

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**](#_Toc115077323) **4**

[**Domain Model**](#_Toc115077324) **4**

[**Evaluation**](#_Toc115077325) **5**

[**Recommendations**](#_Toc115077326) **11**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 02/24/2024 | Patrick Coyne | Initial draft of the software design document for the Draw It or Lose It gaming application. |

## [Executive Summary](#_sbfa50wo7nsh)

Creative Technology Solutions (CTS) is tasked with developing a web-based version of "Draw It or Lose It," a game currently available as an Android app. The client, The Gaming Room, seeks to expand the game's accessibility across multiple platforms. The game, reminiscent of the 1980s TV game "Win, Lose or Draw," involves teams competing to guess what is being drawn.

Our solution involves designing a multi-platform web application that supports multiple teams and players, ensuring unique identifiers for games, teams, and players to maintain game integrity. The game's transition to a web-based platform addresses the need for broader accessibility and enhanced user experience. This document outlines the software architecture, design considerations, and development strategies to meet these requirements while ensuring a seamless and engaging gaming experience.

## Requirements

**Multi-Platform Accessibility**: The game must be accessible on various platforms, transitioning from an Android-only app to a web-based application compatible across different devices.

**Multi-Team Gameplay**: The game should support the participation of multiple teams, allowing for a competitive and interactive gaming experience.

**Player Management**: Each team must be capable of having multiple players, with a system to manage these players effectively within the game.

**Unique Identification**: The game, teams, and players must have unique identifiers. This step is crucial for ensuring that team and game names are unique and to avoid conflicts in game instances.

**Single Game Instance**: The system must enforce that only one instance of the game can be active in memory at any given time, using unique identifiers for each game session to maintain this constraint.

**User-Friendly Interface**: The application should offer an intuitive and engaging user interface that aligns with the gameplay experience of the existing Android app.

**Scalable Architecture**: The software design should accommodate future expansions and updates without significant overhauls.

## [Design Constraints](#_2et92p0)

**Cross-Platform Compatibility**: Developing a web-based game that operates seamlessly across various platforms (like PCs, tablets, and smartphones) poses a significant constraint. Cross-platform compatibility requires responsive design strategies to ensure consistent user experience across different screen sizes and operating systems.

**Network Dependence**: A web-based application relies heavily on network stability and speed. Variations in network quality among users can impact the game's performance and user experience.

**Browser Compatibility**: Ensuring that the game functions correctly on all major web browsers is a constraint, requiring extensive testing and possibly limiting the use of certain advanced web technologies.

**Concurrency Management**: Managing multiple active games and players simultaneously, especially with the requirement of a single active game instance, poses a challenge in ensuring data integrity and synchronization.

**Scalability**: The need to scale the application for a potentially large number of users while maintaining performance is a crucial constraint, influencing the choice of architecture and technologies.

## [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)

**Inheritance**: The Game, Team, and Player classes inherit from the Entity class. The inheritance is depicted by lines with open arrows pointing from these classes to Entity. Inheritance allows these classes to share properties like id and name, promoting code re-usability and a hierarchical organization.

**Encapsulation**: Each class encapsulates its data and behavior. For instance, the Game class has a list of Team objects and related methods, and Team has a list of Player objects, and so forth. This encapsulation ensures data is contained within appropriate classes, enhancing maintainability and integrity.

**Association**: The diagram shows associations between classes. GameService is associated with Game (0...\*), indicating that one GameService can manage zero to many Game objects. Similarly, Game is associated with Team, and Team with Player, indicating their relationships.

**Singleton Pattern**: The GameService class implements the Singleton pattern, as indicated by the method getInstance(). The Singleton pattern ensures that only one instance of GameService exists, managing game creation and data, which aligns with the requirement of a single active game instance.

**Use relationship**: The ProgramDriver class uses the SingletonTester class (indicated by a <<uses>> relationship), meaning ProgramDriver is responsible for initializing the application and utilizing SingletonTester to ensure the Singleton pattern is correctly implemented.

The principles and relationships in the UML class diagram contribute to fulfilling the software requirements efficiently by establishing a clear structure, promoting code reuse, ensuring data integrity, and aligning with the game’s specific requirements like unique identifiers and single instance management.

