

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  1.1  2.0 | 11/16/24  11/30/24  12/14/2024 | Marissa Castellane  Marissa Castellone  Marissa Castellone | Initial draft of the software design document, including the executive summary and design constraints.  Added information for the evaluation section  .  Updated recommendations section. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has asked that their well-liked mobile game, Draw It or Lose It, be made available online. Teams compete to guess drawings made from a library of stock photos in this game, which is presently accessible as an Android app. Teams have a set amount of time to correctly guess the drawing; if no team can, other teams get a chance to solve the puzzle.

## Requirements

• There will be the option for one or more teams to participate in a game, and each team will be assigned multiple players.

• To enable users to verify if a name is already being used when selecting a team name, game and team names need to be distinct.

• The game can only have one instance running in memory at once. Making distinct IDs for every occurrence of a game, team, or player will help achieve this.

## [Design Constraints](#_2et92p0)

* The image library available for the game will not be unlimited, meaning the game will be restricted to a finite number of images that can be used during gameplay.
* The game includes strict timing for each round (e.g., a one-minute timer for drawing and 15-second timers for guesses). These timers must function accurately across all clients and keep the game synchronized in real-time.

## [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)

Even though the System Architecture View is not needed for this project, it's vital to remember that it might be in the future. The interaction between the game's frontend, backend, database, and real-time communication should all be part of the logical topology. To ensure that the structure and performance needs of the system are clearly understood, it should also explain how data moves between clients, the server, and the database. This entails specifying the physical elements, such as load balancers or cloud architecture, as well as the tiers, such as the presentation, application, and data layers.

