

<Name-of-Software-Application>

# **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 | 03/23/24 | Christian Foster | Evaluating the Operating Platform to ensure it can run web- based applications |
| 1.1 | 04/07/24 | Christian Foster | Evaluating the Operating Platform to ensure it can run web- based applications |

**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)

Creative Technology Solutions (CTS) is delighted to embark on a new project with The Gaming Room to develop a web-based version of their popular game, Draw It or Lose It. This game adaptation aims to broaden accessibility by serving multiple platforms, catering to a wider audience. The primary objective is to maintain the essence and excitement of the original game while leveraging web technologies for seamless integration across various devices.

## Requirements

The success of our game development project relies on meeting several crucial requirements to deliver an exceptional gaming experience. Firstly, it's imperative to support multiple teams, enabling various players to participate and fostering engaging multiplayer dynamics. Ensuring that game and team names are unique is essential to prevent confusion and streamline team management effectively. Maintaining a single instance of the game in memory at any given time is paramount for ensuring data consistency and reliability, thus avoiding potential conflicts. Real-time communication capabilities are also crucial, facilitating seamless interaction among players and teams, thereby promoting collaboration and enhancing the overall gaming experience. In addition to functionality, scalability, and performance considerations are vital to accommodate growing user demand without compromising responsiveness. Cross-platform compatibility is equally essential, allowing users to access the game seamlessly across different devices, thereby maximizing accessibility and audience reach. Furthermore, implementing robust security measures, including authentication, encryption, and access control mechanisms, is imperative to safeguard sensitive user data and maintain confidentiality and integrity throughout the gaming experience. Finally, a user-friendly interface with intuitive navigation and clear instructions is paramount for enhancing usability and ensuring a positive experience for all players, regardless of their technical proficiency. By addressing these requirements comprehensively, we can create a game that not only meets but exceeds user expectations, providing an immersive and enjoyable gaming experience for all.

## [Design Constraints](#_2et92p0)

## In transitioning to a web-based environment, we encounter the challenge of distributed computing, requiring seamless communication and synchronization across various devices and locations. To address this, our development efforts will prioritize the implementation of a robust client-server architecture, leveraging advanced technologies such as WebSocket for real-time communication and RESTful APIs for efficient data exchange. Additionally, ensuring the uniqueness of game and team names is essential for providing users with a clear and straightforward experience. To achieve this, validation mechanisms will be integrated into the application's backend, verifying the uniqueness of game and team names during creation to prevent duplicate entries and streamline the user registration process. These measures aim to enhance user experience and ensure smooth gameplay, reinforcing the integrity and efficiency of our web-based gaming environment.

## [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 game application contains six classes with a SingletonTester class designed for testing. The Game, Team, and Player classes, derived from the Entity class, share identical properties and actions of the Entity class. This relationship is indicated by arrows pointing from these classes to the Entity class, visually displayed in the UML diagram. The interaction between the GameService, Game, Team, and Player classes is illustrated as a zero-to-many relationships, meaning that a team can have zero to multiple players, multiple teams can participate in zero to many games, and GameService can manage zero to multiple game instances. This integration demonstrates the interdependence of the application’s architecture.

Each class contains its data along with relationships with other classes. For example, the Entity class acts as a base structure that holds shared properties and actions. The Team class contains the group name, ID, and associated players, while the Game class maintains the name of the game, ID, and the teams it participates in. The GameService class manages game instances.

Distinguishing between private and public methods in each class is an example of encapsulation, ensuring that only important data is exposed and can be accessed by other classes. For example, array lists to store class-specific data, iterate identifiers for elements, and its GameService instance are specified as private to prevent inadvertent changes at runtime. This encapsulation fosters robust and error-resistant application design.