"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac servers benefit from a Unix-based foundation, offering stability, security, and efficient resource management, which is essential for web-based applications. One of the primary advantages of using a Mac server includes its integration with Apple's ecosystem, allowing seamless operation with other Apple services and software. This integration can significantly enhance the development workflow, particularly for applications targeting iOS or macOS clients. Additionally, Mac servers are known for their user-friendly interface, simplifying server management and maintenance tasks. However, there are notable weaknesses to consider. The hardware and software costs associated with Mac servers are generally higher than those for Linux or Windows, potentially increasing the overall budget for server infrastructure. Furthermore, while macOS is robust and secure, the server variant is less commonly used in enterprise environments than Linux and Windows servers, potentially limiting community support for troubleshooting and optimization. | Linux's most defining characteristic is its open-source nature, allowing extensive customization to meet specific performance and security requirements. This flexibility is critical for a web-based application that needs to scale up to thousands of users, as it allows for fine-tuned optimization of server resources to handle large volumes of traffic efficiently.  The advantages of Linux are numerous, including no licensing costs for the operating system, which can significantly reduce the overall expense of server deployment. Linux servers are renowned for their stability, security, and performance, making them a preferred choice for many web applications. However, there are weaknesses to consider. Linux requires more technical expertise to manage and optimize effectively, which may necessitate additional training or hiring specialized personnel. The command-line interface, while powerful, can be daunting for those accustomed to graphical interfaces, potentially complicating server administration tasks.  Holistically, when considering the client's requirement for a scalable, web-based application, Linux presents an attractive server-side solution due to its cost-effectiveness, robust performance, and adaptability. Its compatibility with a wide array of development tools and environments aligns well with the need for a flexible development and hosting platform to accommodate a growing user base. The challenge of requiring technical proficiency can be mitigated through strategic staffing and leveraging the extensive Linux community for support, making Linux a compelling choice for hosting the client's web application. | A key characteristic of Windows servers is their integration with the broader Microsoft ecosystem, including Active Directory, SQL Server, and .NET, which can streamline development and deployment processes for applications reliant on these technologies. This integration facilitates a cohesive environment where administration and development tools are tightly coupled, enhancing productivity and reducing compatibility issues. The advantages of opting for Windows servers include a user-friendly interface and comprehensive support from Microsoft, making it an accessible choice for organizations with existing Windows infrastructure or those that prefer a GUI-driven management approach. However, Windows servers have weaknesses to consider. Licensing costs for Windows Server editions can significantly impact the budget, especially for large-scale deployments. Additionally, while Windows servers can host various web applications, they may not offer the same performance optimization or customization level as Linux for specific web technologies, potentially affecting scalability and efficiency. | A primary characteristic of mobile devices is their portability and built-in connectivity, which support on-the-go access to information and services. However, when it comes to hosting applications, they face significant challenges, such as limited processing power, storage capacity, and network bandwidth compared to dedicated server hardware. These constraints make mobile devices unsuitable for serving web applications, especially those requiring scalability to support thousands of users simultaneously. |
| **Client Side** | Mac users expect applications to leverage their devices' high-resolution display and performance capabilities, necessitating a focus on responsive design and testing on Safari, the default web browser on macOS. Cost considerations include acquiring Mac hardware for development and testing, ensuring compatibility, and optimizing performance. In terms of time, developing for Mac may require additional effort in designing interfaces that meet macOS users' aesthetic and usability standards, potentially including adaptations for Safari-specific behaviors. The development process must prioritize cross-browser compatibility while optimizing for Safari, requiring a nuanced understanding of web standards and Apple's design guidelines. | Linux users often utilize a variety of web browsers, including Firefox, Chrome, and Opera, necessitating comprehensive testing to ensure compatibility. Cost is generally lower regarding software, as many development tools and testing environments are available for free or at minimal expense. However, the time investment in testing across multiple desktop environments (e.g., GNOME, KDE) can increase, as developers must ensure consistent performance and appearance despite the fragmentation. Expertise is required in web development standards and understanding the nuances of different Linux distributions and desktop environments. Leveraging containerization tools like Docker to streamline testing and deployment processes may be helpful. The open-source nature of Linux encourages a development approach that is