

<Name-of-Software-Application>

**CS 230 Project Software Design Template**

Version 1.0

**Table of Contents**

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 05/22/25 | John Pate | Added Executive Summary, Design Constraint, and Domain model description. |

**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**

The Gaming Room is developing Draw It or Lose It and wants a web-based version of the this gaming app. To meet these requirements we plan on using a scalable, cloud-ready application with unique name validation for the games, teams, and the players, and a singleton pattern to ensure only one active game instance at a time in memory.

**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**

1. Only one game instance can exist in memory at a time so we will need to use a singleton pattern to make sure all users interact with the same game state.

2. Each game, team, and player name must be different so we will need to make sure of this in the backend. This makes things more complex because we will need to constantly check for this.

3. The images will need to be need to be rendered over time and be done rendering at the 30 second mark, this means jthe application will need precise timing and synchronization between all users.

4. The game needs to support multiple teams and players that will all be interacting with each other simultaneously, so we will need a way for them to interact in real-time, and a way to show which team each player is on.

**System Architecture View**

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**

This UML diagram is a model for their web-based game Draw It or Lose It and shows plenty of object-oriented programming principles. Game, Team, and Player all inherit from Entity, GameService manages a list of Game objects, Game has alist of Team objects, and Team has a list of Player objects. The diagram shows encapsulation as each class manages its own data and exposes necessary functionality. As well it shows inhertiance from the base class "Entity".



**Evaluation**

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 offers a stable Unix-based environment with strong security features, making it suitable for development and light-duty hosting, especially for apps made for apple. However, it is less ideal for large scale web hosting due to limited scalability, higher hardware costs, and fewer native server management tools compared to Linux-based systems. | Linux is widely preferred for hosting web based applications due to it being open source, having strong security, flexibility, and efficient performance under heavy loads. Its main drawbacks include a steeper learning curve for and occasional compatibility issues with proprietary software or hardware. | Windows offers a user friendly interface, strong support for .NET applications, and compatibility with a wide range of enterprise software, making it a solid choice for hosting web-based applications. However, it can be more resource-intensive, has higher licensing costs, and is more frequently targeted by malware compared to Linux. | Mobile devices offer portability and accessibility, making them ideal for consuming web based applications on the go. However, they have limited processing power, storage, and battery life compared to traditional servers, and are not well-suited for hosting due to security, performance, and reliability constraints. |
| **Client Side** | Supporting multiple types of clients on Mac requires investing in Apple hardware and enrolling in the Apple Developer Program, which increases cost. Development also demands expertise in macOS tools like Xcode and languages such as Swift, and may take additional time to ensure compatibility with Apple’s design and security standards. | Supporting multiple client types on Linux is generally cost-effective due to its open-source nature and lack of licensing fees. However, it requires developers with expertise in Linux based environments, command-line tools, and possibly multiple programming languages, and extra time may be needed for testing across various distributions. | Supporting multiple client types on Windows often involves higher licensing costs and development tool expenses, such as for Visual Studio or Windows Server. While Windows provides strong IDE support and compatibility, developers need expertise in .NET technologies and additional time for managing system updates, security configurations, and potential integration with legacy systems. | Supporting multiple client types on mobile devices requires expertise in both iOS and Android development, or using cross platform frameworks like Flutter or React Native. Development time and cost increase due to the need for platform specific testing, UI adaptation, and App Store compliance, making mobile support more resource intensive than desktop platforms. |
| **Development Tools** | For deploying web based software on Mac, developers commonly use languages like Swift for native macOS applications, along with JavaScript, HTML, and backend languages like Python, Java, or Node.js for cross-platform web applications. Popular development tools are Xcode for native Mac development and cross platform IDEs like Visual Studio Code, IntelliJ IDEA, or Eclipse for web and backend development. | For developing software to deploy on Linux, common programming languages include C, C++, Python, and Java due to their strong Linux support and versatility. Popular IDEs and tools used are Visual Studio Code, Eclipse, CLion, and command-line tools like GCC, Make, and GDB for compiling, building, and debugging Linux applications. | For building software to deploy on Windows, common programming languages include C#, C++, and .NET languages like VB.NET, as well as Python and Java for cross platform needs. Popular IDEs and tools are Microsoft Visual Studio, Visual Studio Code, and tools like MSBuild and Windows SDK for development, compiling, and debugging Windows applications. | For developing software for mobile devices, programming languages like Swift and Objective C are used for iOS apps, while Java and Kotlin are common for Android development. Popular tools and IDEs include Xcode for iOS, Android Studio for Android, and cross platform frameworks like Flutter and React Native that use languages such as Dart and JavaScript. |

