

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 | 9/28/2025 | Raegan Haight | *Updated Executive Summary, Design Constraints, and Domain Model* |
| 2.0 | 10/12/2025 | Raegan Haight | *Updated Evaluation* |
| 3.0 | 10/26/2025 | Raegan Haight | *Updated Recommendations* |

## [Executive Summary](#_sbfa50wo7nsh)

*The Gaming Room wants to expand upon their application, Draw It or Lose It, to be developed as a multi-platform web-based version of the game. Draw It or Lose it is currently only available on Android and is based on the 1980s game show Win, Lose or Draw, where teams compete to guess what is being drawn. The online platform instead renders images from a library of stock drawings as clues with each game consisting of four rounds lasting one minute each. Clues should be fully rendered at 30-seconds as the teams take their turns. If the active team does not guess correctly within the time limit, the remaining teams should be able to guess once each within a 15-second time limit.*

## Requirements

1. *The game will be translated to a web-based distributed environment and serve multiple platforms.*
2. *A game will have the ability to have one or more teams involved.*
3. *Each team will have multiple players assigned to it.*
4. *Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.*
5. *Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers of each instance of a game, team, or player.*

## [Design Constraints](#_2et92p0)

1. Must utilize appropriate programming languages/practices necessary to support multiple platforms and operating systems as well as adhere to online publishing guidelines.
2. Must support one or more teams per game with multiple players assigned to each.
3. Code must ensure game and team names are all unique identifiers and prevent multiple-instances/duplicate instances of each game.

## [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 following UML model demonstrates the attributes and relationships of the classes within the Draw It or Lose It game. For example, *Entity* is the base class that holds the common attributes and behaviors that *Game, Team,* and *Player* inherit from. This illustrates the object-oriented programming (OOP) principle of inheritance that promotes organization and simplicity. The *0…\** displays zero-to-many association relationships between *GameService* and *Game, Game* and *Team,* and *Team* and *Player.* In other words, *GameService* can hold zero-to-many instances of *Game*, *Game* zero-to-many instances of *Team*, and *Team* zero-to-many instances of *Player*. This allows for multiple players and teams per game. Modeling these relationships also ensures organization and a clear understanding of how the classes interact with one another. Finally, the *ProgramDriver* class holds the main function of the game and is dependent upon the *SingletonTester* class to retrieve the data necessary to run the game as well as to prevent multiple instances of the *GameService* class.

**"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 is derived from Unix, known for its security, and offers server-based deployment. It is well-established - the second most used operating system. Not only this, but there are also fewer active users which results in fewer virus attacks. Supports Windows file system formats and can run Windows. Low graphics capability and less software availability. | Key strengths include stability and reliability. Designed to handle large workloads and run for extended periods without reboots. Popular choice for servers and supercomputers. Increased user flexibility and customizability. Lack of standardization leading to potential issues with compatibility. Open source and free to use. | Support for all hardware. Great ease of use between versions making transitions smooth. Best suited for game and software developers due to large audience and better compatibility features. High amount of virus/hacker attacks. Frequent reboots necessary with slow performance. More affordable and accessible. | Android is an open-source platform with increased customization capabilities. It offers highly effective multitasking capabilities and file system access. Compatibility issues, security concerns, limited software support. iOS is known for robust security, optimized hardware and software integration, and reliability/stability. On the other hand, limited customization, closed ecosystem, and costly hardware. |
| **Client Side** | Optimized for Apple hardware but limited compatibility raises costs. Furthermore, this also results in less hardware specifications. Must follow Apple guidelines for publishing software applications. | Steep learning curve, limited software and hardware compatibility requires compatibility layers or other alternatives – raising increasing cost/time. However, open sourcing makes it much more affordable | Most graphics software is paid and requires purchase/monthly costs to use. Lacking in technical support, could increase risk in costs/time. Standard OS for most commercial software – well-known and understood. | Requires expertise across multiple mobile device operating systems such as Android OS and iOS. Less costs associated with Android development vs iOS development. Must adhere to Apple/Android guidelines. |
| **Development Tools** | Ideal for Objective-C or Swift development language support. Popular IDEs include Visual Studio, Xcode, PyCharm. | Commonly used languages include Java, C, and C++. Popular IDEs include IntelliJ IDEA, Eclipse, Visual Studio Code | Relevant programming languages are Python, JavaScript, and C#. Popular IDEs include Visual Studio, PyCharm, and Eclipse. | A few popular programming languages for mobile devices are Swift, Kotlin, and Java. Relevant IDEs include Android Studio, Xcode, and Visual Studio. |