adaptable and community-supported, enriching the development process with broader testing and feedback loops. | Windows users access web applications through various browsers, including Edge, Chrome, and Firefox, requiring that applications are thoroughly tested for cross-browser compatibility. Cost considerations include potential investments in development tools and software, although many, such as Visual Studio Code, offer robust features for free. Time may be impacted by the need to support a wide range of Windows versions, each with its peculiarities affecting web application behavior. Ensuring accessibility and consistent user experience across these versions demands extensive testing and conditional logic in the codebase. Expertise in modern web technologies is essential, with additional emphasis on accessibility features and security measures that cater to corporate and individual users alike. Familiarity with Microsoft's development ecosystem can streamline the process, leveraging tools and libraries optimized for Windows environments. | Supporting mobile devices introduces a set of development considerations that significantly impact cost, time, and expertise. Cost factors include the necessity for responsive design frameworks and cross-platform development tools like React Native or Flutter to manage a single codebase for both iOS and Android, which can reduce overall development and maintenance expenses. Time considerations are critical, as developing for multiple mobile platforms can significantly extend the development cycle, especially if native applications are required. Efforts must be allocated for testing across various devices and screen sizes to ensure a consistent and engaging user experience. Expertise is paramount; developers must be proficient in specific languages and development environments and understand the intricacies of mobile UI/UX design. The development process must also account for the submission and review processes of the App Store and Google Play, requiring knowledge of platform guidelines and restrictions. |
| **Development Tools** | For Mac development, primarily targeting web-based applications, various programming languages and tools are employed to ensure compatibility and performance. Xcode stands out as Mac's primary integrated development environment (IDE), offering extensive support for Swift and Objective-C, languages essential for developing macOS and iOS applications. Visual Studio Code is a popular choice for web development among Mac developers due to its versatility, extensive plugin ecosystem, and support for languages like HTML, CSS, JavaScript, Python, and PHP. Safari's Web Inspector provides powerful tools for debugging and optimizing web applications specifically for the Safari browser. Developers may also leverage React or Angular frameworks for front-end development to create responsive and dynamic user interfaces. Combining these tools and languages facilitates the development of web-based applications optimized for the Mac ecosystem, providing a seamless experience across Apple devices. | Developing for Linux involves many open-source programming languages and tools, reflecting the platform's flexibility and community-driven nature. Eclipse and Visual Studio Code are widely used IDEs that support many languages, including Java, Python, PHP, and JavaScript, catering to the diverse needs of web application development. Given Linux's strong association with server-side technologies, languages like Python, with its powerful frameworks such as Django and Flask, and PHP, commonly used with Apache or Nginx web servers, are prevalent. For client-side development, web technologies such as HTML5, CSS3, and JavaScript frameworks like Vue.js are essential for creating responsive designs. The Linux environment also benefits from an extensive suite of development tools like Git for version control, Docker for containerization, and Bash scripting for automation, enhancing the efficiency and scalability of development workflows. | Development tools and languages are chosen for Windows to leverage the operating system's widespread use and integration capabilities. Visual Studio emerges as the premier IDE, providing comprehensive support for C#, .NET, and ASP.NET—critical technologies for developing web applications on Windows. These Microsoft-centric languages and frameworks offer robust libraries and APIs for building secure, scalable web applications. Visual Studio Code, a lighter IDE, is also popular for web development, supporting various programming languages such as JavaScript, TypeScript, Python, and PHP. Windows developers frequently utilize Microsoft SQL Server for database management, integrating seamlessly with .NET-based applications. Additionally, front-end development tools and libraries, including React, Angular, and Bootstrap, are employed to design responsive user interfaces compatible with Windows browsers like Edge and Chrome. | Developing for Mobile Devices requires specialized tools and languages tailored to iOS and Android platforms. For iOS development, Xcode is the essential IDE, with Swift and Objective-C as the primary programming languages. Xcode provides an all-encompassing environment for designing, coding, testing, and deploying iOS applications. Android Studio, the official IDE for Android development, supports Kotlin and Java, facilitating the development of Android applications with features such as code completion, debugging, and simulation tools. Cross-platform development frameworks like React Native and Flutter allow developers to write once and deploy on iOS and Android using JavaScript (React Native) and Dart (Flutter). These tools streamline the development process, enabling the creation of mobile applications that can interact seamlessly with web-based back-end applications, ensuring a consistent user experience across all device types. |

