

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 | <01/28/2024> | <Caleb Leavell> | Completion of Executive summary and Design Constraints |
| 2.0 | 02/10/2024 | Caleb Leavell | Completion of Evaluation |
| 3.0 | 2/25/2024 | Caleb Leavell | Completion of 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 requires a software design to address a game called Draw It or Loose It. This design requires the ability to accommodate one or more teams, assign multiple players to each team, ensure each game and team have unique names, and ensure the existence of only one game saved to memory at any one time. To meet these requirements I purpose an application with a game manager responsible for creating and managing unique instances of games, teams, and players.

## 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)

1. The game must work on the web, so we'll use web technologies like HTML, CSS, and JavaScript.
2. The game should work well on different web browsers like Chrome, Firefox, Safari, and Edge, so we'll test it thoroughly.
3. We need to protect the game from cyber threats by encrypting data, using secure communication, and implementing authentication and authorization measures.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**"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** | Macs offer a reliable, secure, and user-friendly platform for hosting web-based software applications, particularly for developers who value integration with the Apple ecosystem and prioritize ease of use. However, they may not be the most cost-effective option for all use cases, and their scalability could be a limiting factor for some applications. | Linux offers a powerful, flexible, and cost-effective platform for hosting web-based software applications. Its stability, performance, and security features make it a popular choice among developers and system administrators. However, users should be prepared to invest time in learning Linux and navigating the diverse ecosystem of distributions and software options. Overall, Linux's strengths outweigh its weaknesses for many hosting scenarios, particularly for those seeking customization, performance, and community support. | Windows offers a user-friendly platform with strong support for Microsoft technologies, making it an appealing choice for organizations with existing investments in the Microsoft ecosystem. However, licensing costs, resource requirements, security considerations, and limitations in CLI tools may impact its suitability for certain web hosting scenarios. | Mobile devices offer accessibility, portability, and hardware integration advantages for hosting web-based applications, they also present challenges such as limited resources, security concerns, and network dependence. Mobile hosting may be suitable for lightweight, user-focused applications targeting mobile users but may not be ideal for hosting complex or enterprise-grade web applications requiring robust infrastructure and scalability. |
| **Client Side** | Supporting multiple types of clients on Mac platforms demands meticulous software development considerations. Developers must weigh the costs associated with acquiring Mac-specific development tools, hardware for testing, and ongoing maintenance. Time investment is crucial, requiring developers to familiarize themselves with macOS development technologies, design intuitive user interfaces adhering to macOS design guidelines, and conduct thorough compatibility testing across different versions of macOS and hardware configurations. Expertise in Mac development skills, UI/UX design principles, and compatibility testing is indispensable for delivering high-quality applications that meet the diverse needs of Mac users. | Supporting multiple types of clients on Linux platforms involves several critical software development considerations. Firstly, developers must account for the diverse range of Linux distributions, each with its unique package managers, libraries, and configurations. This necessitates thorough testing and compatibility checks across various distributions to ensure the application functions reliably across different environments. Secondly, developers need to select development tools and frameworks that are compatible with Linux, such as GCC, CMake, Python, and Node.js, among others. Additionally, understanding the intricacies of Linux system administration is essential for optimizing performance, managing dependencies, and ensuring security. Furthermore, open-source collaboration and adherence to Linux community standards are important for fostering community support and ensuring the application's integration within the Linux ecosystem. Finally, developers should prioritize scalability and performance tuning to accommodate varying hardware specifications and optimize resource utilization for different client devices. | Supporting multiple types of clients on Windows platforms requires careful consideration of several software development factors. Firstly, developers must account for the cost associated with acquiring licenses for development tools, such as Visual Studio and other Microsoft development frameworks. Additionally, testing and ensuring compatibility across various versions of Windows, including different editions (e.g., Home, Pro, Enterprise), can incur additional expenses. Time is another crucial