

# 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 | 03/24/24 | Paulina Weaver | Initial Release |
| 1.1 | 04/07/24 | Paulina Weaver | Evaluation |
| 1.2 | 4/28/24 | Paulina Weaver | Recommendation |

**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)

Creative Technology Solutions (CTS) is partnering with The Gaming Room to develop a web-based version of their popular game, Draw It or Lose It. This transition from the existing Android app to a web-based platform offers a unique opportunity to broaden the game's accessibility and appeal across various devices and platforms. In this adaptation of the classic Win, Lose or Draw game, teams compete to guess drawings rendered from a vast library of stock images. To ensure seamless gameplay, CTS will implement support for multiple teams, assign multiple players to each team, and ensure unique names for both games and teams, preventing confusion and enhancing user experience. Additionally, by implementing a singleton pattern, only one instance of the game will exist in memory at any given time, optimizing performance and resource utilization.

## Requirements

1. The game must be accessible across various platforms, including web browsers on desktop and mobile devices, to reach a wider audience.
2. The game will support multiple teams, each comprising multiple players. This team-based structure fosters collaboration and competition among players.
3. Game and team names must be unique to avoid confusion and ensure clarity when selecting teams and joining games.
4. To maintain consistency and prevent conflicts, only one instance of the game should exist in memory at any given time. This ensures seamless gameplay experiences for all users.

## [Design Constraints](#_2et92p0)

1. Developing the game for a web-based distributed environment introduces challenges related to compatibility, performance, and scalability. The choice of technologies and architecture must accommodate these constraints to ensure a smooth gaming experience across platforms.
2. Implementing mechanisms to validate the uniqueness of game and team names adds complexity to the application logic. This validation process must be efficient and scalable to handle a potentially large number of concurrent users.
3. Employing the Singleton design pattern for managing the game instance requires careful consideration of thread safety, resource management, and concurrency control. Ensuring that only one instance of the game exists while maintaining responsiveness and scalability is crucial.

## [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 consists of several classes interconnected to facilitate game management and player interaction. At the center is the Entity class, serving as a base for all game entities with common attributes and methods. The creation process is managed by the GameService class, which follows the singleton design pattern to ensure only one instance exists in memory at any given time. This is achieved by setting the constructor of GameService to private and providing a getInstance() method for instantiation.

Once GameService is instantiated, the driver class can utilize methods like addGame(), addTeam(), and addPlayer() to create and manage games, teams, and players. The Game, Team, and Player classes are subclasses of the Entity class, inheriting common attributes like id and name. The Game class represents individual games and contains teams, while the Team class represents teams with associated players. The Player class represents individual players.

These classes are related through composition and aggregation relationships, showcasing object-oriented principles such as encapsulation, inheritance, and polymorphism. Encapsulation is demonstrated by encapsulating data and behavior within each class, ensuring data integrity and reducing dependencies between classes. Inheritance is showcased through the subclass relationships, where subclasses inherit attributes and behaviors from their parent classes, promoting code reuse and maintaining a hierarchical structure. Polymorphism is evident in the way objects of different classes can be treated interchangeably, allowing for dynamic behavior based on the context.

