

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 | 07/16/2025 | Wyatt Moore | Added Executive Summary, Design Constraints, and Domain Model |
| 2.0 | 08/03/2025 | Wyatt Moore | Added development requirements. |
| 3.0 | 08/16/2025 | Wyatt Moore | Added sources to development requirements and adjusted analysis based on feedback.  Added 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)

“The Gaming Room” has contracted our firm to develop a web-based version of their current gaming app, Draw It or Lose It, that will be available on multiple platforms. In the game, teams compete to guess a phrase, title, or object based on an image from a library of stock drawings that will be gradually rendered over a 30 second interval. If the active team fails to guess the answer, other teams are given 15 seconds each to solve the puzzle.

The client has stated that the developed game must be able to accommodate one or more teams, with each team being able to have multiple players assigned to it. Each instance of a game must have a unique name and each team belonging to an individual game must also have a unique name. Lastly, the game must be developed to only allow one instance of the game to exist in memory at a given time.

## 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)

* The Web-based game must support multiple user roles such as Developers, QA testers, IT members, and end users, with each role requiring a separate environment.
* The Web-based game will need a separate secure authentication system, requiring users to enter a unique username and password.
* Rendering images for the Web-based game will come with increased server memory and storage demands.
* Developing a Web-based game to accommodate a physical mouse and keyboard will require a redesign of the user interface to accept new input methods.

## [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 Entity class is a parent for the Game, Team, and Player classes, holding shared private attributes id and name, and proving public getter methods to control access these attributes. The three child classes inheriting from the Entity class reduces code duplication and maintains consistency across the three child classes. The Entity class also includes a private constructor with no arguments to prevent objects from being created without attributes. The Entity class demonstrates encapsulation by making attributes private then providing the public getters to access them, inheritance by acting as a parent class, abstraction by defining common functionality without exposing unnecessary internal details, and polymorphism by overriding the toString method.

The Game class is a child class, inheriting the id and name attributes from the Entity class. The class maintains a private list of teams including their name and id. The addTeam method is responsible for adding teams to the game. This class demonstrates inheritance by inheriting the id and name attributes from its parent class, Entity, encapsulation by keeping the team list private, abstraction with the use of public methods to hide internal functionality, and polymorphism by overriding the toString method. The Game class also has a zero to many association with Team, showing that each game instance can contain multiple teams.

The Team class is also a child class, inheriting the id and name attributes from the Entity class. The class maintains a private list of players including their name and id. The Team class also includes the addPlayer method which is responsible for adding players to a team. This class demonstrates inheritance by inheriting the id and name attributes from its parent class, Entity, encapsulation by keeping the player list private, abstraction with the use of public methods to hide internal functionality, and polymorphism by overriding the toString method. The Team class also has a zero to many association with Player, showing that each team instance can contain multiple players.

The Player class is also a child class, inheriting the id and name attributes from the Entity class. This class represents individual users in the game. This class demonstrates polymorphism by overriding the toString method.

The GameService class is the central hub of the application. It maintains a private list of all games including their name and id. The class is also responsible for generating unique ids for Games, Teams, and Players. The id are generated through private counters accessed with public getter methods. The GameService class also includes a singleton pattern to only allow for one instance of GameService to exist in memory at a time. This single instance ensures all games, teams, and players get unique id’s. The addGame method is used to create games. GameService has a zero to many association with Game, meaning the single instance of GameService can contain multiple games. This class demonstrates encapsulation by keeping the id counters private and then providing public getter methods to access them, and abstraction by using controlled public methods to access and modify internal data.

The ProgramDriver class contains the main method, which is used to run the program, and the SingeltonTester class contains the testSingleton method which is used to test the functionality of the singleton pattern in the GameService class.