consideration, as developers need to allocate sufficient time for familiarizing themselves with Windows-specific development technologies and design principles, such as the .NET framework, C#, and the Universal Windows Platform (UWP). Furthermore, expertise in Windows development skills, including UI/UX design for Windows applications, is essential for delivering a consistent and intuitive user experience across different client devices. Thorough testing and debugging across different Windows versions and hardware configurations are also necessary to ensure compatibility and optimal performance. Overall, addressing these software development considerations for supporting multiple types of clients on Windows platforms requires a combination of financial investment, time allocation, and expertise in Windows development technologies and practices | < Supporting multiple types of clients on mobile devices involves several critical software development considerations. Firstly, developers must account for the cost associated with acquiring development tools, such as IDEs (Integrated Development Environments) like Android Studio or Xcode for Android and iOS development respectively. Additionally, there may be costs associated with acquiring physical devices for testing across different platforms (e.g., iOS and Android) and various device models to ensure compatibility. Time is another crucial consideration, as developers need to allocate sufficient time for learning platform-specific development languages and frameworks (e.g., Java/Kotlin for Android, Swift for iOS) and understanding platform-specific design guidelines for creating intuitive user interfaces. Furthermore, expertise in mobile development skills, including knowledge of platform-specific APIs (Application Programming Interfaces) and optimization techniques for performance and battery efficiency, is essential for delivering high-quality mobile applications. Thorough testing across different devices, screen sizes, and operating system versions is also necessary to ensure compatibility and provide a seamless user experience. Overall, addressing these software development considerations for supporting multiple types of clients on mobile devices requires a combination of financial investment, time allocation, and expertise in mobile development technologies and best practices. |
| **Development Tools** | For deploying software on Mac, developers primarily utilize Swift, the main programming language for macOS and iOS application development, alongside Objective-C, which is still used for maintaining legacy applications. The essential Integrated Development Environment (IDE) for macOS development is Xcode, offering a comprehensive suite of tools for coding, debugging, interface design, and performance analysis. Alternatively, Visual Studio Code with Swift extensions and AppCode by JetBrains provide additional options for Swift and Objective-C development. To manage dependencies, developers leverage tools like CocoaPods for dependency management and Swift Package Manager for Swift projects. These languages and tools collectively empower developers to create native Mac applications efficiently, leveraging the capabilities of the macOS platform. | For deploying software on Linux, developers rely on a diverse set of programming languages and tools tailored to the Linux environment. These include C and C++, which are widely used for system-level programming and high-performance applications, alongside Python, favored for its simplicity and versatility in scripting and automation tasks. JavaScript, particularly with the Node.js runtime, is increasingly popular for web servers and command-line tools, while Go provides a robust option for building scalable and concurrent applications. Integrated Development Environments (IDEs) such as Visual Studio Code and Eclipse offer comprehensive development environments with support for multiple languages, while text editors like Vim and Emacs provide efficient and customizable options for coding. Additionally, tools like GCC for compiling code and Git for version control are indispensable for managing software projects on Linux. | For deploying software on Windows, developers rely on a range of programming languages and tools tailored specifically for the Windows environment. These include languages such as C#, which is widely used for developing desktop applications, Windows services, and Universal Windows Platform (UWP) apps. Additionally, Visual Basic .NET (VB.NET) offers a more accessible syntax for Windows application development. C++ is commonly employed for system-level programming and game development on the Windows platform. With the rise of web technologies, JavaScript is increasingly utilized for building Windows applications, especially for web development and cross-platform applications using frameworks like Electron. To facilitate development, Microsoft's Visual Studio serves as the primary Integrated Development Environment (IDE), offering a comprehensive suite of tools for building, testing, and debugging Windows applications in various languages. Visual Studio Code (VSCode) is also popular among Windows developers for its lightweight yet powerful features and support for multiple programming languages. .NET Framework and .NET Core SDK provide the necessary libraries, runtime environments, and