**"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** | Since Mac is more expensive and less flexible than Linux or Windows, it is usually not the first option for hosting web applications. MacOS is frequently viewed as more of a development computer than a production server, even if it can host a web-based software application utilizing tools like Apache, Nginx, or Docker. Stability, great developer support, and a solid UNIX-based architecture are some of its advantages. But compared to Linux, it uses more resources and scaling is more difficult.  **Update**:  Although macOS is more frequently utilized for development and small-scale settings, it does allow web hosting. Although MacOS Server has built-in web hosting capabilities (such as Apache, Nginx, etc.), it is not scalable enough for large, traffic-heavy online applications.  Not made for hosting on a wide scale.  Runs macOS, Windows, and Linux apps side-by-side. Since Macs are expensive, overall prices go up. | Most people agree that Linux is the greatest option for hosting web-based applications, especially for open-source web stacks. Since Linux uses system resources efficiently, it is extremely adaptable, economical (because it is frequently free), and able to handle heavy traffic loads.  Compared to Windows or macOS, security may be more complicated to set up and maintain in production settings.  **Update**:  A popular choice for server-side installations is Linux. Numerous server configurations are supported, including those for Apache, Nginx, MySQL, and container technologies like Docker. Linux is well known for being flexible and scalable when it comes to web hosting.  Supports large-scale applications with ease. Some proprietary software has compatibility problems. Linux itself has no license fees. However, there can be a cost associated with enterprise support choices (like Red Hat). Expenses are determined by paid support or cloud hosting infrastructure. | The primary benefit of Windows is that it's compatible with many commercial tools and programs. Teams that are already accustomed to Windows' environment can also find it easy to utilize.  In addition to being costly (license costs) and resource-intensive, it might be vulnerable to security flaws if improperly maintained. In comparison to Linux, it also provides less flexibility for specific kinds of open-source development.  **Update:**  With IIS (Internet Information Services) and complete integration with enterprise solutions (such as Active Directory and SQL Server), Windows Server offers a reliable web hosting option. Although it is more resource-intensive, it supports large-scale applications.  Perfect for business settings and Outstanding for software that is based on Microsoft (e.g.,.NET, SQL Server).  Requires   software (like SQL Server) and operating system licenses. Depending on the number of cores, users, or processors, Windows Server licenses might be costly. | | Lightweight web server applications can be run on mobile devices for very small-scale applications or for local testing. Mobile devices are only used as endpoints for online apps, and cloud solutions are typically used for this purpose.  Although mobile operating systems (iOS and Android) are made for contexts with limited resources, they can't be used as reliable servers.  **Update**:  Although web apps can technically run on mobile devices, large-scale production hosting is not a good fit for them. Mobile solutions are frequently cloud-based and are best suited for tiny, low-traffic applications.  Has the ability to put up simple web hosting solutions fast.  The processing power, memory, and storage capacity of mobile devices are constrained.  While some mobile web server apps are free or extremely inexpensive, using cloud-based solutions (such cloud hosting and data transport) may come with recurring fees. |
| **Client Side** | Cost - Compared to other platforms, macOS development frequently necessitates the use of Mac hardware, which can be expensive.  Developing for macOS can be time-consuming if the team is not familiar with the platform.  macOS development requires familiarity with Apple's proprietary tools and programming languages,  **Update**:  Cross-Browser Testing: Verify compatibility with other widely used browsers (such as Chrome, Firefox, and Safari), which is macOS's default browser.  Typically, macOS development calls for Mac hardware, which might be more costly than, macOS programming calls on Mac hardware, which might be pricey in comparison to other platforms. Additional expenses for macOS development tools could include buying software tailored to Apple products for other platforms. Additionally, using macOS programming tools could cost more than just buying Apple-specific software.  If the team is unfamiliar with macOS, developing for it can take a lot of time, particularly if the software needs native integration with Apple-specific tools and APIs (such as Swift, Objective-C).  Programming languages (Swift/Objective-C) and Apple's proprietary tools are required of developers; these might complicate the process, particularly for cross-platform programs. | Since Linux is free and open-source, developers find it to be a desirable alternative. Usually, the only expenses are those related to expensive tools (such specialist databases or IDEs) and sometimes server or cloud infrastructure.  Linux can be a fast and efficient development environment, but it does require developers to be familiar with the system, especially when working with distributions, package managers, and various tools.  Linux developers must be comfortable working in a terminal-based environment and be familiar with the specific distributions used by the target audience  **Update**:  Operating System: Windows or macOS are usually used for desktop-based application development, however Linux is frequently used in server contexts and for development.  From a financial standpoint, Linux is a desirable alternative because it is open-source and free. The only possible expenses are associated with cloud server infrastructure and premium products like specialist databases or IDE subscriptions.  Web-based apps can be developed quickly and effectively using Linux, but teams who are not familiar with the platform may find it time-consuming, especially if they are working with distributions, package managers, or tooling.  Developers need to know the Linux distribution being used for the project and feel at ease working in a terminal-based environment. | Windows development necessitates a paid license for both the operating system and, if necessary, the development tools (like Visual Studio).  Developing for Windows is relatively straightforward given the vast number of tools, libraries, and frameworks available.  Expertise in Windows programming usually centers on Microsoft products like Visual Studio, C#, and.NET. The Windows OS and related libraries, APIs, and tools should be known to developers.  **Update**:  Windows is frequently used for web development as well as desktop program development. It works nicely with Microsoft tools like Visual Studio and supports a wide range of tools.  Both the operating system and perhaps the programming tools, such Visual Studio, must be licensed for Windows development. On the other hand, smaller teams or individual engineers can use free tools like Visual Studio Community Edition.  With so many tools, libraries, and frameworks available, Windows development is comparatively simple.  Windows programming, which frequently uses Microsoft technologies like Visual Studio, C#, and.NET, requires developers to be adept. | Purchasing mobile devices (for testing on actual hardware) and possibly software licenses for tools are extra expenses associated with mobile development.  When handling larger applications, lower hardware specifications might make development and user usage challenging.  Proficiency in the corresponding platforms is necessary for mobile development.  **Update**:  Additional expenses for mobile devices (for testing) and software licensing for development tools (such as Xcode for iOS and Android Studio for Android) may be incurred during the mobile development process. Licenses for frameworks (like Flutter and React Native) could also be required if cross-platform technologies are being used.  Mobile development can be time-consuming, particularly when creating for both iOS and Android, and need for expertise in the corresponding mobile platforms. It also requires time to handle cross-platform compatibility (such as screen sizes and navigation paradigms).  It is necessary to be proficient in the native mobile development environments. While iOS programming necessitates familiarity with Xcode, Swift, and the iOS SDK, Android development demands proficiency with Android Studio, Java/Kotlin, and the Android SDK. | |
| **Development Tools** | Swift (for native iOS/macOS apps), Objective-C (for older iOS/macOS apps), and JavaScript (for web-based apps) are popular programming languages on macOS.  **Update**:  For creating native iOS/macOS apps, developers will need to be familiar with Xcode, Swift, and Objective-C. They can leverage cross-platform web frameworks (React, Vue, etc.) and JavaScript for creating web applications.  macOS hardware: Apple products have a high initial cost. | Programming languages including Python, JavaScript, Java, C++, Ruby, PHP, and Go are widely used on Linux.  A popular platform for open-source development is Linux.  **Update**:  Linux is appropriate for server-side development, open-source projects, and online applications since it provides a wide variety of programming languages (such as Python, Java, C++, Ruby, etc.). Development teams must feel at ease utilizing Linux-based IDEs, package managers, and terminal commands when working in a Linux environment.  Atom, Sublime Text, and Visual Studio Code are all free IDEs. Advanced features could require premium licenses. | Programming languages that are frequently used on Windows include C++ (for system-level programming), JavaScript (for web applications), and C# (for.NET).  **Update:**  Development teams must be proficient with the C#,.NET, and ASP.NET components of Microsoft's ecosystem, particularly when making desktop Windows apps.  Visual Studio: Individuals can use the Community Edition for free, while the Professional and Enterprise editions need a license. Some IDEs require a license also which can be costly. | | Swift is the main programming language used to create iOS apps, whereas Kotlin and Java are used to create Android apps.  **Update:**  Cross-Platform Development: To cut down on the overhead of maintaining distinct native codebases for iOS and Android, teams must decide whether to use React Native (JavaScript) or Flutter (Dart).  Xcode is free but only available on macOS.  Flutter and React Native are free, but there might be a cost for third-party libraries or specialized tools. |

## 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**: The suggested operating platform for development and deployment is Linux-based server environments for backend operations, combined with cross-platform client solutions like Web (HTML5, JavaScript), and React Native for mobile applications, given the client's wish to extend Draw It or Lose It beyond the Android platform to support multiple platforms (likely including web-based and possibly iOS).