**"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** | When it comes to Mac hosting web software applications, one of the key features that stands out is its architecture. Mac is based on Unix, which is known for its robustness and reliability. The server software is built-in to Mac, making setup and configuration a breeze. The user interface is easy to use, and the developer tools are robust, making it easier to develop web applications. Another benefit is security. macOS has strong built-ins for security, and regular updates are available. However, the hardware options are limited, and the cost is higher than other platforms. Despite the drawbacks, Mac’s integration with Apple products and stability makes it a good choice for hosting web apps, especially for companies that are heavily invested in Apple products. | Linux is one of the most popular open-source operating systems for hosting web applications. It is highly stable, reliable, and secure, making it an ideal choice for organizations looking for robust hosting solutions. Linux is a highly customizable operating system that offers developers the flexibility and versatility they need to optimize server configurations while managing web applications. It can be difficult for beginners to get started with, as it has a command-line interface and system administration tasks. Linux has a wide range of hardware compatibility and scalability, making it a great choice for businesses looking for a reliable and affordable way to host their web applications. | Windows has an easy-to-use interface and compatibility with other Microsoft products. Windows provides comprehensive development tools such as Visual Studio and supports frameworks such as.NET. It is also well-suited for web application development and deployment. Windows servers integrate seamlessly with other Microsoft products, making them more compatible and interoperable. On the other hand, licensing costs may be an issue. Windows servers may also be more vulnerable to security vulnerabilities than open-source options. Performance limitations may also occur under high load, requiring careful optimization. All in all, Windows provides a well-known and feature-packed platform to host web applications. | Mobile devices are omnipresent and widely available, providing users with the ability to access applications at any time, from anywhere. They are natively integrated with device functionality and offer offline access, improving user experiences and providing new functionalities. On the other hand, mobile devices are limited in resources, have different screen sizes and connectivity limitations, making it difficult to host complex web applications. Additionally, device fragmentation and security concerns make it difficult to ensure consistent performance across devices and platforms, while also protecting sensitive data and ensuring compatibility. Optimizing apps for mobile environments, however, can offer possibilities for increased visibility and monetization, as well as user engagement through the distribution of apps on the app store and other innovative features. |
| **Client Side** | Developing software for macOS typically involves higher costs due to the need for Apple hardware, software licenses, and acquiring macOS-compatible development tools. Additionally, allocating sufficient time for platform-specific development, testing, and optimization is essential, along with addressing any macOS-specific issues that may arise. Expertise in macOS development, including proficiency in languages like Swift and familiarity with macOS-specific frameworks and APIs, is necessary for optimizing software performance on macOS devices. | Developing software for Linux generally incurs lower costs, as many distributions are open-source. However, expenses may still arise from acquiring Linux-compatible development tools and resources. Similarly, developers need to allocate time for platform-specific development, testing, and optimization, ensuring compatibility across different distributions and versions. Proficiency in programming languages like C, C++, Python, and Shell scripting, along with familiarity with Linux-specific frameworks, libraries, and APIs, is crucial for optimal performance on Linux platforms. | Developing software for Windows may involve costs associated with acquiring development tools, licenses, and hardware. Like with other platforms, sufficient time is needed for platform-specific development, testing, and optimization, while addressing any Windows-specific issues. Expertise in languages like C#, .NET, and Visual Basic, along with familiarity with Windows-specific frameworks, libraries, and APIs, is necessary for optimal performance on Windows devices. | Developing software for mobile devices may require expenses for app store licenses, development tools, and testing on multiple platforms and devices. Time must be allocated for platform-specific development, testing, and optimization, considering unique characteristics like screen size and touch input. Proficiency in languages like Java (for Android) and Swift or Objective-C (for iOS), along with familiarity with mobile-specific frameworks, libraries, and APIs, is essential for optimal performance. Knowledge of platform-specific design guidelines and user experience principles is crucial for delivering a seamless user experience across different mobile platforms. |
| **Development Tools** | Developing software for macOS typically involves using programming languages such as Swift and Objective-C, with Xcode serving as the primary integrated development environment (IDE). Xcode provides a comprehensive set of tools for designing, coding, testing, and debugging macOS applications, including features like Interface Builder for designing user interfaces and XCTest for testing. | For Linux development, the commonly used languages are C, C++, Python, and Shell scripting, with IDEs such as Visual Studio Code, IntelliJ IDEA, Eclipse, and Sublime Text providing development environments. Additionally, command-line tools like Vim or Emacs are often utilized for coding and compilation. | Developing software for Windows typically involves using languages like C#, .NET, and Visual Basic, with Visual Studio serving as the primary IDE. Visual Studio provides a comprehensive suite of tools for designing, coding, testing, and debugging Windows applications, along with features like the NuGet Package Manager for managing dependencies. | For mobile app development, programmers commonly use languages like Java (for Android), Swift, Objective-C (for iOS), and Dart (for cross-platform development), with IDEs such as Android Studio, Xcode, Visual Studio, and IntelliJ IDEA providing development environments. Additionally, tools like Flutter/Dart DevTools and React Native CLI are utilized for cross-platform development. |

