

“Draw It or Lose It”

# **CS 230 Project Software Design**

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 |
| --- | --- | --- | --- |
| 3.0 | 10/15/23 | Tinisha Cain-Beckford | Added recommendations for operating platform, operating systems architectures, storage management, memory management, distributed systems and networks, and security. |

## [Executive Summary](#_sbfa50wo7nsh)

The software design problem involves creating a web-based gaming application called "Draw It or Lose It," inspired by the popular 1980s TV game show "Win, Lose or Draw." Firstly, the application must support multiple teams, each comprised of multiple players. Secondly, it should enforce uniqueness for game and team names to prevent naming conflicts. Finally, the application should ensure that only one instance of the game can exist in memory at any given time, achieved through the use of unique identifiers.

To address these requirements, we propose a comprehensive software design that incorporates a robust database system to manage game instances, teams, and players. Each game instance will be associated with a unique identifier, ensuring only one game is active at any moment. Team and player information will be stored in the database, allowing for easy management, and checking for name uniqueness during team creation. Additionally, a user-friendly interface will be developed to facilitate team creation and gameplay. By implementing this solution, we aim to streamline the development of "Draw It or Lose It" and provide a scalable and efficient gaming application that meets specific software requirements.

## Requirements

The client's business requirements for the development of the web-based gaming application are centered around creating an engaging multiplayer experience. The game should support multiple teams, each with multiple players, mirroring the collaborative and competitive aspects of the original TV game show. Furthermore, there is the importance of unique game and team names to enhance user experience by avoiding confusion or duplication. Ultimately, the business goal is to deliver a captivating gaming platform that offers an entertaining experience while ensuring smooth and intuitive team management.

On the technical front, they want a system that can manage unique identifiers for each instance of the game, team, and player to ensure only one game can exist in memory at any given time. This necessitates the development of a database system to handle data storage and retrieval efficiently. Additionally, the technical solution should offer a user-friendly interface to facilitate team creation and gameplay, enhancing the overall user experience. The combination of these technical requirements aims to provide a stable and scalable gaming application that aligns with the client's business objectives.

## [Design Constraints](#_2et92p0)

Design constraints for developing the game application in a web-based distributed environment include:

1. Latency and Network Communication: In a distributed web-based environment, there can be network latency due to data transmission over the internet. This constraint impacts real-time gameplay, as there might be delays in rendering images or transmitting guesses between players.

2. Scalability: As the game is intended to support multiple teams and players, the application must be designed to handle a potentially large user base simultaneously. Scalability constraints require a robust architecture that can dynamically allocate resources and handle increased traffic during peak usage.

3. Data Security: Ensuring the security of user data is paramount, especially in a multiplayer environment. The application must adhere to strict security constraints to protect sensitive user information and prevent unauthorized access or data breaches. This necessitates the implementation of encryption, secure user authentication, and data access controls.

4. Cross-Platform Compatibility: To reach a broad audience, the application should work seamlessly across various web browsers and devices, including desktops, laptops, tablets, and mobile phones. Designing for cross-platform compatibility is a constraint that requires thorough testing and possibly the use of responsive design techniques to adapt to different screen sizes and resolutions.

5. Data Consistency and Reliability: In a distributed environment, ensuring data consistency and reliability can be challenging. Design constraints here require implementing mechanisms like distributed databases or synchronization protocols to maintain the integrity of game data across different servers or instances. Handling scenarios like network failures and server crashes becomes crucial to prevent data loss or inconsistencies.

6. Browser Limitations: Web-based applications are subject to constraints imposed by web browsers, such as limited local storage, browser compatibility issues, and varying JavaScript performance. Developers need to be aware of these limitations and work within them to ensure the application functions correctly on a wide range of browsers.

7. User Experience: Delivering a seamless and enjoyable user experience is a design constraint that can be influenced by factors like device capabilities, screen sizes, and input methods. Ensuring a consistent and intuitive user interface across different platforms is essential for user satisfaction.

8. Performance Optimization: The application must be optimized for performance, especially for rendering images and handling real-time interactions. Design constraints related to performance require careful coding practices, efficient algorithms, and server resources allocation to deliver a responsive and smooth gaming experience.