**"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** | macOS Server offers a range of features for hosting web applications, but finding hosting services for macOS Server can be challenging and expensive. macOS Server comes with licensing costs that may be prohibitive for some clients. The pricing structure includes USD 499 for a 10-client license or $999 for an unlimited license. However, the challenge lies in finding hosting services for macOS Server, as it often requires dedicated hardware, leading to additional expenses. Moreover, the licensing costs are recurring, adding to the overall cost of ownership. | Linux distributions are renowned for their stability, security, and scalability, making them an excellent choice for hosting web applications. Licensing costs for Linux servers are typically minimal, as most distributions are open-source and free to use. However, there may be associated costs for enterprise support, extended security updates, or specific software packages, depending on the chosen distribution and support model. These additional costs can vary widely depending on the vendor and level of support required. | Windows Server is a popular choice for hosting web applications, particularly for businesses already invested in the Windows ecosystem. Licensing costs for Windows Server can vary significantly depending on the edition, the number of cores or processors in the server, and whether the client opts for perpetual licenses or subscription-based licensing models. Prices can range from $6,200 (up to 16 core licenses) to $500 (up to 50 clients) per installation per year. Additionally, there may be additional costs for Client Access Licenses (CALs) and other optional features, further adding to the overall cost | Mobile devices, such as smartphones and tablets, are not typically used as primary servers for hosting web applications due to their limited hardware capabilities and lack of scalability. While they can technically serve certain limited functions as personal web servers or file hosts for small-scale applications or personal use, they are not suitable for multi-user serving or enterprise-level hosting. The costs associated with using mobile devices as servers are relatively unknown and may vary significantly depending on factors such as the specific hardware used, any necessary hosting tools or applications, and potential development efforts required to adapt the device for server use. Additionally, there may be hidden costs associated with data usage, power consumption, and maintenance. |
| **Client Side** | **Costing:** Developing for Mac involves moderate to high costs, including expenses for Mac hardware (if needed), development tools like Xcode, software licenses, and testing infrastructure.  **Time:** Development time for Mac applications is moderate, including tasks such as designing responsive interfaces, backend integration, testing across macOS versions, and adhering to Apple's design guidelines.  **Expertise:** Moderate expertise is required, including proficiency in Swift/Objective-C, macOS development frameworks, and knowledge of Apple's development ecosystem. | **Costing:**  Like Mac, Linux development incurs moderate to high costs for development tools, software licenses, and testing infrastructure.  **Time:** Development time for Linux applications is moderate, involving tasks like responsive design, backend integration, testing across Linux distributions, and ensuring web compatibility.  **Expertise:** Moderate expertise is required, including knowledge of web development technologies, backend integration with Node.js, and familiarity with Linux environments. | **Costing:** Developing for Windows requires moderate to high costs for development tools, licenses, and testing resources.  **Time:** Development time for Windows applications is moderate, including tasks such as responsive design, backend integration, testing across Windows versions, and adherence to Windows design guidelines.  **Expertise:** Moderate expertise is necessary, including proficiency in web technologies, familiarity with Windows development tools, and adherence to Microsoft's design principles. | **Costing:**  Developing for iOS and Android incurs moderate to high costs for development tools, licenses, testing resources, and potential app store fees.  **Time:**  The development time for mobile apps is moderate, including tasks such as responsive design, backend integration, testing across various devices/versions, and adherence to platform-specific design guidelines.  **Expertise:**  Moderate to high expertise is required, including programming skills (Swift/Objective-C for iOS, Java/Kotlin for Android), knowledge of respective development frameworks, and understanding of platform-specific UX/UI guidelines. |
| **Development Tools** | **Programming Languages:**  Swift, Objective-C  **IDEs and Tools:** Xcode (Integrated Development Environment), Interface Builder, Cocoa Pods.  **Impact on the Development Team:** The requirement to use Xcode and develop in Swift/Objective-C may necessitate specialized knowledge and skills within the development team. Depending on the project scope and complexity, multiple development teams may be needed, such as front-end developers focusing on UI/UX design and back-end developers handling server-side logic.  **Licensing Costs:** Xcode is priced at USD 99 per year | **Programming Languages:** HTML, CSS, JavaScript, Python, PHP, Ruby, Node.js  **IDEs and Tools:** Visual Studio Code, Sublime Text, Atom, Vim, Emacs.  **Impact on Development Team:** Linux-based development typically involves a diverse set of programming languages and tools. Developers need proficiency in web development technologies and backend frameworks like Node.js. Depending on the project requirements, multiple development teams may be needed to cover front-end, back-end, and system administration tasks.  **Licensing Costs:** Most IDEs and tools for Linux are open-source or have free versions available. | **Programming Languages:** HTML, CSS, JavaScript, C#, .NET Framework  **IDEs and Tools:** Visual Studio, Visual Studio Code, Sublime Text, Atom  **Impact on Development Team:** Development for Windows often involves using Microsoft's development tools and frameworks. Developers need expertise in web development technologies as well as knowledge of the .NET ecosystem for building Windows applications. Depending on the project's complexity, multiple development teams may be required to handle frontend, backend, and Windows-specific development tasks.  **Licensing Costs:** Visual Studio and some related tools may range from $45 - USD 250. This also depends on which feature is used. | **Programming Languages:** Swift, Objective-C (iOS), Java, Kotlin (Android)  **IDEs and Tools:** Xcode (iOS), Android Studio (Android), Visual Studio Code  **Impact on Development Team:** Development for iOS and Android requires specialized knowledge of their respective ecosystems, including programming languages, SDKs, and platform-specific design guidelines. Separate development teams may be needed for iOS and Android platforms, especially for larger projects with platform-specific requirements.  **Licensing Costs:** Xcode is priced at USD 99 for Mac users, while Android Studio is free. However, there may be licensing costs associated with certain development tools and SDKs used for iOS and Android development. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

