

<DRAW>

# **CS 230 Project Software Design Template**

Version 1.1

## 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 |
| --- | --- | --- | --- |
| 1.2 | <12/15/2024> | <Gaku Kikuchi> | <Added recommendations.> |

**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 goal is to redesign the "Draw It or Lose It" game for the web, making it available on multiple platforms. Key challenges like ensuring only one game instance and unique names for games, teams, and players will be solved using the Singleton and Iterator design patterns. This will make the game scalable, efficient, and easy to maintain.

## Requirements

*The game must support multiple teams, each with multiple players.*

*Game, team, and player names must be unique.*

## [Design Constraints](#_2et92p0)

Distributed environment: The system must handle multiple users and ensure consistent data across sessions.

Memory efficiency: Only one game service instance should exist (handled by the Singleton pattern).

Unique names: Names must be checked efficiently using the Iterator pattern to prevent duplicates.

## [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 shows the structure of the game application.  
The Entity class is the base for Game, Team, and Player, sharing common attributes like id and name.  
The GameService class manages all games, teams, and players. It uses the Singleton pattern to allow only one instance.  
The Game class contains teams, and each Team contains players.

The Entity class is inherited by Game, Team, and Player, reusing the common attributes.

**"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 provides high security and stability but is not commonly used for server purposes. Licensing costs are high, and it is not ideal for large-scale scalability.> | <Linux is open-source with no licensing costs, making it highly suitable for server use. It offers excellent scalability and has strong community support.> | <Windows Server is easy to set up and operate with its GUI-based interface, but licensing costs are high. It is suitable for small to medium-sized projects.> | <Mobile devices are rarely used as servers and are not suitable for hosting from a server perspective.> |
| **Client Side** | <Mac systems support popular browsers like Safari and Chrome, ensuring compatibility with web applications. However, the smaller market share compared to Windows means fewer developers target Mac first.> | <Linux offers strong browser support and is a developer-favorite platform for flexibility. However, it has a small desktop user base, requiring additional testing for web compatibility.> | <Windows is widely used and supports all major browsers like Edge, Chrome, and Firefox. Its broad adoption makes it a critical platform for testing.> | <Both iOS and Android dominate the mobile market. Developing responsive web applications and native apps ensures compatibility with these platforms.> |
| **Development Tools** | <Xcode is the primary IDE for iOS development, and it supports languages like Swift and Objective-C. Other tools like Eclipse and IntelliJ IDEA are also available for cross-platform development.> | <Linux supports open-source IDEs like Eclipse and Visual Studio Code. It is cost-effective due to the lack of licensing fees.> | <Windows provides IDEs like Visual Studio, which supports various programming languages. Licensing costs can vary based on the edition.> | <For Android, Android Studio is the go-to IDE, while Xcode is used for iOS. Cross-platform tools like React Native or Flutter can streamline development.> |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>

Amazon Web Services is recommended as the operating platform. AWS supports a wide range of operating systems and has robust capabilities for scaling applications across multiple geographic regions, making it ideal for expanding "Draw It or Lose It" into various computing environments. The global reach of AWS ensures minimal latency for users regardless of their location.

1. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

AWS supports multiple architectures, including Linux-based systems and Windows Server. This flexibility allows you to choose the architecture that best fits the game’s needs:

* Linux-based Systems: Offers reliability, scalability, and lower cost with support for various distributions such as Ubuntu, CentOS, and Red Hat.
* Windows Server: Provides easy integration with other Windows-based applications and services, although it may involve higher licensing costs.

1. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>

Amazon S3 (Simple Storage Service) is recommended for storage management. S3 offers high durability, availability, and scalability, making it perfect for storing game assets such as high-definition images and player data. It also supports lifecycle policies for managing data automatically as it ages, reducing storage costs.

1. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>

AWS utilizes advanced memory management techniques that include automatic scaling of memory resources according to the application's needs. This is crucial for "Draw It or Lose It" to handle variable loads during peak and off-peak gaming hours. Elastic Load Balancing and Auto Scaling help distribute game traffic and allocate resources dynamically, enhancing game performance and reducing latency.

1. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>

AWS provides several services to support distributed systems, such as Amazon EC2 for virtual server deployment and Amazon Route 53 for DNS web service. These allow seamless communication across different platforms and devices. AWS’s global network of data centers helps manage dependencies in distributed systems by ensuring connectivity and minimizing outages. For instance, Multi-AZ (Availability Zones) deployments enhance fault tolerance by running instances in several locations simultaneously.

1. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>

Security on AWS is comprehensive. To protect user information on and between various platforms, employ AWS Identity and Access Management (IAM) to control access to resources securely. Utilize AWS Shield for DDoS protection and Amazon Cognito for secure user authentication. All data stored on AWS can be encrypted at rest using Amazon S3’s built-in encryption features, and data in transit can be secured using SSL/TLS. Regular security audits and compliance checks further enhance the security posture of the gaming application.