## Recommendations

1. **Operating Platform**: Based on the needs and considerations, Linux is the best operating platform to host the game. Linux is compatible with web browsers and devices such as desktop and mobile, making it widely accessible for players. Its proven stability and reliability ensure a stable gaming experience, while its open-source nature offers cost savings by removing licensing fees. Furthermore, Linux’s adaptability and flexibility allow developers to customize server configurations according to the game’s requirements. Combined with Linux’s built-in security features and frequent updates, it provides a solid basis for implementing strong security measures to protect user information and preserve the integrity of gaming experience. All in all, Linux offers a stable, reliable, and cost-effective solution for hosting Draw It or Lose It, aligning perfectly with the project's goals and requirements.
2. **Operating Systems Architectures**: Linux architecture is largely based on a kernel architecture. In a kernel architecture, each operating system service, including process control, memory control, device control, and file control, runs as a single large executable in the kernel space. This architecture allows for rapid communication between different parts of the kernel, resulting in efficient system performance. Additionally, Linux supports a modular approach through loadable kernel modules, allowing device drivers and other features to be dynamically loaded and unloaded as needed, enhancing flexibility and scalability. Furthermore, Linux can be customized extensively, with users having the ability to compile and configure the kernel to include only the necessary components for their specific requirements, minimizing the overhead and optimizing system resources.
3. **Storage Management**: A combination of a traditional file system and a cloud storage solution would allow optimal performance and scalability. For the traditional file system component, ext4 is a mature and stable choice with strong performance, reliability, and data integrity. It has been widely used and tested in various environments, making it a dependable option for hosting web-based applications. Additionally, integrating Amazon S3 cloud storage with Linux servers offers scalability, reliability, and advanced features such as high availability, durability, and automatic backup. Amazon S3's flexible storage tiers and pay-as-you-go pricing align well with Draw It or Lose It's requirements for scalability and cost-effectiveness.

By combining ext4 file system with Amazon S3 cloud storage, Draw It or Lose It can benefit from the reliability and performance of the traditional file system while leveraging the scalability and advanced features of cloud storage. This approach ensures optimal performance and scalability for the application, meeting the needs of its growing user base effectively.

1. **Memory Management**: Linux utilizes virtual memory, a technique that allows the system to use disk space as an extension of RAM, thereby providing a larger address space for applications to use. This ensures efficient memory allocation for the game, even when system resources are limited. Additionally, Linux implements paging, dividing physical memory into fixed-size blocks called pages, which allows the system to manage memory more effectively by swapping out less frequently used pages to disk and bringing them back into memory when needed. This approach minimizes memory fragmentation and maximizes available memory for the game's processes, ensuring smooth gameplay experiences.

Furthermore, Linux employs caching mechanisms to enhance memory utilization and system performance. By storing frequently accessed data in memory, caching reduces the need to fetch data from slower storage devices, thereby improving responsiveness and reducing latency during gameplay. This is particularly beneficial for Draw It or Lose It, as caching frequently used game assets or data, such as graphics or game state information, can significantly enhance overall performance.

1. **Distributed Systems and Networks**: To communicate across different platforms, it is essential to implement a distributed software architecture that is supported by a network. Client-server communication protocols such as RESTful APIs over HTTP/HTTPS are used to ensure compatibility between different platforms.

Linux server-side component, would serve as the central hub for managing game sessions, storing game data, and handling user interactions. It would expose a set of APIs that clients on different platforms can interact with to perform actions like creating a game, joining a team, submitting guesses, and retrieving game information. On the client side, whether on desktop browsers, mobile devices, or other platforms, applications would utilize these APIs to communicate with the server. This allows for a consistent user experience across different devices and platforms while preserving data integrity and synchronization.

To manage connectivity issues and outages in distributed systems, it's important to handle component dependencies carefully. Redundancy measures such as load balancing and failover mechanisms are vital for system availability and resilience. Also, implementing error handling and retry mechanisms helps deal with network disruptions and maintain session consistency. Also, regularly monitoring and maintaining the network infrastructure and distributed components to promptly identify and resolve any connectivity issues.

**Security**: Linux provides dependable security measures to safeguard user information. Encryption techniques such as HTTPS should be used to encrypt data transfer between user devices and servers. Utilizing tools like Apache, which support HTTPS, ensures secure data exchange, maintaining confidentiality and integrity. Furthermore, the user authentication mechanism JWT securely verifies user identities across different platforms, thereby preventing unauthorized access.

Additionally, Linux has strong support for role-based access control through tools like SELinux, empowering administrators to oversee user permissions and limit access to sensitive data or features within the game. By enforcing role-based access policies aligned with user roles and privileges, Linux enhances security measures effectively. Moreover, it's essential to regularly apply security updates and patches to the Linux server to address vulnerabilities and reduce potential risks. By staying vigilant with security measures, Linux ensures resilience against evolving threats, ensuring the safeguarding of user information and upholding a secure gaming environment for Draw It or Lose It players.