The association between the Game class and Team class is labeled as a zero to many, showing that a single game will be able to contain one of more teams. The association between the Team class and Player class is also labeled as zero to many, showing that a single team will be able to contain one or more players. The private lists in the Team and Game class will be used in the addTeam and addPlayer methods to enforce maintain unique game and team names in the program. The singleton pattern in the GameService class guarantees there will be only one instance of the game in memory at a given time, also the private counters in GameService, nextGameId, nextPlayerId, and nextTeamId, will be used to generate unique identifiers for each game, team, and player.

**"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 platforms offer a secure, UNIX-based development environment with native support for iOS application development (*What is Xcode and how to use it?*). However, Apple doesn’t offer a dedicated server edition of macOS, which limits scalability and makes Mac impractical for the Draw It or Lose It application. Mac hardware is also expensive, and Apple’s licensing model requires macOS servers to run on Apple hardware, increasing project costs. | Linux is highly suited for server-side development and deployment due to it being open-source, low resource usage, and high scalability. It offers a wide array of tools and server environments for web application hosting (*Linux for developers: Tools, tips, and why it's the preferred OS for development* 2024). Linux would work particularly well with *Draw It or Lose It*, which requires scalable and cost-effective hosting for many concurrent users. The main drawback is its steeper learning curve, which could increase development time for teams unfamiliar with Linux. | Windows Server offers a user-friendly interface and strong compatibility with .NET-based tools and services. It supports web hosting platforms like IIS and integrates easily with Microsoft technologies, which could benefit some aspects of Draw It or Lose It (Anandmeg, What is the visual studio Ide?). However, Windows requires licensing fees and tends to be more resource intensive. | Mobile Devices do not support server-based deployment and cannot be used to host web applications. |
| **Client Side** | For Mac clients, the application must be compatible with Safari as well as other browsers like Chrome and Firefox. Apple’s hardware and operating system are more consistent than Windows, but developers still need to account for different screen sizes and display resolutions. Touchpad gestures and other macOS-specific input features may require additional UI considerations. | Linux clients typically use open-source browsers like Firefox or Chromium. Compatibility with these browsers requires developers to test the application on different Linux distributions and environments, which can vary more than Windows or macOS. While Linux is not the most common desktop platform, its users often expect high performance and compatibility with open standards. Developers should ensure that the application does not rely on proprietary browser features and performs well in resource-constrained systems. | Developing the application for Windows clients requires compatibility with popular browsers like Chrome, Edge, and Firefox. Since Windows supports a wide variety of browsers and hardware configurations, developers must test the application across different screen resolutions and devices to maintain consistent user experience. Windows is used in desktop environments, so the application must handle keyboard and mouse input efficiently. | For mobile clients, the application will be available for download via the Google Play Store and Apple App Store, requiring development of native or cross-platform mobile applications. These apps must be optimized for small screens, touch input, and varying device performance. Developers will need to follow Android and iOS platform guidelines for app submission, security, and UI standards. In addition to native apps, the application must be compatible with mobile web browsers so that players can access the game directly without installing an app. This requires responsive design, lightweight assets, and thorough testing across popular browsers like Chrome and Safari to provide consistent and reliable gameplay experience. |
| **Development Tools** | For Mac desktop development, the most commonly used IDE is Xcode, which is specifically designed for Apple platforms. The primary programming language is Swift, although Objective-C may also be used in some cases. While Xcode itself is free, development must be done on Apple hardware, increasing costs for the team (*Xcode vs VSCode: Everything you need to know* 2025). Developing the application for Mac will require developers with experience in Apple’s tools and environment to ensure compatibility and proper functionality. | Linux development commonly uses Eclipse, VS Code, or IntelliJ IDEA, and languages such as Java, Python, or JavaScript for server-side and web-based development (*Linux for developers: Tools, tips, and why it's the preferred OS for development* 2024). Linux tools are typically open source and free, reducing licensing costs. Linux’s flexibility and support for scripting and automation allow for scalable deployment and testing. However, developers need Linux expertise, and teams may require additional training. | Windows developers frequently use Visual Studio, especially for .NET-based development with C#. Visual Studio has both free and paid versions, with advanced features behind a license (Anandmeg, *What is the visual studio Ide?*). Windows development tools are well-documented and offer good integration with Microsoft services, reducing development complexity for Windows-based applications. | Mobile devices lack the tools and support to develop web-based applications. |

