

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 | 01/25/24 | Amy Houseal | Initial Draft |

**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 has requested Creative Technology Solutions to develop a web-based game that serves multiple platforms based on their existing game, Draw It or Lose it. This game is loosely based on the 1980’s television game Win, Lose or Draw, where teams compete to guess what is being drawn. The application will render images from a large library of stock drawings as clues for the team members to solve the puzzle (a phrase, title, or thing). There are four rounds lasting one minute each. Drawings are rendered at a steady rate and are fully complete at the 30 second mark. If the team does not guess the puzzle before the time expires, the remaining teams can offer one guess each to solve the puzzle within 15 seconds.

The Gaming Room would like this to be a web-based game to have one instance of the game exist in memory at any given time. To implement this, CTS will utilize a singleton pattern to ensure only one instantiation of the game while utilizing unique identifies. Software requirements include having the ability to have one or more teams, each team can have multiple players, and that game and team names are unique and be checked if they are in use.

## Requirements

Technical Requirements

* Web Based
* Available on multiple platforms
* A game will have the ability to have one or more teams involved.
* Each team can have multiple players assigned to it.
* Game and team names are unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

Draw It or Lose it is an implementation of an existing program that currently only exists on Android app. To bring this game to other platforms, CTS may need to implement the use of other coding languages such as swift for iOS to make this possible. In addition to making this available on other platforms, The Gaming room has requested that this becomes web based which will require engineer familiar with web languages such as JavaScript, html, or C#. In addition, CTS will need to confirm if that current technology being used to support Draw It or Lose it is compatible with modern technologies such as varying web browsers (FireFox, Edge, Safari, Chrome). To ensure that both technological constraints are addressed, lengthy testing will be necessary. This could present a business constraint as this could delay the project’s timeline or require additional finances. It would be beneficial to receive more information on The Gaming Rooms timeline and budget.

## [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 is a visual representation of the program’s logical implementation. The game will begin at the ProgramDriver class which will initiate the GameService class. The GameService class is responsible for the creation of games, teams, and players. The singleton pattern can be seen within this class and is what will satisfy The Gaming Rooms technical requirement of having only one instance of the game in memory at any given time.

The principle of encapsulation is exercised through the Entity class which extends to the Team, Game, Player classes. Inheritance can be seen through this as well because the attributes are private, and Game inherits from Entity. Other object-oriented programming principles can be seen through the Game, Team, and Player classes being subclasses of the super class Entity which inherits from those subclasses.

**"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** | The latest macOS is Big Sur and High Sierra which includes features such as Caching Server, File Sharing Server, and Time Machine Server. These features are only available with High Sierra or later. | The operating system for Linux is open-source and free for use, modifying and sharing. Since there is no cost to using the OS, this is a major advantage when implementing a web hosting server. Because of this and the large community of developers that use Linux, there are ample resources and support. | Windows servers are compatible with MS Access, which is GUI based, making it easily mapped. This removes technical challenges and is convenient for those with less command line experience. The price for implementing a Windows server can range from $500 to $6000. | Mobile devices are capable of hosting basic web applications that don’t require multi-user access. Due to this, there are no hosting costs, but the scalability is limited. |
| **Client Side** | Development for the Mac operating system, apple hardware is required. The IDE, Xcode does come with macOS but it is limited to Object-C and Swift which is can be a challenge. Using an open-source framework will allow the use the Kivy. | Development can be done with C, C++, Python and Java.  There is no cost for development and there are ample resources. There is support for multi-users in GNU/Linux platform. | Windows supports the bulk of programming languages and tools. Because of the large range of support, it encourages ease of testing compared to other operating systems. | Mobile devices can only host single user clients which is a negative when implementing a multi-user application. Apple devices are required development using Xcode with Objective-C or swift. In contrast, Android supports the more commonly used language, Java. |
| **Development Tools** | MacOS will support the use of programming languages, C, C++, Objective-C & Swift.  Xcode is the main IDE used for iOS development. Xcode is free for download with a $99 subscription fee for cloud use. Visual Studio Code can be used as a text editor macOS applications. | The main language for Linux is C. There are multiple IDE’s that can be used with C that are compatible with the Linux OS. Visual Studio and Eclipse are both free and compatible with C and the Linux OS. | Windows operating systems support the bulk of programming languages, with the primary languages being C & .NET. Most IDEs have versions available for Windows operating systems with the most popular being Visual Studio.  Visual Studio is free to download but more commercial versions of this can range for a fee between $500 and $6000 based on requirements. | iOS is developed in Xcode using programming languages Objective-C and Swift.  Android uses the IDE Android Studio, which is free and supports programming languages Java, C++, Go and Kotlin. |

## 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 Gaming Room to expand Draw It or Lose it would be Linux. My primary reasoning is its low cost for licensing and the low restriction of access to its data centers. Since Linux is so widely supported and has a large community, it offers good operability and security due to the availability of security tools. The front end can be written in nearly any language which provides the opportunity for scalability.
2. **Operating Systems Architectures**: The suggestion for the operating systems architecture would be to have the back-end server manage the data handling logic while the front-end server handles all rendering. This will manage the number of calls to the API and reduce the cost due to the decrease in calls and data center costs. Furthermore, this will increase the performance since the client can provide caching which will assist with loading images quicker in comparison to calling the API. Allowing the client-server to render the images will also reduce any network issues when it comes to framerate.
3. **Storage Management**: As for storage devices, Solid-State Drives are the current industry standard. In comparison to Hard Disk Drives, SSDs are more costly, but the benefits include their aversion to malfunctioning make it worth the investment. Hard Disk Drive have a reduced cost and are portable. With performance in mind, my recommendation would be a Solid-State Drive. Cloud tools for the server-side will allow scalability for The Gaming Room.
4. **Memory Management**: Using Page Caching, Draw It or Lose It should place the images into page cache so that they can be retrieved for future instances. This will improve the load time by not having to read from the disk. Page caching works by storing frequently used data into RAM. Once the data is read, it will check the page cache to see if the data has been previously requested. In the scenario it has never been requested, it will create a new entry into page cache and fill with the data from the disk.
5. **Distributed Systems and Networks**: For a program such as Draw It or Lose it, a distributed system where the hardware, data and software is shared would be recommended as it can easily be managed due to the size of the project. Since the resources are shared, this encourages the opportunity for scaling the project and in turn improving the program’s performance.

The front and back end should communicate via RESTful API which will allow the client/server to communicate transparently to the front end. This will also allow for nearly any operating platform to be supported.

1. **Security**: Throughout the development of this program, we have prioritized security. Through this, we were able to identify potential security vulnerabilities and handle them accordingly. One measure of security that has been taken is using encryption, specifically Transfer Layer Security encryption for the API. This is often used when transferring data between applications via the internet. In addition to this, we recommend implementing a firewall to filter HTTP traffic. In addition to this, the principle of least-privilege should be implemented. This will limit the user to their role within the game. Anything outside of the user’s role should be escalated to higher team members.