* **Update**: Since The Gaming Room wants to make Draw It or Lose It available on more platforms, a Linux-based server environment is the suggested operating system for backend operations. This platform is perfect for managing the performance, scalability, and adaptability required to support a cross-platform, multiplayer game. Scaling resources as the user base increases is made simple by Linux's strong, high-performance infrastructure and superb cloud deployment capabilities. It is ideal for overseeing the backend infrastructure, which includes databases, gaming servers, and real-time networking, guaranteeing seamless gameplay on various platforms.

1. **Operating Systems Architectures**: The Windows graphical operating system facilitates the seamless and interchangeable use of files, software, games, films, coding, the internet, and many other capabilities.

* Update: The Windows operating system architecture combines a graphical user interface (GUI) that makes it simple to interact with apps and a strong kernel for resource management to create a smooth and intuitive user experience. Large applications may be handled effectively because to its \*\*virtual memory management\*\* and \*\*NTFS\*\* file systems, which facilitate seamless file sharing between programs. Windows works well with cloud services for backend scalability and includes robust development tools, such as \*\*DirectX\*\* for game graphics and multimedia. Because of its architecture, it is the perfect platform for both consumers and developers, guaranteeing performance and versatility across a range of applications, such as software development, gaming, and video.

1. **Storage Management**: Windows 10 and 11 comes with **Storage Sense**, a built-in feature that monitors disk usage and automatically deletes unneeded files to free up space, ensuring that storage is optimally managed.

* Update: The integrated Storage Sense feature offers an effective storage management system for the suggested Windows 10 and 11 platforms, guaranteeing optimal disk usage with automatic storage monitoring and management. To clear up space, Storage Sense periodically scans for unnecessary files, such as temporary files, system cache, and outdated downloads, and removes them. By avoiding disk overloads, this feature preserves performance, which is especially helpful for resource-intensive programs like Draw It or Lose It that could produce a lot of data. Windows also supports the New Technology File System (NTFS), which offers powerful capabilities like encryption, data compression, and journaling to improve data security and integrity. OneDrive integration offers cloud storage for larger-scale deployments or cloud-based apps, facilitating smooth data synchronization between devices and lowering the need for local storage. As the game grows and produces more data, the combination of Storage Sense with NTFS guarantees that storage management stays effective and the system functions without hiccups.

1. **Memory Management**: Windows is excellent at virtual memory management and paging, which makes it easier to handle big apps or several users at once without running out of memory.

* Update: For resource-intensive apps like Draw It or Lose It, Windows shines at virtual memory management and paging, which are crucial, particularly in settings with several people using the system at once. By combining RAM and disk space, Windows' virtual memory enables each process to access more memory than is physically accessible. Windows frees up RAM for running processes by moving less frequently used data to the paging file on the disk when physical memory is nearly full. The system will never run out of memory thanks to this paging feature, even when managing huge programs or numerous users at once. Furthermore, Windows employs memory compression to maximize available memory by compressing pages of inactive memory, which enables more data to be stored in RAM. Draw It or Lose It can manage high memory needs and retain maximum performance thanks to this, memory protection, and the ability to dynamically assign memory to processes. This is especially true in real-time, multiplayer gaming situations.

1. **Distributed Systems and Networks**: Distributed systems and networking are essential for a cross-platform, multiplayer game such as Draw It or Lose It. The architecture of Windows can facilitate cross-platform communication (e.g., web browsers, Windows PCs, and mobile apps).

* Update: A distributed system architecture is necessary for \*Draw It or Lose It\* to function flawlessly across several platforms (web browsers, Windows PCs, and mobile apps). WebSocket’s or RESTful APIs can be used to host the game's backend on several servers, allowing clients to synchronize and communicate in real time. These technologies are supported by Windows, allowing for seamless platform integration. Scalability is offered via cloud-based infrastructure, which also uses redundancy and load balancing to guarantee availability and lower latency. Component dependencies, however, could provide problems like heavy traffic or network disruptions. Caching and automated failover techniques can be used to lessen this, preserving resilience and performance while guaranteeing a seamless cross-platform experience even in the face of interruptions.

1. **Security**: Network-based multi-interaction systems typically need to develop unique architecture for data exchange and player communications. To communicate the data among players and to help each user remember the info, some kind of database will be utilized.

* Update: Encryption is necessary to protect user data for Draw It or Lose It on several platforms. To prevent interception, data should be encrypted in transit using TLS and at rest using AES-256. Multi-factor authentication (MFA) should be used in conjunction with OAuth 2.0 or JWT tokens for authentication to increase security. Strong security measures, such as firewalls and Intrusion Detection Systems (IDS), are available on the Linux-based server architecture to prevent unwanted access. Access control and encryption for sensitive user data can improve database security, while data retention policies can reduce risks by erasing or anonymizing historical data. These safeguards guarantee that user data is safe across all platforms, both during network communications and while it is being stored.