tools for building and deploying Windows applications in languages like C#. Windows Forms and WPF Designer within Visual Studio enable developers to design user interfaces visually and generate code automatically. Additionally, tools like Git for version control and NuGet for package management streamline the development process for Windows applications. | For building software deployed on mobile devices, developers leverage a variety of programming languages and tools tailored specifically for mobile development. Java and Kotlin are predominantly used for Android app development, with Java serving as the official language and Kotlin gaining popularity for its modern features and interoperability with Java. On the iOS side, Swift is the primary language for building apps, known for its safety features and concise syntax, while Objective-C remains relevant for maintaining legacy code. Android Studio stands as the official IDE for Android development, offering a comprehensive suite of tools, while Xcode serves the same purpose for iOS development, providing features for coding, debugging, and interface design. Visual Studio Code offers cross-platform support for mobile development, enabling developers to work with Java, Kotlin, Swift, and Objective-C. Additionally, frameworks like Flutter and React Native allow for cross-platform development, enabling developers to write code once and deploy it on both iOS and Android platforms, using languages like Dart and JavaScript, respectively. These languages and tools empower developers to create a diverse range of mobile applications, from productivity tools to games and social networking apps, catering to the needs of a vast mobile user base. |

## 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**: For The Gaming Room’s expansion of "Draw It or Lose It" across multiple computing environments, I recommend utilizing a cloud-based operating platform, specifically Amazon Web Services (AWS). AWS offers a highly scalable, reliable, and efficient cloud computing environment that supports a wide range of operating systems architectures. This flexibility will enable "Draw It or Lose It" to seamlessly operate across different devices and platforms, catering to a broader audience and ensuring a consistent gaming experience.
2. **Operating Systems Architectures**: AWS supports various operating systems architectures, including Linux and Windows Server, making it a versatile choice for deploying "Draw It or Lose It." The ability to run on both architectures allows for the development of cross-platform capabilities, ensuring that the game can cater to users regardless of their device’s operating system. Linux servers, known for their stability and performance, could be particularly beneficial for backend services, while Windows Server could be utilized for components that may rely on specific Windows-based technologies.
3. **Storage Management**: AWS offers a comprehensive suite of storage management solutions that can be tailored to the needs of "Draw It or Lose It." Amazon Simple Storage Service (S3) is recommended for storing game assets, user data, and backups. S3 provides high durability, availability, and scalability, ensuring that game data is securely stored and accessible when needed. For database storage, Amazon RDS (Relational Database Service) can be utilized to manage user accounts, game state data, and leaderboards, offering automated backups, patching, and scalability.
4. **Memory Management**: AWS's infrastructure is designed to optimize memory management, essential for ensuring the smooth operation of "Draw It or Lose It." AWS EC2 (Elastic Compute Cloud) instances provide various options for memory-optimized instances, such as R-series, X-series, and Z-series, which are tailored for high-performance databases, in-memory databases, and large-scale caching. These instances can be leveraged to ensure that "Draw It or Lose It" has sufficient memory resources to manage game state, user sessions, and real-time data processing efficiently.
5. **Distributed Systems and Networks:** "Draw It or Lose It" can leverage AWS’s global infrastructure to facilitate communication across various platforms through distributed systems and networks. AWS provides services like Amazon CloudFront for content delivery network (CDN) services, ensuring low latency and high-speed access to game assets worldwide. Additionally, AWS’s VPC (Virtual Private Cloud) and Direct Connect can be used to create a secure, dedicated network connection between different deployment environments, enhancing connectivity and reducing dependencies and potential outages.
6. **Security:** Security is paramount for protecting user information on and between various platforms. AWS offers comprehensive security features that align with The Gaming Room’s needs. AWS Identity and Access Management (IAM) allows granular control over access to AWS services and resources, ensuring that only authorized personnel can access sensitive data. Amazon Cognito provides user authentication and data synchronization across different devices, securing user data while providing a seamless gaming experience. Furthermore, AWS’s data encryption capabilities both at rest (using S3 and RDS encryption features) and in transit (using SSL/TLS) ensure that user data is protected from unauthorized access and breaches.