**Operating Platform**: Linux is the best choice for web software application hosting due to its flexibility, stability, and extensive community support. Additionally, being open-source Linux can be modified and optimized making it appropriate for scalable deployments. Moreover, Linux servers come with robust security features and economical resource management that are vital in maintaining data integrity and system performance.

1. **Operating Systems Architectures**: Linux's modular design and the kernel-based approach contribute to its scalability and stability, making it possible for efficient resource utilization as well as system optimization. The main component of Linux is the kernel, which plays an essential role in managing hardware resources, scheduling tasks, and providing user-space applications with fundamental services. This design promotes flexibility and customization by giving users the ability to add or remove modules at will while still maintaining system functionality.

**Storage Management**: Storage management in Linux is concerned with the different tools and techniques that can be used to streamline how storage resources are organized and managed. Linux offers several file systems such as ext4, XFS, and Btrfs, each having unique features as well as performance characteristics. Logical Volume Management (LVM) gives an added advantage in managing volumes, which includes dynamic resizing, mirroring, as well as snapshots. Data protection and performance enhancement are some of the RAID configurations. Other than this, Linux has made things easy for disk partitioning as well as network file sharing; it can work seamlessly with storage-area networks (SAN) and network-attached storage (NAS) solutions while also allowing for disk encryption which is essential for data protection. Linux offers highly effective and resourceful disk management tools. In a nutshell, Linux is an excellent system to rely on in terms of managing the storage capacity, considering different storage requirements. It guarantees both dependability and the possibility of scalability while protecting the data in such a way that no mistakes could happen about its reliability. This will be helpful if needing more storage for our game.

1. **Memory Management**: Memory management in Linux is a multifaceted process that focuses on efficiently allocating and utilizing system memory resources to ensure optimal performance and stability. Linux employs virtual memory techniques to extend available physical memory by swapping out inactive memory pages to disk storage when necessary, preventing system slowdowns. Various memory allocation algorithms, such as the buddy system and slab allocator, ensure fair distribution of memory resources among user-space processes. Additionally, Linux implements memory paging and swapping mechanisms to handle memory overflow situations, freeing up space for critical system operations by swapping less frequently used memory pages to disk. The kernel manages its own memory space separately from user-space processes, ensuring system stability and security. Memory Management Units (MMUs) are crucial in translating virtual memory addresses to physical addresses and enforcing memory protection mechanisms to prevent unauthorized access to system memory. Overall, memory management in Linux aims to optimize memory utilization, prevent memory-related issues, and maintain system performance and stability.
2. **Distributed Systems and Networks**: Linux has very strong networking capacities as well as a support for distributed computing environments. In this regard, it is equipped with TCP/IP stacks, firewall configurations, and network protocols which ensure effective communication among the various network systems required in web-based application hosting across varying natures of networks.
3. **Security**:   
   Linux security features are vast and beefed up by the availability of many security tools for protecting the server environment. This includes user authentication, access control systems, encryption, intrusion detection systems, and other multiple levels of security in Linux for preventing various threats and vulnerabilities. Besides this, constant updating and patching of security mitigations on Linux servers guarantees that they remain resistant to evolving security risks.