## Recommendations

1. **Operating Platform**: The best suited operating platform that will allow for the expansion of Draw It or Lose It to other computing environments is Linux. This is due to the low cost, flexibility, open-source software, scalability, low infrastructure requirements, and effective security of Linux OS. For example, Linux servers utilize open-source software that require no licensing fee and offer more flexibility (Zivanov, 2025). Furthermore, Linux “is considered foundational to cloud computing environments” that would contribute greatly to the maintenance, scalability and expansion of Draw It or Lose It (Flinders & Smalley). Another advantage is the low resource consumption that benefits clients and the low infrastructure requirements that display “excellent performance rates” (Zivanov, 2025). Not only this, but also the current platform for Draw It or Lose It is Android, the mobile OS for Linux, which would further aid in the expansion of the application. Moreover, Linux is “highly secure against malware and cyber threats” and is known for this security and reliability (Zivanov, 2025).
2. **Operating Systems Architectures**: The Linux OS was derived from Unix and depends “heavily on the Linux kernel, a free, open-source computer program that gives a user control over the hardware and software layers of a computer system” (Flinders & Smalley). Main components include the Application, Shell, Kernel, Hardware, and Utilities (GeeksforGeeks, 2023). The kernel is one of the most significant components, located between the shell and hardware, responsible for memory management, resource allocation, process management, application interaction, and security (GeeksforGeeks, 2023). In other words, the kernel aids in hardware and system communication, functions, and security. Furthermore, Linux is widely supported by serverless architecture and cloud providers such as AWS Lambda, Azure, and Google Cloud Functions.
3. **Storage Management**: Storage management is essential for Draw It or Lose It and should be supported by incorporating serverless architecture and investing in a cloud provider for lower costs, application size optimization, improved library scaling, and decreased labor. For example, pricing for serverless is based on the number of executions which reduces costs compared to the maintenance costs required for traditional architecture. Additionally, the servers are dynamically managed by the cloud provider which lowers costs further and allows for more time to dedicate to growth and development. Not only this, but the image library would also be stored on the cloud server for simpler scalability and to reduce the size of the initial application – improving storage. Amazon Web Services (AWS) is a great serverless option as it is the largest cloud provider in the market, and it emphasizes security, reliability, and performance.
4. **Memory Management**: Linux employs advanced memory management techniques such as virtual memory, paging, kernel memory allocation, and caching. Virtual memory creates the illusion of more physical space, or RAM, which allows for more processes to run and more efficient utilization of memory. Moreover, paging divides physical and virtual memory into pages that the kernel can use to most effectively allocate resources between the RAM and disk (Yogesh, 2024). Lastly, caching “stores frequently accessed file pages in memory to speed up disk reads” and “improve overall system performance by reducing the need for repeated disk accesses” (Yogesh, 2024).
5. **Distributed Systems and Networks**: Draw It or Lose It would work best across platforms through the utilization of a distributed system. This separates the game’s main components onto multiple servers, or services, that work together through the internet. As a result, updates and scaling becomes much easier to handle without disrupting the entire game. An important aspect of distributed systems is load-balancing to ensure that traffic is spread evenly, and the servers are not overwhelmed. Furthermore, managing traffic helps to prevent lagging and connectivity issues. These services can communicate through APIs, which support cross-platform communication and data exchange. Also, the communication should be secured through a protocol such as HTTPS that encrypts data to protect the integrity and confidentiality of the information.
6. **Security**: As discussed above, security is a major benefit of a Linux operating system. Linux is derived from Unix, which has well-established security measures, and its servers are known to experience less malware, cyber, and other security threats compared to others. Furthermore, encryption methods like HTTPS ensure that all data is unreadable to hackers/security threats and maintains privacy. Incorporating multi-factor authentication measures is another way to better protect users by verifying credentials and preventing unauthorized individuals from accessing data. Moreover, having a cloud-based server often involves firewalls, encryption, and other security features/management.

References

Dubey, S. (2024). *Linux vs. Windows vs. Mac: An In-Depth Comparison of OS*. [online] Wbcom Designs. Available at: <https://wbcomdesigns.com/linux-vs-windows-vs-mac/>.

Flinders, M., & Smalley, I. (2025, March 26). *Linux Servers*. Ibm.com. https://www.ibm.com/think/topics/linux-servers?utm\_source=chatgpt.com

GeeksforGeeks. (2023, September 25). *Architecture of Linux Operating System*. GeeksforGeeks. https://www.geeksforgeeks.org/linux-unix/architecture-of-linux-operating-system/

Nimit Suwannagate (EThaiZone (2024). *Windows or Mac: Which is Better for Software Development?* [online] Medium. Available at: https://medium.com/@ethaizone/windows-or-mac-which-is-better-for-software-development-b22321f18f9a.

Rehman, J. (2019a). *Advantages and disadvantages of windows operating system*. [online] IT Release. Available at: https://itrelease.com/2019/07/advantages-and-disadvantages-of-windows-operating-system/#google\_vignette.

Rehman, J. (2019b). *What are advantages and disadvantages of macOS*. [online] IT Release. Available at: https://itrelease.com/2019/10/what-are-advantages-and-disadvantages-of-macos/#google\_vignette.

Saha, S. (2023). *What are the Top Advantages and Disadvantages of Linux?* [online] Scaler Topics. Available at: https://www.scaler.com/topics/advantage-and-disadvantage-of-linux/.

Virtubox (n.d.). *Advantages and Disadvantages of Android & iOS*. [online] VirtuBox Blog. Available at: <https://www.virtubox.io/blog/advantages-and-disadvantages-of-android-and-ios>.

Yogesh. (2024, November 15). *Linux — Memory Management - Yogesh - Medium*. Medium. https://lambdafunc.medium.com/linux-memory-management-2c13b5e386f1

Živanov, S. (2025, October 20). *Linux vs. Microsoft Windows Servers, the Ultimate Showdown*. PhoenixNAP Blog. https://phoenixnap.com/blog/linux-vs-microsoft-windows-servers?utm\_source=chatgpt.com