

# **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 | 03/19/23 | Asif Iqbal | Executive Summary, Design Constraints and Domain Model sections |
| 1.1 | 09/29/23 | Asif Iqbal | Filled out the Evaluation table. |
| 1.2 | 04/16/23 | Asif Iqbal | Recommendations and final 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 development team at CTS has been approached by The Gaming Room to create a web-based gaming application named Draw It or Lose It, which is inspired by the 80s gameshow Win, Lose or Draw. However, The Gaming Room's team lacks the expertise to set up the required environment for the game's development. CTS will assist in streamlining the development process of the web-based game application, and the hardware requirements will be evaluated only after finalizing the software application decisions.

## 

## 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](#_2et92p0)

• The game must be written in Java and web-based.

• The game must not have more than one instance in memory at a given time.

• The game must allow one or more teams to play, with multiple players on each team.

• Unique game and team names are mandatory.

• The game comprises four rounds, with each round lasting one minute.

• Drawings must fully render continuously for 30 seconds.

• Images sourced from a vast stock library.

• The first team has 30 seconds to guess correctly, followed by the remaining teams with 15 seconds to make one guess.

These design constraints help establish clear expectations for the game's functionality and meet the client's requirements. Additionally, given the web-based nature of the game, these constraints must be taken into consideration during the development process to ensure that the game works correctly.

## [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 class diagram demonstrates the implementation of object-oriented programming principles in the Game, Team, and Player classes, which inherit the Entity class through inheritance. Encapsulation is exhibited through the use of private attributes and public methods to access or modify them in those four classes and the GameService class. The diagram also shows zero-to-many relationships between the classes, such as between Team and Player, Game and Team, and GameService and Game. Abstraction is implemented in the diagram through various methods such as addPlayer(), addTeam(), and addGame(). These methods abstract the addition process of different objects from the user by only displaying that the object has been added. Additionally, the diagram illustrates the ProgramDriver class's relationship with the SingletonTester 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** | MacOS is a feasible option for hosting a web application on a server.  However, the requirement of utilizing Apple hardware can make it financially challenging.  The good news is that there is no licensing fee for the macOS software as it comes pre-installed with the hardware.  Although, due to the high cost of the hardware, it may pose difficulties in scaling up to accommodate more users. | Linux is the leading operating system for web hosting purposes. Due to its open-source nature, it often has lower maintenance and licensing expenses compared to closed-source operating systems like Windows.  Additionally, popular cloud providers such as Google and Amazon prioritize Linux over Windows. | Windows can be utilized as a server to host a web application.  However, to achieve this goal, a particular version of Windows, Windows Server, is recommended, as it is explicitly designed for this purpose.  Out of all the available options, Windows Server has the highest licensing fee for any version of the Windows operating system.  This high licensing cost could potentially make it cost-prohibitive to scale up.  On the plus side, commonly available hardware can be used with Windows Server. | While it is technically feasible to use mobile devices as personal webservers or file servers, they are not well-suited for serving multiple users.  The hardware limitations of mobile devices, such as insufficient RAM, means that they cannot match the scalability of blade servers.  Moreover, the cost of developing and constructing hosting tools for mobile devices in-house is often unknown.  These factors underscore the challenges of relying on mobile devices as a reliable option for serving web applications to multiple users. |
| **Client Side** | Developing software for Macs requires the use of a Mac computer with the latest version of XCode.  Additionally, the macOS Software Development Kit (SDK) is typically coded in Objective-C or SWIFT, which are lesser-known programming languages.  Windows usage dominates 75% of the market compared to macOS's 16% means that there is a relatively smaller market opportunity for Mac software developers. | Developing software on the Linux platform is typically uncomplicated as popular programming languages such as Java, C/C++, and Python are available options.  Furthermore, Linux offers support for multi-user environments, making it a suitable choice for developing software meant for multiple users.  Despite its strengths, Linux development may have limited value as it is not widely adopted, and its market share remains relatively small. | Developing a web application for this platform can be a cost-effective choice as the development costs can be shared among other desktop operating systems.  In fact, this platform, combined with other desktop OSs, can be the most cost-efficient option as it allows developers to cover three operating systems with just one browser-based client.  However, ensuring compatibility with major browsers is crucial to ensure that the web application runs smoothly on this platform. | While mobile devices are not typically intended for multi-user environments, designing a client application for Android or iOS can be relatively simple.  For Android development, the Java-based Android Software Development Kit (SDK) may allow for the use of code that was initially developed for Windows or Linux.  In contrast, developing an iOS application requires the use of Apple's SWIFT programming language and may have similar hardware requirements as developing software for Mac computers. |
| **Development Tools** | When it comes to server development, macOS can support any programming language that is commonly used for web application development.  For client development, macOS can also accommodate any language that is used for web application development.  Most popular integrated development environments (IDEs) have versions that are compatible with macOS.  When developing iOS applications, developers usually work with Swift, a programming language developed by Apple.  IDEs such as Xcode, VS Code, or AppCode can be used for iOS development.  • On the other hand, Java or Kotlin are typically used for Android development. | Linux development can be done using popular languages such as C/C++, Java, or Python.  One of the advantages of using Linux for development is the availability of free and open-source IDEs. For instance, Python IDEs like NotePad++ and PyCharm are often free to use.  There are many IDEs for C/C++ development, but not all of them are compatible with Linux. However, Eclipse is a versatile IDE that supports multiple programming languages and is also free.  Overall, Linux provides developers with a wide range of options for developing and testing web applications without having to pay for expensive licenses or tools. | Windows is a versatile platform for web app development, supporting multiple programming languages such as Python, Java, C#, and more.  For client development, Windows can handle any language commonly used, including HTML, CSS, and JavaScript.  Most popular IDEs like Visual Studio, Eclipse, and PyCharm have versions compatible with Windows, providing a seamless experience for developers.  While iOS apps are typically developed using Swift, it is possible to use VS Code on Windows for iOS development, but it may have limited functionality.  On the other hand, Java and Kotlin are primarily used for Android app development, making Windows a viable option for Android development. | Developing software on a mobile device is not a viable option due to various limitations.  The smaller screen size and limited computing power make it difficult to perform complex tasks efficiently.  Although there are browser-based versions of some IDEs like VS Code available, they are still optimized for a desktop experience.  Furthermore, most programming languages are not available for mobile devices, which means you won't be able to compile, build, or test your code.  As a result, it is highly recommended to use a desktop or laptop computer for development purposes to ensure maximum productivity and efficiency. |