## [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 serves as the foundational class in our application, encapsulating common attributes and behaviors shared by all entities. It incorporates the 'id' and 'name' attributes, implying that every entity within the application will inherently possess an identifier and a name through inheritance. Building upon this foundation, the Game class inherits from the Entity class, following the structure outlined in the UML diagram.

To manage the uniqueness and identification of game, team, and player instances, the GameService class employs static variables, responsible for tracking the next identifier to be assigned for game id, team id, and player id. Leveraging the singleton pattern, the GameService class is transformed into a singular entity in memory, ensuring that only one instance of it exists at any given time. This is achieved through the creation of unique identifiers for each instance of a game, team, or player. Additionally, a vital constraint requires that every team and player have a distinct name, requiring a search for duplicate names before the addition of a new instance.

**"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 offers a Unix-based environment, which can be advantageous for web development due to its compatibility with popular web technologies like Apache, PHP, and MySQL. It provides a stable and secure platform, often favored by developers for its ease of use and development tools. However, Mac may have limitations in terms of scalability and performance compared to dedicated server solutions. It may not be as cost-effective as other hosting options, and it might be less common in enterprise-level hosting environments. Additionally, compatibility issues with certain Windows-specific software components may arise, potentially impacting the development workflow. | Linux is a highly versatile and widely used choice for hosting web-based software applications. Its characteristics include a stable and reliable environment, open-source nature, and a vast array of software tools and libraries. Linux excels in terms of security, making it a preferred option for hosting sensitive applications. Its scalability and performance capabilities are robust, and it can efficiently run a variety of web server software like Apache, Nginx, and databases such as MySQL or PostgreSQL. However, Linux might have a learning curve for users unfamiliar with Unix-based systems. | Windows can be a suitable choice for hosting a web-based software application, especially if the application relies on Microsoft technologies. Its characteristics include a user-friendly interface, widespread familiarity, and strong support for .NET framework and Microsoft SQL Server, making it a natural fit for applications developed using these technologies. Windows hosting provides excellent compatibility for ASP.NET applications and can integrate seamlessly with other Microsoft products and services. Windows servers are generally perceived as less secure than Linux servers, requiring diligent maintenance and security measures. | Mobile devices offer portability and flexibility, enabling users to access the application from anywhere with an internet connection. This can enhance accessibility and user engagement. Additionally, mobile devices often come with built-in sensors and capabilities like GPS, cameras, and touchscreens, which can be leveraged to create innovative and interactive features within the application. However, hosting on mobile devices presents significant limitations, such as limited processing power, memory, and storage capacity, making them unsuitable for hosting complex or resource-intensive applications. Mobile devices may also face challenges related to security, reliability, and scalability. |
| **Client Side** | Supporting multiple types of clients on the Mac platform may increase development costs due to the need for specialized hardware and software resources. Developing and testing software for different Mac models, OS versions, and screen sizes can be time-consuming and costly. Second, expertise in Mac development and compatibility is crucial, which could necessitate training developers to be proficient in macOS and its development tools. Overall, while catering to Mac users is essential for broader accessibility, it's important to allocate resources effectively to manage the potential increase in development costs, time, and the need for specialized expertise. | Supporting multiple types of clients on the Linux platform offers the advantage of being open source, reducing licensing expenses, and potentially decreasing overall development costs. However, adapting software for various Linux distributions and versions can consume time and resources, as compatibility and dependencies may differ. Developers need expertise in Linux-specific libraries, tools, and configurations, which may require additional training. Nevertheless, once developed, Linux software tends to be highly portable across distributions, potentially simplifying future maintenance and updates. | Supporting multiple types of clients on the Windows platform typically involves licensing fees for proprietary software tools, which can increase development expenses. While Windows provides a unified ecosystem, ensuring compatibility across different Windows versions and editions may require extra time and effort during development and testing phases. Expertise in Windows-specific technologies, such as the .NET framework, is essential. Fortunately, Windows offers extensive developer support, which can aid in expediting certain aspects of the development process. | Supporting multiple types of clients on mobile devices involve investments in platform-specific development tools, devices for testing, and potentially higher development costs due to the need for separate iOS and Android development. Time constraints arise from the necessity to develop and maintain two separate codebases for these dominant mobile platforms, which can extend development timelines. Expertise in mobile app development, including knowledge of platform-specific programming languages (Swift for iOS and Java/Kotlin for Android), user interface design guidelines, and the app store submission processes, is crucial. Furthermore, the rapid evolution of mobile hardware and operating systems requires ongoing expertise to ensure compatibility and optimization for various device types and screen sizes. While mobile apps provide access to a vast user base, these considerations highlight the need for careful planning, resource allocation, and specialized skills in mobile development to successfully support multiple types of clients on mobile devices.> |
| **Development Tools** | To develop software for the Mac platform, developers commonly use programming languages such as Swift and Objective-C. Swift is Apple's modern, easy-to-learn language designed for macOS, iOS, watchOS, and tvOS development. Objective-C, although older, is still relevant for macOS development, especially when working with legacy codebases. Popular Integrated Development Environments (IDEs) for Mac software development include XCode, Apple's official development environment, which provides a comprehensive set of tools, code editors, and debugging features for both Swift and Objective-C. Additionally, developers may utilize version control systems for dependency management. | Developing software for deployment on Linux include programming languages such as C, C++, Python, and JavaScript. C and C++ offer low-level system access and performance optimization capabilities, making them suitable for system-level programming on Linux. Python is popular for its versatility and ease of use, while JavaScript is utilized for web-based Linux applications. Linux development can be done using a range of integrated development environments such as Visual Studio Code, Eclipse, and JetBrains' IntelliJ IDEA. Additionally, developers rely on powerful terminal-based text editors like Vim and Emacs for code editing and version control tools like Git for collaboration and code management. Linux offers a diverse ecosystem of libraries, APIs, and package managers like apt, yum, and snap, which simplify dependency management and software distribution.> | Developing software for Windows include programming languages such as C#, C++, and Visual Basic.NET. Visual C++ is employed for system-level and performance-critical applications, while Visual Basic.NET caters to rapid application development. Microsoft Visual Studio is the primary IDE for Windows software development, offering robust tools, code editors, debugging features, and support for various programming languages. Additionally, developers may use tools like JetBrains Rider and Xamarin for cross-platform development or specialized IDEs like Unity for game development. Furthermore, version control systems like Git and package managers like NuGet assist in code management and dependency handling, enhancing the development process on the Windows platform.> | Developing software for deployment on mobile devices primarily involves programming languages tailored to each platform. For iOS devices like iPhones and iPads, Swift and Objective-C are the dominant languages. Swift, is a modern and efficient choice, while Objective-C is mostly for maintaining legacy codebases. On the Android platform, Java and Kotlin are the primary languages, with Kotlin gaining popularity for its conciseness and safety features. Cross-platform development tools like Flutter, React Native, and Xamarin enable developers to write code in a single language (Dart, JavaScript, and C#, respectively) for both iOS and Android. IDEs include XCode for iOS development and Android Studio for Android, offering debugging, testing, and emulation tools. Additionally, popular code editors like Visual Studio Code and JetBrains' IntelliJ IDEA can be used for mobile development across platforms, along with version control systems like Git and package managers for handling dependencies specific to each platform.> |