## Recommendations

1. **Operating Platform**: Given the size and nature of the assets used by "Draw It or Lose It," a combination of Linux-based cloud servers is recommended. Cloud platforms such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure offer scalable, secure, and efficient computing environments that can dynamically adapt to the game's workload. The cloud services provide extensive support for Linux, ensuring the game can leverage Linux's performance and customization advantages while benefiting from the cloud's scalability.
2. **Operating Systems Architectures**: The architecture of Linux-based cloud servers is a cornerstone for deploying modern, scalable, and resilient applications like "Draw It or Lose It" across various computing environments. The cloud infrastructure employs virtualization and containerization within the architecture to provide highly flexible and efficient computing resources. Virtualization allows the creation of multiple isolated virtual machines on a single physical server, leveraging Linux's support for hypervisors. This strategy ensures efficient resource utilization and flexibility in resource allocation, which is crucial for managing the dynamic workloads of a multiplayer online game. Furthermore, Linux's container technology, orchestrated by Kubernetes, packages the game and its dependencies into compact, executable units, facilitating rapid deployment, consistent operation across different environments, and seamless scaling of game services to meet player demand. The integration of Linux with these cloud-based technologies enables "Draw It or Lose It" to leverage the full spectrum of cloud computing benefits, including scalability, performance, and geographic reach.
3. **Storage Management**: A Content Delivery Network such as Amazon CloudFront or Google Cloud CDN alongside cloud storage solutions like Amazon S3 or Google Cloud Storage is recommended for managing large images. These services offer high availability and low latency, ensuring the game's large images are quickly accessible to players worldwide. Cloud storage solutions provide scalable, secure, and cost-effective options to store and manage large volumes of image assets, while CDN caches content closer to users, reducing load times.
4. **Memory Management**: Cloud platforms offer automated scaling and load balancing, ensuring efficient use of memory resources based on demand. Automated scaling ensures that the game's server infrastructure can dynamically adjust to the number of active players and games without manual intervention. This feature means that during peak play times, when the demand for the game's resources is highest, the cloud platform can allocate additional memory and computing resources to maintain smooth game performance. Conversely, during periods of lower demand, it can scale down resources to optimize costs.

In a Linux environment, techniques like caching and virtual memory optimize performance. Caching is crucial for "Draw It or Lose It" because it relies on an extensive library of stock drawings. By storing frequently accessed images in faster, temporary storage, the game can reduce the time it takes to load these assets for players, leading to quicker, more responsive gameplay. This technique is essential as drawings are rendered at a steady rate and are fully complete at the 30-second mark of each game round. Efficient caching ensures these images are readily available, minimizing latency and enhancing the player experience.

1. **Distributed Systems and Networks**: To ensure "Draw It or Lose It" delivers a seamless gaming experience across various platforms, employing a distributed system architecture with cloud-based services is crucial. The architecture capitalizes on Linux-based cloud servers for enhanced scalability and reliability, which are essential for accommodating fluctuating player numbers and maintaining game performance. By utilizing HTTP/HTTPS protocols for communication, the game ensures efficient and standardized data exchange across different devices, from desktops to mobiles, without compatibility issues. API gateways play a pivotal role in the ecosystem, managing traffic through routing, request validation, and authentication, bolstering security, and facilitating smooth real-time interactions. Furthermore, Content Delivery Networks are instrumental in optimizing the delivery of game assets, significantly cutting down load times and bolstering responsiveness. Content Deliver Networks enhance the player experience, reduce bandwidth costs, and promote global accessibility. The distributed approach also introduces fault tolerance, ensuring the game remains resilient against server failures or network disruptions, preserving the integrity of gameplay and user data through encrypted transmissions and DDoS protection measures.
2. **Security**: Cloud platforms offer comprehensive security tools and features to protect against threats and ensure data privacy. Transport Layer Security (TLS) for data encryption, Identity and Access Management (IAM) for access control, and DDoS protection services ensure that user information and game data are securely managed. Regular security audits, automated compliance checks, and data encryption at rest and in transit are crucial for safeguarding sensitive information.