#### **The Gaming Room**

**Draw It or Lose It**

CS 230 Project Software Design Template

Version 1.2

**Table of Contents**

[**The Gaming Room 1**](#_3zjr7m9geidt)

[Document Revision History 2](#_yk6h2m8wn1ch)

[Executive Summary 2](#_e3qoge7tkzwz)

[Requirements 3](#_wz5wyq2e7zez)

[Design Constraints 4](#_6mqit3f0s1e5)

[System Architecture View 5](#_qfbl12n782q7)

[Domain Model 6](#_ou4kzcm2nnzq)

[Evaluation 7](#_smr0t94r26cs)

[Recommendations 12](#_zbgmf4hk78qz)

[References](#_5hlvnal05b5)14

#### **Document Revision History**

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| 1.0 | 05/26/2024 | Mohamed Elhassan | Initial document creation |
| 1.1 | 06/09/2024 | Mohamed Elhassan | Updated Evaluation |
| 1.2 | 6/23/2024 | Mohamed Elhassan | Updated Recommendations  Final Draft |

#### **Executive Summary**

The Gaming Room aims to transition its Android-based game, Draw It or Lose It, to a web-based platform accessible from multiple platforms. The web-based game will involve multiple teams with unique names and multiple players per team. This design document outlines a solution utilizing object-oriented principles and design patterns to ensure efficient, scalable, and secure development. The key elements of the design include the use of the Singleton pattern to manage game instances and the Iterator pattern to ensure unique game, team, and player names.

#### 

#### 

#### **Requirements**

The client has outlined the following requirements for the game application:

* The game should support multiple teams.
* Each team should consist of multiple players.
* Game and team names must be unique.
* Only one instance of the game should exist in memory at any given time.

#### 

#### **Design Constraints**

Developing a web-based game application presents several design constraints:

* Concurrency and Scalability: The application must handle multiple users concurrently, ensuring efficient resource utilization and responsiveness.
* Distributed Environment: The application must work seamlessly in a distributed environment, managing communication between clients and servers.
* Unique Identifiers: Unique identifiers must be maintained for games, teams, and players to avoid naming conflicts.
* Security: User data must be protected, and secure communication channels must be established to safeguard information during transmission.

#### 

#### **System Architecture View**

TBD

#### 

#### **Domain Model**

The UML class diagram provided represents the structure of the game application and its components. Key classes include:

* **Entity Class:** A base class with common attributes id and name. This class is inherited by other classes to ensure they have unique identifiers and names.
* **GameService Class:** This class manages all game-related operations, ensuring only one instance exists (Singleton pattern). It tracks lists of games and the next available IDs for games, teams, and players.
* **Game Class:** Represents a game containing multiple teams. It inherits from the Entity class.
* **Team Class:** Represents a team containing multiple players. It inherits from the Entity class.
* **Player Class:** Represents a player with unique attributes. It inherits from the Entity class.
* **ProgramDriver and SingletonTester Classes:** These classes are used to run and test the application, ensuring the correct implementation of the Singleton pattern and other functionalities.

Object-oriented programming principles demonstrated in the diagram include:

* **Inheritance:** The Entity class serves as a base class for Game, Team, and Player classes.
* **Encapsulation:** Each class manages its data and operations, ensuring a modular and maintainable design.
* **Design Patterns:** The Singleton pattern ensures only one instance of the GameService class, and the Iterator pattern ensures unique names for games, teams, and players.

