

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 |
| --- | --- | --- | --- |
| 3.0 | <12/08/2023> | Mohamed Elmarzougui | Architecture Recommendation. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room's ambition to expand Draw It or Lose It across various operating systems presents a multifaceted software design challenge. This document show the business and operating constraints inherent in product delivery, we focus on providing solutions that address these constraints, ensuring the widest possible product delivery to augment The Gaming Room's market footprint.

## [Design Constraints](#_2et92p0)

Design Constraints for Web-Based Distributed Environment:

Security: handling user data and preventing unauthorized access, requires robust security measures, such as encryption and secure authentication.

Browser Compatibility: Differences in how browsers interpret and execute code can lead to inconsistencies in the game's performance. Thorough testing across major browsers is essential to ensure a seamless experience for all players.

Data Consistency: Maintaining consistency in distributed data storage is challenging, which impacts the accuracy and integrity of in-game data, such as scores and player progress.

## [System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

The Entity class establishes a connection between the Game, Team, and Player classes, indicating that they all inherit information from the Entity class. In UML, this relationship is visualized through inheritance, designating Entity as the superclass. Analyzing their interconnection, we observe that Team and Player exhibit a "has a" relationship type. Specifically, Game has a Team, and GameService has a Games. In UML, this association is referred to as aggregation (HAS-A), signifying that one class instance holds a reference to another class instance. In practical terms, when a user "has a," it means they possess an instance of one class while having a reference to an instance of another class. Examining the UML diagram, we notice that GameService has a reference to Games, Games a reference to Teams, and Teams references of 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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Advantages: Mac servers are known for their reliability, security, and ease of use. They integrate well with other Apple products and technologies.  Weaknesses: Limited hardware options compared to other platforms. Not as commonly used for web hosting as Linux or Windows. | Advantages: Linux servers are renowned for stability, security, and cost-effectiveness. A wide range of distributions caters to various needs.  Weaknesses: Compatibility issues may arise with certain proprietary software. Requires a good understanding of command-line interfaces. | Advantages: Windows servers offer seamless integration with Microsoft technologies and excellent support for .NET applications. User-friendly interfaces for configuration.  Weaknesses: Generally considered less secure than Linux. Licensing costs may be higher. | Advantages: Mobile devices can serve as lightweight servers for specific applications. Mobile app development frameworks provide easy deployment options.  Weaknesses: Limited resources compared to dedicated servers. Not suitable for resource-intensive server tasks. |
| **Client Side** | Development on Mac may involve higher initial hardware costs, but macOS software is typically freely available.  Expertise: Knowledge of Swift/Objective-C for macOS development is essential. | Cost: Linux development is cost-effective as most distributions are open-source.  Expertise: Proficiency in languages like C, C++, Python, or Java is crucial. | Cost: Windows development tools may have licensing costs, but Visual Studio Community is free.  Expertise: Proficiency in languages like C#, .NET, or even C++ for Windows development. | Cost: Development costs vary based on the chosen mobile platform (iOS or Android).  Expertise: Proficiency in Swift (iOS), Java or Kotlin (Android), or cross-platform frameworks like Flutter or React Native. |
| **Development Tools** | Languages: Swift, Objective-C. | Languages: C, C++, Python, Java.  IDE: Visual Studio Code, Eclipse. | Languages: C#, C++, .NET.  IDE: Visual Studio. | Languages: Swift (iOS), Java/Kotlin (Android), Dart (Flutter), JavaScript (React Native).  IDE: Xcode (iOS), Android Studio (Android), VS Code for cross-platform. |

## Recommendations

1. **Operating Platform**:

The Gaming Room should adopt a cloud-based (serverless) architecture running a Linux OS distribution. This approach reduces costs, complexity, and inefficiencies, promoting scalability and efficient solutions. Leveraging serverless services allows for distributed operations across multiple computing environments.

1. **Operating Systems Architectures**:

Using the modular structure of the Linux OS, minimizing disk space usage and improving efficiency by avoiding latent communication methods. The Linux OS's reliance on system libraries enhances security by restricting direct access to device hardware.

1. **Storage Management**:

For Draw It or Lose It, direct access storage ensures rapid load times, optimal virtualization, and an enhanced user experience. Indexed allocation, inspired by Linux, eliminates unnecessary sequential reads and minimizes storage fragmentation, enhancing efficiency.

1. **Memory Management**:

In Draw It or Lose It, Linux's memory management leverages virtual memory and demand paging. This allows efficient execution of multiple programs without physical memory constraints. The Least Recently Used (LRU) algorithm minimizes thrashing, optimizing page swaps and sustaining performance for an enhanced user experience.

1. **Distributed Systems and Networks**:

For Draw It or Lose It's growing user base, a serverless architecture scales seamlessly across multiple servers using load balancing for optimized responsiveness. The master-slave replication strategy ensures efficient and consistent database synchronization. HTTP and stateless unique session identifiers support client-server communication. A uniform RESTful API promotes multi-platform support.

1. **Security**

For Draw It or Lose It, prioritize simple login credentials over complex authentication methods. RESTful communication with session-based identifiers ensures stateless interaction. Securely store sensitive data, like passwords, in a hashed state in the database. Linux provides a robust foundation with built-in security features. To protect user information across platforms, implement secure communication protocols such as HTTPS for data in transit.