**References**

29, E. K. E. A., Ekren, E. K., 29, U. A., More posts by this author        Esat Kemal EkreniOS Developer at Netguru, & Netguru, iOS D. at. (n.d.). *What is Xcode and how to use it?*. Digital Acceleration Company. https://www.netguru.com/blog/what-is-xcode-and-how-to-use-it

Dotnet-Bot. (n.d.). .NET implementations - .NET. .NET | Microsoft Learn. <https://learn.microsoft.com/en-us/dotnet/fundamentals/implementations>

*Xcode vs VSCode: Everything you need to know*. BrowserStack. (2025, May 15). https://www.browserstack.com/guide/xcode-vs-vscode?utm

Anandmeg. (n.d.). *What is the visual studio Ide?*. What is the Visual Studio IDE? | Microsoft Learn. https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022

&Ccedil;intay, E. (2024, February 25). *Linux for developers: Tools, tips, and why it’s the preferred OS for development*. Medium. https://medium.com/@Chinacolt/linux-for-developers-tools-tips-and-why-its-the-preferred-os-for-development-ce7cb3cb218f

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>

Windows Server provides the best environment for expanding Draw It or Lose It into multiple computing environments. Its broad compatibility with development frameworks, testing tools, and deployment pipelines makes it the best option for developing this application. Windows offers support for containerization and virtualization, enabling developers to simulate Mac, Linux, and mobile environments during testing without leaving the Windows ecosystem (Vrapolinario, 2023). This reduces the need for separate dedicated hardware during the development phase. Its integration with popular build and deployment tools, such as Visual Studio, and cross-platform frameworks like .NET streamlines the process of creating a single codebase that can be adapted for various platforms (Dotnet-Bot, 2024). Additionally, the strong debugging and monitoring capabilities built into Windows Server can help maintain consistent performance and stability across all client platforms (Robinharwood, 2025).

1. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

Windows Server is Microsoft's enterprise-grade server platform that powers applications and services across all environments. Its architecture is built on a 64‑bit hybrid Windows NT kernel with a clear separation between user-mode, for application code and services, and kernel-mode, for core OS functions (Wiki, Hybrid kernel)This design makes Windows Server exceed at process isolation, virtual memory management, and efficient I/O operations, which are all important for rendering game assets quickly. Windows Server also supports mixed environments, allowing Unix-like workloads, such as Linux containers, to run side by side, which simplifies cross-platform testing and compatibility during development (Vrapolinario, 2023.).

1. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>

Google Cloud Storage is my recommendation for storage management because it provides a single API endpoint for all stored data (Product overview of cloud storage | google cloud 2025). This allows every platform, Mac, Linux, Windows, and mobile, to pull from the same unified resource pool without requiring separate storage implementations. As the game expands to new environments, this centralized approach ensures instant compatibility and consistent access speeds for all players. Additionally, the cloud’s scalability enables the storage system to grow seamlessly, whether by adding new images, expanding game functions, or storing more player profiles, without altering the core storage logic (Product overview of cloud storage | google cloud 2025).

1. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>

Windows Server uses paging and memory compression to efficiently manage system memory and maintain performance under heavy workloads. Paging allows the operating system to treat available memory as a combination of physical RAM and virtual memory stored on disk. This ensures that even when the game is running multiple instances and memory demand is high, less active data can be swapped to disk while keeping active game logic and image rendering in RAM (TutorialsPoint, n.d.). Memory compression works alongside paging by compressing inactive pages before they are written to disk, allowing more data to remain in physical memory (Wikipedia contributors, 2025). For Draw It or Lose It, this means that images from the library can be loaded, rendered, and transitioned between rounds with minimal delay, even when several games are active at once. Together, paging and memory compression reduce lag, prevent stuttering during image transitions, and allow for smooth, consistent gameplay experience across all supported platforms.

1. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>

To enable Draw It or Lose It to run across multiple platforms, the application will be built as a distributed client–server system where Mac, Linux, Windows, and mobile clients communicate with a centralized Windows Server backend using a RESTful API over HTTPS (What is rest? 2025). The game logic, user authentication, and image assets will reside on the server, while clients handle only the user interface and data exchange. Load balancers can be used to distribute requests among application servers to increase availability during peak hours and enable failover if a server fails (Load balancing (computing) 2025).To mitigate disruptions, the client-server design can include graceful degradation, such as local client caching during short connectivity interruptions, and adheres to a stateless server model so that game sessions can be resumed on any available server (GeeksforGeeks, Graceful degradation in Distributed Systems 2025).

1. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>

To protect user data on the Windows Server platform, all communications between clients and the server should use HTTPS with TLS encryption, so that data in transit, such as login credentials, cannot be intercepted or altered. On the server, passwords should never be stored in plain text. Instead, they must be hashed using a strong algorithm like bcrypt or Argon2, which are resistant to brute-force attacks (Password storage cheat sheet). Role-based access control can be used to limit user permissions so that players and administrators only have access to the features they need. The Windows Server environment also provides built-in security features such as Windows Defender, firewall configurations, and Active Directory integration for secure authentication (Windows firewall 2025) (Active directory 2025). Additional safeguards should include multi-factor authentication for administrative and player accounts, regular security patches, and monitoring tools like Windows Security Log to detect suspicious activity across platforms (Windows security log 2023).

**References**

Robinharwood. (2025.). What is windows server?. Microsoft Learn. https://learn.microsoft.com/en-us/windows-server/get-started/overview

Google. (2025). Product overview of cloud storage  |  google cloud. Google. https://cloud.google.com/storage/docs/introduction

Wiki, C. to M. (n.d.). Hybrid kernel. Microsoft Wiki. https://microsoft.fandom.com/wiki/Hybrid\_kernel   
  
Dotnet-Bot. (2024). .NET implementations - .NET. .NET | Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/fundamentals/implementations

Vrapolinario. (2023.). Lift and shift to containers. Lift and shift to containers | Microsoft Learn. https://learn.microsoft.com/en-us/virtualization/windowscontainers/quick-start/lift-shift-to-containers

TutorialsPoint. (n.d.). Memory management in operating systems. Tutorialspoint. https://www.tutorialspoint.com/operating\_system/os\_memory\_management.htm

Wikipedia contributors. (2025, July 15). Virtual memory compression. In Wikipedia, The Free Encyclopedia. Retrieved 15:16, August 10, 2025, from <https://en.wikipedia.org/w/index.php?title=Virtual_memory_compression&oldid=1300595696>

What is rest?. REST API Tutorial. (2025, April 1). https://restfulapi.net/

Wikimedia Foundation. (2025, August 6). Load balancing (computing). Wikipedia. https://en.wikipedia.org/wiki/Load\_balancing\_(computing)   
  
GeeksforGeeks. (2025, July 23). Graceful degradation in Distributed Systems. https://www.geeksforgeeks.org/system-design/graceful-degradation-in-distributed-systems/

Password storage cheat sheet. Password Storage - OWASP Cheat Sheet Series. (n.d.). https://cheatsheetseries.owasp.org/cheatsheets/Password\_Storage\_Cheat\_Sheet.html

Wikimedia Foundation. (2025a, June 15). Windows firewall. Wikipedia. https://en.wikipedia.org/wiki/Windows\_Firewall   
  
Wikimedia Foundation. (2025c, August 16). Active directory. Wikipedia. https://en.wikipedia.org/wiki/Active\_Directory   
  
Wikimedia Foundation. (2023, November 16). Windows security log. Wikipedia. https://en.wikipedia.org/wiki/Windows\_Security\_Log