game can exist in memory at a time. Using unique identifiers for each instance of the game, team and player will accomplish this.

## [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 Entity class creates a relationship between Player, Game, and Team. Entity is the base class used to hold basic attributes and behaviors. While each class below Entity has additional unique attributes, they also use inheritance to share the attributes of Entity. The SingletonTester is being pointed to by ProgramDriver. This will test the ability to have only one instance in memory at any given time. GameService holds the method and game functionality. The classes are designed so that each has a unique name.

**"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** | MacOS offers a strong GUI and server-based environment. Hardware is expensive and fixing hardware issues may require 3rd party repairs. If the client is hosting their own servers, the hiring pool for engineers able to administer MacOS server environments can be smaller than Windows or Linux engineers. Licensing is included in the purchase of the hardware & operating system. | Linux can run on a wide range of hardware, optimizing cost, and allowing in-house repairs. The OS is free/open source, costing nothing in licensing unless enterprise support such as service from Redhat is required. Hiring pool for engineers able to administer Linux server environments is large. | Hosting with Windows requires licensing and can vary in cost based on the requirements needed for the host. Licensing for Windows can be complicated depending on what features of Windows and how many clients your application would need to be supported. It is advised to consult someone familiar with Microsoft’s pricing structure. | A mobile device would not be a good option for use as a dedicated server. In general, the hardware used in mobile devices is limited compared to typical hardware found in dedicated servers and would generally not be able to perform well enough to serve as a dedicated server. |
| **Client Side** | All the common OpenSource browsers such as Chrome, Firefox, and Opera support MacOS. In addition, MacOS provides the Safari browser.  Apple has a limited set of configurations for their hardware, which makes testing easier. This allows getting a reasonable level of test coverage with a small number of devices, depending on what generations might be supported. | All the common OpenSource browsers such as Chrome, Firefox, and Opera support Linux.  Linux OS and hardware configurations vary widely. Given the vast number of Linux distributions, GUI managers, and hardware capable of running Linux as a desktop environment, test coverage for even a small subset can be challenging.  It would be worth identifying the lower range of hardware that would be supported, and test performance and compatibility there. | All the common OpenSource browsers such as Chrome, Firefox, and Opera support Windows.  Like Linux, hardware configurations for Windows can vary. The number of OS versions is more limited, though.  It would be worth identifying the lower range of hardware that would be supported, and test performance and compatibility there. | Building clients for mobile devices offers its own set of considerations, given that there are several types of mobile devices (iOS, Android).  There are many variations of Android devices, including different processors, screen sizes, OS versions & capabilities making test coverage very difficult.  iOS devices have fewer variations differing mostly in CPU speed and available memory. iOS versions can differ, however older versions tend to be used a lot less, and testing focus can be on the last 2-3 OS versions. |
| **Development Tools** | Most modern programming languages can target MacOS. Apple has an IDE called Xcode which provides common tools like a debugger, and profiler for measuring performance.  Other tools are available for MacOS as well, such as Eclipse, VSCode, IntelliJ IDEA.  While there are IDEs and other development tools for both Web and Server-side development that require licensing fees, there are plenty of OpenSource, and free options. | Most modern programming languages can target Linux.  Linux has access to common opensource tools such as GDB, and LLDB for debugging, as well as perf for profiling.  Other tools such as Eclipse, VSCode, and IntelliJ IDEA are available for Linux as well.  While there are IDEs and other development tools for both Web and Server-side development that require licensing fees, there are plenty of OpenSource, and free options. | Most modern programming languages can target Windows.  Microsoft has its own set of tools for debugging and performance such as WinDbg and Visual Studio.  Other tools such as Eclipse, VSCode, and IntejjiJ IDEA are available for Windows as well.  While there are IDEs and other development tools for both Web and Server-side development that require licensing fees, there are plenty of OpenSource, and free options. | Developing for mobile devices often requires dedicated teams to target both iOS and Android. A team focused on each OS can help deliver the same functionality for both on time.  Generally, when developing for iOS, Swift or Objective-C is used. This allows easy access to the iOS frameworks that Apple provides its developers. Xcode can be used to develop in both Swift and Objective-C.  When developing for Android, Java is typically chosen, though Kotlin is picking up a lot of support as well. Eclipse, and Android Studio are IDEs that support developing for Android.  While there are IDEs and other mobile development tools that require licensing fees, there are plenty of OpenSource, and free options. |

## 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 recommendation from CTS is to use Amazon Game Lift and Linux.

1. **Operating Systems Architectures**: Game lift will host the game with all the power of AWS behind it. It will utilize dedicated servers and is capable of quickly scaling, deploying, and operating Draw It or Lose it. It is one of the best options for multiplayer games, and monitors as it runs to replace unhealthy instances as well as load balancing the server traffic. The service is available in 15 global regions on 5 continents. It is also one of the most secure platforms to choose and AWS additionally offers over 175 other features and services. It is recommended to have Game Lift launch Linux based instances, benefitting from the stability and efficiency of Linux.
2. **Storage Management**: Amazon Game Lift offers SSD-based storage without a monthly commitment. This serves the requirement for Draw It or Lose It. By running in AWS, access is granted to DynamoDB and other cloud-based databases where game information and user information can be stored and accessed from the game server instances.
3. **Memory Management**: Game Lift allows you to scale vertically or horizontally by either increasing CPU when more players are playing or allow more servers to operate to adjust for capacity. It allows The Gaming Room to customize configuration of memory, CPU and network capacity. Having the ability to auto-scale both ensures a great gaming experience as well as optimizing use of the available memory.
4. **Distributed Systems and Networks**: The restful API is what allows the game to be distributed amongst all the supported platforms. One of the benefits of using the service such as Game Lift is that it guarantees connectivity and security with as little outage time as possible. Being able to depend on such a large infrastructure provides networking reliability that a smaller infrastructure would be less likely to emulate. Since Game Lift is available in 5 continents and 15 global regions the servers can spin up instances where needed, allowing services to be physically closer to the players. This enhances connectivity as well as limits latency issues.
5. **Security**: Game Lift offers multiple security services. The Gaming Room would benefit from having such a robust environment such as AWS regarding security. It protects users from DDOS (distributed denial of services) attacks. Currently, a feature is offered at no additional cost that generates a TLS certificate and generates a DNS entry for each instance. It will authenticate the client and server connection and encrypt all game communications, which is of greater value in a multiplayer game such as Draw It or Lose It.