## 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**: To ensure that Draw It or Lose It can be easily expanded to other computing environments and platforms, it's advisable for The Gaming Room to adopt a web-based architecture. Developing the game as a web application would allow it to run on a variety of operating systems, including Windows, macOS, Linux, and even mobile devices with internet browsers. This approach leverages the platform-agnostic nature of web technologies, such as HTML5, CSS, and JavaScript, ensuring broad accessibility and scalability. It also simplifies updates and maintenance since changes can be made on the server-side without requiring users to download and install new software. By choosing a web-based architecture, The Gaming Room can efficiently expand Draw It or Lose It to reach a wider audience across diverse computing environments.
2. **Operating Systems Architectures**: The chosen operating platform architecture for Draw It or Lose It is a web-based architecture. In this architecture, the game application is hosted on a web server and accessed by users through web browsers on various operating systems, including Windows, macOS, Linux, and mobile devices. The game's frontend is built using web technologies such as HTML5, CSS, and JavaScript, ensuring cross-platform compatibility. User interactions are processed on the server, allowing for centralized control and data management. This architecture offers scalability, as it can accommodate a growing user base without significant changes to the client-side software. It also simplifies updates and maintenance, as updates can be deployed server-side without requiring users to install new software versions. Overall, the web-based architecture enables the game to reach a broad audience across different computing environments seamlessly.
3. **Storage Management**: For Draw It or Lose It, an appropriate storage management system would be a combination of relational databases and cloud storage solutions. Relational databases like PostgreSQL or MySQL can efficiently handle structured game data, such as user profiles, team information, and game statistics. They provide robust data consistency, security, and query capabilities. In parallel, cloud storage solutions like Amazon S3 or Google Cloud Storage can be used to store media assets like images and drawings associated with the game, ensuring scalability and easy access for users across different platforms. This hybrid approach combines the strengths of relational databases for structured data and cloud storage for media assets, providing a comprehensive storage management system that aligns with the web-based architecture's scalability and accessibility requirements.
4. **Memory Management**: In this web-based architecture, the application's code and assets are loaded into the memory of the web server. When users interact with the game, server-side memory management techniques, such as dynamic memory allocation and garbage collection, are employed to allocate and release memory resources as needed to handle user requests and game logic. This approach minimizes memory leaks and optimizes resource allocation, ensuring that the application can serve multiple users simultaneously without exhausting system memory. Additionally, client-side memory management is handled by web browsers, which automatically manages memory for rendering the game's user interface and processing user input. Overall, these memory management techniques contribute to the smooth and reliable performance of Draw It or Lose It in a web-based environment.
5. **Distributed Systems and Networks**: Draw It or Lose It should implement RESTful APIs, allowing different platforms to exchange data and commands seamlessly. These APIs should be designed with platform-agnostic data formats, such as JSON, to ensure compatibility.   
    In terms of network connectivity, the system should carefully handle different network conditions. Implementing retry mechanisms for failed network requests, handling timeouts, and optimizing data transmission for low-bandwidth connections can enhance the user experience. Redundancy and load balancing in the network architecture can help mitigate the impact of network outages or high traffic loads.  
    Furthermore, dependencies between components within the distributed system should be managed carefully. Service discovery mechanisms and containerization technologies like Docker can assist in dynamically locating and deploying services as needed.