**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**: I believe a appropriate server platform for The Gaming Room would be Linux, specifically a distribution like Ubuntu Server or Rocky Linux. Linux is highly scalable, secure, cost-effective, and widely supported across cloud providers and hardware. It offers strong compatibility with game server technologies, web services, and cross-platform development tools. Additionally, it supports containerization and orchestration, which are useful for deploying the game to various environments like mobile, desktop, and web. This flexibility makes Linux an ideal foundation for scalable game expansion.
* **Operating Systems Architectures**: Linux server distributions such as Ubuntu Server and Rocky Linux share a modular, monolithic‑kernel architecture that is highly customizable. The kernel handles process scheduling, memory, and exposes a POSIX‑compliant system call interface, while separate user space daemons provide networking, security, and storage services. Package managers streamline updates with signed repositories, making the platform stable, secure, and ideal for hosting scalable game services like Draw It or Lose It across diverse computing environments.
* **Storage Management**: An appropriate storage management system for the recommended Linux server platform is Logical Volume Manager. LVM provides flexible disk management by allowing administrators to create, resize, and manage logical volumes across one or more physical disks without downtime. It supports features like dynamic volume resizing, snapshots for backups or testing, and striping for improved performance. LVM integrates seamlessly with Linux file systems such as ext4 or XFS. This flexibility is particularly valuable for a scalable game application like Draw It or Lose It, where storage demands may grow over time due to user-generated content, updates, or logs.
* **Memory Management**: The recommended Linux operating platform uses advanced memory management techniques that support the performance and reliability needs of a game like Draw It or Lose It. The kernel’s memory manager dynamically allocates RAM using paging and swapping, moving inactive pages to disk when memory is tight, while keeping active data quickly accessible. Features like shared memory and memory-mapped files enhance performance by reducing redundant data loading, especially when handling high-definition image assets. Additionally, caching and buffering mechanisms ensure that frequently accessed files, such as images and UI components, are kept in memory for faster access. These techniques collectively help ensure that the game remains responsive and stable, even under varying user loads or system conditions.
* **Distributed Systems and Networks**: To enable Draw It or Lose It to communicate across various platforms, a distributed software architecture can be used where the game logic and data are managed by centralized servers, while clients on mobile, desktop, or web platforms connect via the internet using APIs or WebSockets. This approach ensures real-time communication and synchronization of game states across devices. Middleware and message queues can coordinate events and maintain consistency, while client side caching and graceful degradation allow the game to continue functioning during temporary disruptions. This setup ensures a seamless and responsive cross platform experience.
* **Security**: To protect user information across platforms, a combination of secure communication protocols, data encryption, and platform level security features should be used. All data transmitted between clients and servers should be encrypted using to prevent eavesdropping and tampering. On the server side, the recommended Linux platform offers robust security capabilities such as SELinux or AppArmor for enforcing access controls, firewalls for managing network traffic, and encrypted storage using tools like LUKS to protect data at rest. User credentials and sensitive data should be securely hashed and stored using algorithms like bcrypt or Argon2. Authentication mechanisms such as OAuth 2.0 or multi-factor authentication can add extra layers of security for users. Additionally, regular security updates, system hardening, and logging with intrusion detection tools help maintain a secure environment and protect user data across all platforms.