## 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**: When it comes to choosing an operating platform, Linux-based servers are highly recommended regardless of the frontend software used. Not only does Linux help to minimize license costs, but it also doesn't impose any restrictions on access to data centers like Windows servers can. In addition, the frontend can connect to the backend via APIs without any compatibility issues.

The Linux operating system offers robust security and efficient operability. Due to its wide usage as a server platform, there are numerous tools available including security software that can be easily integrated into your system.

Furthermore, since the frontend is agnostic, you have the freedom to write it in the language of your choice. For instance, if you are developing for iOS, you can write your frontend in SWIFT, for Android, in Java, and for Windows, in .NET.

1. **Operating Systems Architectures**: The proposed architecture for the Draw It or Lose It game involves a backend server responsible for managing the game environment, while the frontend handles client-based rendering. Given that the gameplay does not rely on immediate responses, low latency between the frontend and backend is not a critical requirement, meaning that asynchronous transmission could be utilized.

For enhanced scalability, a more advanced backend system utilizing containerized microservices and Kubernetes or Docker could be employed. The choice of cloud provider should be determined before finalizing the architecture, as each provider usually has unique proprietary tools that may affect the architecture design.

Rendering using the frontend could help minimize the more resource-intensive parts of the game, thereby reducing monthly data center expenses. Additionally, client-side rendering could help insulate gameplay from network issues since the game's framerate is critical to its smooth running. The client could cache a predetermined number of subsequent images ahead of active gameplay to ensure smooth rendering.

The choice of whether to create a browser-based game or a Java app for use on PCs or Macs still needs to be decided. However, it could be easier to develop a browser-based game and enable it via PWA (Progressive Web App) technology.

1. **Storage Management**: If the Game Room is not interested in buying their own hardware, there is no pressing need to make a choice between HDD and SSD storage media. Both options should deliver the performance required by the application, particularly when utilizing caching behavior and client-side rendering.

Adopting cloud-native tools would bring several benefits to the Game Room, including increased agility, simplified deployment, and improved reliability. Cloud services can be accessed from any location with an internet connection, enabling developers to work collaboratively on the application. Furthermore, cloud providers offer numerous features that can enhance the game's performance and functionality, while auto-scaling capabilities can accommodate fluctuating user demand for a seamless experience. By relying on cloud services, the Game Room can reduce the need for in-house IT support and minimize hardware maintenance costs.

1. **Memory Management**: Linux implements a pagecache system to manage data stored in main memory and virtual memory for allocated pages. It utilizes demand paging to reduce memory usage by only loading actively used pages into memory. Page replacement is performed using the Least Recently Used (LRU) algorithm.

The Android Runtime (ART) and Dalvik virtual machine rely on paging and memory-mapping (mmapping) to manage memory. This means that any modifications made to memory by an app, such as allocating new objects or accessing mmapped pages, remain resident in RAM and cannot be paged out.

Initially, memory management in iOS required retaining and releasing objects since it lacked Automatic Reference Counting (ARC). However, with the adoption of ARC, retaining and releasing objects is no longer necessary. Xcode automates this process during compilation.

Client-side rendering requires minimal RAM on the server, but the cost will scale with the number of users if utilizing a modern architecture with containers and microservices.

Client-side RAM usage is also minimal, with only 1-2 images being stored in memory at any given moment. Any RAM required to drive the client application, such as a browser, should also be taken into account.

1. **Distributed Systems and Networks**: Ensuring high uptime and avoiding outages is a primary concern for modern application development, which has led to the widespread adoption of cloud-native architectures. Cloud providers offer advanced replication and deployment capabilities, allowing services to be distributed across multiple locations and safeguarding against large-scale outages.

To facilitate communication between the frontend and backend, a RESTful API architecture will be utilized, enabling asynchronous data exchange. This approach ensures transparency between the client and server communication, regardless of the deployed frontend platform, such as Android, Windows, or iOS. The RESTful API architecture also allows for flexible, scalable, and secure communication between the frontend and backend, enhancing the overall performance and user experience.

1. **Security**: To ensure a high level of security for the game environment, role-based authorization will be implemented through an entitlements interface. This interface will provide effective administration of roles and accounts, allowing for granular control over user access and permissions.

The principle of least privilege will be employed to limit users to game controls such as game creation, team name creation, and team enrollment. In cases where greater user scope is required, a team-captain/member hierarchy may be introduced to enable limited user editing of teams or player addition/removal.

To further enhance security, no user will be granted ADMIN privileges within the system.

APIs will be secured using encryption, utilizing SHA 256 with 128-bit keys. TLS below version 1.2 will be disallowed to ensure the highest level of protection. Entrust will be the chosen vendor for certificate purchase, ensuring the highest level of security.

Finally, an industry-standard firewall will be implemented on the server, utilizing best practices for default settings. This additional security layer will help safeguard the game environment against potential attacks or breaches.