

# Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 2.0

## Table of Contents

[Draw It or Lose It 1](#_Toc159098825)

[**CS 230 Project Software Design Template** 1](#_Toc159098826)

[Table of Contents 2](#_Toc159098827)

[Document Revision History 2](#_Toc159098828)

[Requirements 2](#_Toc159098829)

[Design Constraints 2](#_Toc159098830)

[System Architecture View 2](#_Toc159098831)

[Domain Model 3](#_Toc159098832)

[Evaluation 3](#_Toc159098833)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 3.0 | 2/25/2024 | Scott Wells | Provide Software Design and recommendations |

[Executive Summary](#_sbfa50wo7nsh)

Creative Technology Solutions (CTS) aims to develop a web-based version of the game "Draw It or Lose It" for The Gaming Room. The game involves multiple teams and players, with each game having unique identifiers. To address these requirements, a robust software design is proposed, incorporating the singleton pattern for the `GameService` class and the iterator pattern for handling game and team names. The design ensures uniqueness and efficient management of game-related entities.

## Requirements

- A game supports one or more teams.

- Each team has multiple players.

- Game and team names must be unique.

- Only one instance of the game can exist in memory at any time.

## [Design Constraints](#_2et92p0)

The game application will be developed in a web-based distributed environment. Design constraints include ensuring scalability, responsiveness, and security. Implications involve selecting appropriate web technologies, frameworks, and security measures to meet client requirements.

## [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 a robust domain model:

- `GameService` is a singleton managing games, teams, and players.

- The iterator pattern is used for adding and retrieving games and teams, ensuring uniqueness.

- Inheritance is employed with the `Entity` base class holding common attributes.

- Object-oriented principles, such as encapsulation and inheritance, enhance efficiency.

**"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** | Mac offers robust server capabilities with macOS Server, providing built-in features for web hosting and scalable deployments. Licensing costs for macOS Server may be higher compared to other platforms, but it offers seamless integration with Apple's ecosystem and development tools. | Linux is renowned for its server capabilities, offering a wide range of distributions suitable for web hosting. It provides cost-effective solutions with minimal licensing costs and extensive community support. | Windows Server offers comprehensive web hosting features but may incur higher licensing costs compared to Linux. However, it integrates well with Microsoft technologies and development tools. | Mobile devices require a backend server for hosting the web application. Cloud-based solutions like AWS, Google Cloud Platform, or Azure are commonly used for hosting mobile backend services. They offer scalability, reliability, and flexible pricing options, although there may be additional costs based on usage. |
| **Client Side** | Supporting multiple client types on Mac requires using web development technologies like HTML5, CSS3, and JavaScript for building responsive web interfaces. Development considerations include compatibility testing across different browsers and devices, as well as utilizing tools like Xcode and Safari for iOS development. | Similar to Mac, supporting multiple clients on Linux involves using web development technologies for building responsive web interfaces. Development tools like Visual Studio Code, Atom, or Sublime Text are commonly used for web development on Linux. | Windows supports web development using technologies like HTML, CSS, and JavaScript. Development tools like Visual Studio are widely used for building responsive web interfaces for Windows-based desktops and mobile devices. | Mobile devices require developing native apps using platform-specific languages like Swift for iOS and Kotlin or Java for Android. Cross-platform frameworks like React Native or Flutter can also be used to build apps for both iOS and Android simultaneously. Development tools like Xcode for iOS and Android Studio for Android are essential for mobile app development. |
| **Development Tools** | For Mac development, Xcode is the primary IDE used for building macOS and iOS applications. Other tools include Sublime Text, Atom, or Visual Studio Code for web development. | Linux development tools include Visual Studio Code, Atom, Sublime Text, or Vim for web development. Terminal-based tools like Git and SSH are also essential for Linux development. | Windows development primarily relies on Visual Studio for building web applications. Other tools like Visual Studio Code, Sublime Text, or Atom can also be used for web development on Windows. | Mobile app development on iOS requires Xcode as the primary IDE, along with tools like Swift and Objective-C for iOS development. For Android development, Android Studio is the preferred IDE, along with Kotlin or Java for Android app development. Cross-platform frameworks like React Native or Flutter can be used for building apps for both iOS and Android. |

## 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:** Linux is indeed a cost-effective and flexible choice for hosting web-based applications like Draw It or Lose It. However, it's important to consider the specific requirements of the game and the development team's expertise. If the team is already familiar with Linux and comfortable with its administration, then it's a solid choice. Otherwise, it might be worth considering other platforms that align better with their skills and requirements.

**2. Operating Systems Architectures:** Ubuntu and CentOS are both excellent choices for stable and reliable architectures. However, the specific choice should depend on factors such as compatibility with existing infrastructure, support availability, and any specific features or preferences of the development team. It's essential to evaluate these factors before making a final decision.

**3. Storage Management:** Cloud-based storage solutions like AWS S3 and Azure Blob Storage offer scalability and reliability, which are crucial for a game application like Draw It or Lose It. However, it's important to consider factors such as cost, data transfer speeds, and integration with other cloud services. Additionally, evaluating the specific storage needs of the game, such as file sizes, access patterns, and latency requirements, will help make an informed decision.

**4. Memory Management:** Linux-based operating systems do indeed employ efficient memory management techniques. However, it's essential to consider the specific memory requirements of the game and optimize resource allocation accordingly. Techniques such as caching, virtual memory management, and memory pooling can help improve performance and scalability.

**5. Distributed Systems and Networks:** Implementing a distributed architecture with load balancers and redundant servers is a sound approach for ensuring seamless communication and minimizing downtime. However, it's important to design the architecture carefully, considering factors such as latency, bandwidth, and network topology. Additionally, implementing monitoring and failover mechanisms will help maintain reliability and performance.

**6. Security:** Linux-based servers do offer robust security features, but it's essential to implement additional security measures specific to the game application. This includes measures such as regular security updates, access controls, intrusion detection systems, and data encryption. Additionally, implementing secure coding practices and conducting regular security audits will help mitigate security risks effectively.