Overall, achieving effective communication between platforms in a distributed software system like Draw It or Lose It necessitates a combination of well-defined protocols, network resilience strategies, and thoughtful management of component dependencies. This approach ensures that the game functions reliably across various devices and under diverse network conditions.

1. **Security**: Protecting user information on and between various platforms in Draw It or Lose It is paramount for ensuring security. To achieve this, the web-based architecture should implement multiple security layers. First, user authentication and authorization mechanisms should be robust, utilizing industry-standard protocols. The operating platform can leverage its built-in security features, such as access control lists and user authentication, to enhance user protection.  
    Data transmission between platforms should be encrypted using TLS/SSL protocols to safeguard user information in transit. Additionally, sensitive user data, including passwords and personal information, must be securely stored using best practices.  
    Regular security audits, penetration testing, and code reviews are essential to identify and mitigate vulnerabilities. Continuous monitoring and logging of user interactions and system activities can help detect and respond to security threats promptly. Compliance with data protection regulations should also be a priority.  
    Incorporating security into the development lifecycle and staying updated with security patches and updates for the recommended operating platform are vital. By combining these measures with the platform's built-in security capabilities, Draw It or Lose It can provide a secure environment for user information on and between different platforms, safeguarding user data against potential threats and vulnerabilities.

**References**

*How do you handle high traffic and demand for data storage and retrieval in web applications?* (n.d.). Www.linkedin.com.

*Memory Leaks and Memory Consumption in web applications (Par*. (2011, April 26). Telerik Blogs

*Part Two - Process Management - Operating System Concepts, 8th Edition [Book]*. (n.d.). Www.oreilly.com.

‌‌

(2023). Vitalsource.com.

*11 Web Application Security Best Practices You Need to Know | Mobindustry*. (2022, April 1).

‌

‌

**‌**