

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 |
| --- | --- | --- | --- |
| 1.0 | 05/26/24 | Sarah Wagner | Creation of document, add in executive summary, constrains and UML diagram explanation. |
| 2.0 | 06/09/24 | Sarah Wagner | Filled out evaluation table |
| 3.0 | 06/23/24 | Sarah Wagner | Filled out 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)

To help The Gaming Room transition "Draw It or Lose It" from an Android app to a multi-platform web-based game, we will ensure the game can support multiple teams and players, guarantee unique game and team names, and maintain a single instance of the game in memory. This approach addresses both business needs for unique identifiers and technical requirements for game functionality and memory management. The goal is to streamline development and provide a clear roadmap for implementing the web-based game.

## 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)

Developing the game application in a web-based distributed environment involves several design constraints. The application must be scalable to support a potentially large number of concurrent users and teams without performance issues. This will require an architecture that can scale as needed with load balancers and distributed servers.

Additionally, the game requires real-time updates and interactions. This will require WebSocket connections and optimized data handling to ensure rendering of images and processing of guesses. These constraints will influence the development process and require robust solutions to maintain performance and responsiveness.

## [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 ProgramDriver class is the entry point of the application. It utilizes the SingletonTester class to test the Singleton design pattern implementation.

The Entity class, acting as a base class, encapsulates common attributes and methods. These are inherited by the Game, Team, and Player classes, promoting code reusability and efficient management of shared attributes.

The GameService class, a singleton, manages the game's lifecycle and ensures unique identifiers for games, players, and teams. It demonstrates the Singleton pattern by maintaining a single instance throughout the application.

The Game class manages teams within a game session, and the Team class manages players within a team, both reflecting composition relationships. Each class encapsulates its attributes and methods, ensuring internal state protection. These principles ensure the game application is modular and scalable, effectively fulfilling the client's software requirements.

**"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 can host web-based software applications using server software like Apache, Nginx, or the built-in macOS Server. However, it is not typically used in production environments due to its higher licensing costs and lack of scalability compared to Linux servers. MacOS is more suited for development environments where developers use Mac hardware. For production, MacOS may be limited in terms of community support and available server management tools. Licensing costs can be significant, making it less attractive for large-scale deployment. | Linux is the preferred choice for hosting web-based software applications due to its open-source nature. It offers a variety of server distributions (e.g., Ubuntu Server, Debian) that are free to use. Linux servers are highly scalable and can handle thousands of concurrent users. The cost-effectiveness and extensive toolset for server management make Linux the most attractive option for server-side deployment. Its security features and customization capabilities also add to its advantages. | Windows Server is another viable option for hosting web-based software applications. It is known for its ease of use, particularly for organizations already invested in the Microsoft ecosystem. Windows Server offers support for .NET applications and integrates well with other Microsoft services. However, licensing costs can be high, and it might not be as scalable or cost-effective as Linux. Windows Server also requires more resources for maintenance and management, which can add to the overall cost. | Mobile devices are generally not used for hosting web-based software applications due to their limited processing power and storage. However, mobile devices can access and interact with web applications hosted on traditional servers. Mobile apps can serve as clients to web services but are not suited for server-side hosting. |
| Client Side | Developing client-side applications for Mac requires proficiency in macOS-specific development tools and languages, such as Swift and Xcode. The development process is relatively streamlined within the Apple ecosystem, but expertise in macOS development is necessary. The cost and time associated with developing for Mac can be higher due to the need for specialized hardware and software. Supporting multiple clients requires compatibility across different macOS versions and browsers, which can add complexity. | Linux client-side development often focuses on creating cross-platform applications using tools like Electron or web technologies (HTML, CSS, JavaScript). The cost is low due to the availability of open-source tools, but expertise in various Linux distributions and desktop environments is required. Ensuring compatibility across different Linux distributions can be challenging and time-consuming, but the flexibility and low cost make it a viable option. | Developing for Windows is often facilitated by tools like Visual Studio and languages like C# and .NET. The cost of development can be higher due to licensing fees for development tools and the need for specialized knowledge. However, Windows offers a large user base and extensive support for a wide range of applications. Ensuring compatibility across different versions of Windows and web browsers is necessary for a seamless user experience. | Developing for mobile devices requires knowledge of iOS and Android development environments, such as Xcode for iOS and Android Studio for Android. The cost and time can be significant due to the need for different codebases and expertise for each platform. Cross-platform tools like Flutter or React Native can reduce these costs, but ensuring consistent performance and compatibility across devices adds to the development effort. |
| Development Tools | For macOS development, the primary tools are Xcode, Swift, and Objective-C. Xcode is a comprehensive IDE that supports the entire development lifecycle but comes with a steep learning curve. There are no additional licensing costs for using Xcode, but it requires macOS, which requires investment in Apple hardware. The impact on the development team includes the need for familiarity with Apple's development ecosystem and guidelines. | Linux development leverages a wide range of tools and languages, including GCC, Python, PHP, Ruby, and various IDEs like Visual Studio Code, Eclipse, and PyCharm. These tools are generally open-source and free to use, making Linux a cost-effective platform for development. The impact on the development team involves managing various tools and ensuring compatibility across different Linux distributions. | Windows development often involves using Visual Studio, .NET, and C#. Visual Studio is a powerful IDE but comes with licensing costs, especially for the Enterprise edition. The impact on the development team includes the need for knowledge of Microsoft technologies and tools. While the initial costs can be high, the integrated development environment and extensive support can streamline the development process. | For mobile development, tools like Xcode for iOS, Android Studio for Android, and cross-platform frameworks like Flutter and React Native are commonly used. Xcode and Android Studio are free but require investment in specific hardware (macOS for Xcode). Cross-platform tools can reduce development time and cost but require expertise in managing and optimizing cross-platform code. The impact on the development team includes the need for diverse skill sets and the ability to manage different development environments. |

## 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: I recommend using Linux as the primary server operating platform. Linux is known for its scalability. This is important for expanding to various computing environments. Its flexibility supports a wide range of hardware and software. This will make it easier to integrate with different platforms. Additionally, Linux is open-source and free to use, reducing the overall cost for The Gaming Room. With a large community of developers constantly improving and supporting Linux, it makes it a reliable choice.
2. Operating Systems Architectures: The architecture of the Linux operating system is designed to be efficient. At its core is the kernel, which operates in kernel mode and is responsible for managing system resources. The kernel ensures that the system runs smoothly by handling all low-level tasks. Above the kernel, the system includes various user space processes, which are managed separately to ensure stability and security. This modular design allows for greater flexibility.
3. Storage Management: For storage management, I recommend using the ext4 file system. It is well-supported by Linux and offers great performance and reliability. The ext4 file system supports large volumes and file sizes. This ensures that the Draw It or Lose It software can handle large amounts of data efficiently and reliably, providing a smooth user experience.
4. Memory Management: Linux employs both physical and virtual memory, with a paging mechanism that allows for the efficient allocation of memory. This includes the use of swap space to extend physical memory. This ensures that the system can handle large applications and multiple users simultaneously. Additionally, Linux supports various memory management algorithms, which helps in managing the cache and ensuring optimal performance.
5. Distributed Systems and Networks: To enable Draw It or Lose It to communicate between various platforms, a distributed software architecture can be employed. This can be achieved using microservices. Each component of the game operates as an independent service that communicates with other services over a network. This approach ensures that the game is scalable and can handle failures gracefully. Each service can be deployed and updated independently. The network connecting these services can use protocols such as HTTP/HTTPS for communication, with load balancers to distribute the load evenly across different servers. This setup ensures high availability and reliability, even in the face of connectivity issues or outages.
6. Security: Security is a critical aspect for The Gaming Room, especially when handling user information. Using secure communication protocols such as HTTPS ensures that data transmitted over the network is encrypted. Additionally, implementing authentication ensures that only authorized users can access sensitive data. On the server side, employing firewalls and intrusion detection systems can help protect against unauthorized access. Regular security updates and patches should be applied to all systems to address vulnerabilities promptly. By using the security capabilities of the recommended Linux platform, The Gaming Room can ensure that user information is protected effectively.