#### 

#### **Evaluation**

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| Server Side | Mac servers are known for their stability and ease of integration with other Apple products. However, they are less commonly used for hosting web-based applications due to higher costs and less community support compared to Linux. Mac servers are typically used in environments where integration with other Apple products is crucial, such as creative industries and education. | Linux is the most popular choice for web servers due to its stability, security, and cost-effectiveness. It offers a wide range of distributions that can be tailored to specific needs, extensive community support, and compatibility with a wide range of software. Linux servers are often used in high-traffic environments and are known for their ability to handle large-scale deployments efficiently. Common Linux distributions for servers include Ubuntu Server, CentOS, and Debian. | Windows servers offer good compatibility with other Microsoft products and are user-friendly for administrators familiar with Windows. However, they tend to be more expensive due to licensing fees and can be less stable than Linux servers. Windows servers are often used in enterprise environments that rely on other Microsoft technologies. Common use cases include applications that require integration with Active Directory and Microsoft SQL Server.. | Mobile devices are typically not used for hosting server-side applications due to their limited processing power and storage capacity. Instead, they are best used as clients that access server-hosted applications. Mobile devices are integral in providing accessibility and a wide user base for applications. The focus for mobile should be on optimizing client-side performance and user experience. |
| Client Side | Developing for Mac clients requires expertise in macOS and potentially higher development costs due to the need for Apple hardware and software licenses. However, the macOS ecosystem provides a consistent user experience and high performance. Mac users often represent a high-value user base, particularly in creative and professional sectors. Development tools include Xcode, which supports Swift and Objective-C, and cross-platform tools like Flutter and React Native. | inux clients offer flexibility and cost savings, but developing for them requires knowledge of various distributions and potentially more testing to ensure compatibility. Linux is less common as a client OS for end-users but is popular in tech-savvy and developer communities. Common development tools include Eclipse, IntelliJ IDEA, and various text editors. Popular programming languages include Python, Java, and C++. Cross-platform tools like Electron can also be used. | Windows is widely used on client machines, offering a broad user base and extensive development tools. Development costs are moderate, but ensuring compatibility across different versions of Windows can require additional effort. Windows is prevalent in both consumer and enterprise markets, making it a critical platform for broad accessibility. Development tools include Visual Studio, which supports C#, C++, .NET, and cross-platform tools like Xamarin. | Developing for mobile devices requires knowledge of iOS and Android platforms, potentially increasing development costs and complexity. Mobile clients benefit from widespread usage and accessibility, providing a large potential user base. Development tools for mobile devices include Android Studio for Android development and Xcode for iOS development. These IDEs support Java, Kotlin (Android), Swift, and Objective-C (iOS). Cross-platform frameworks like React Native and Flutter are also popular choices. |
| Development Tools | Development on Mac typically uses tools like Xcode, which supports Swift and Objective-C. These tools are well-integrated with macOS but can be more expensive due to Apple's licensing. Xcode provides a robust environment for building high-quality macOS and iOS applications, including features like interface design, testing, and debugging tools. Additionally, cross-platform tools like React Native and Flutter can be used to develop applications for multiple platforms.. | Linux development often uses open-source tools like Eclipse, IntelliJ IDEA, and various text editors. Languages commonly used include Python, Java, and C++. These tools and languages are free and have strong community support. Linux offers a highly customizable development environment, which can be tailored to specific project needs. Cross-platform development tools like Electron and Qt are also available. | Windows development commonly uses Visual Studio, which supports multiple languages such as C#, C++, and .NET. It is a powerful IDE but can be costly, especially for enterprise versions. Visual Studio offers a comprehensive suite of tools for development, testing, and deployment, making it a popular choice for enterprise applications. Additionally, cross-platform frameworks like Xamarin can be used to target multiple platforms. | Development tools for mobile devices include Android Studio for Android development and Xcode for iOS development. These IDEs are specialized and support Java, Kotlin (Android), Swift, and Objective-C (iOS). Mobile development environments are designed to provide extensive support for building, testing, and deploying mobile applications efficiently. Cross-platform frameworks like React Native and Flutter can significantly reduce development time and costs by allowing code reuse across different mobile platforms. |

#### **Recommendations**

**Operating Platform:** I recommend using Linux as the operating platform for hosting the web-based version of Draw It or Lose It. Linux offers the best balance of stability, security, cost-effectiveness, and community support. It is widely used in the industry for web hosting and provides extensive tools and distributions to suit various needs.

**Operating Systems Architectures:** The Linux architecture will be based on a secure distribution like Ubuntu Server or CentOS due to their stability, security features, and extensive support. Ubuntu Server is known for its user-friendliness and frequent updates, making it suitable for rapid development environments. CentOS, on the other hand, offers long-term support and robustness, which is ideal for enterprise-level applications. Both distributions support Docker, which allows for containerized application deployment, ensuring consistency across development, testing, and production environments. Docker containers encapsulate the application and its dependencies, making it easier to manage and scale the application across different platforms.

**Storage Management:** A combination of local storage and cloud-based storage solutions such as AWS S3 or Google Cloud Storage is recommended. Local storage provides fast access to frequently used data, reducing latency and improving performance. Cloud-based storage offers scalability and redundancy, ensuring that large datasets, such as the game's image library, are always available and can be expanded as needed. AWS S3 provides robust security features, including encryption and access control, while Google Cloud Storage offers high availability and global reach. Both solutions integrate seamlessly with the Linux environment and support automated backups and disaster recovery plans.

**Memory Management:** The Linux operating system employs various memory management techniques to optimize performance. Virtual memory allows the system to use disk space as an extension of RAM, enabling the application to handle larger datasets. Memory allocation strategies, such as the buddy system and slab allocation, efficiently manage memory fragments and reduce overhead. For languages that support it, garbage collection will automatically reclaim unused memory, preventing memory leaks. The application will be designed with proper coding practices, such as avoiding memory bloat and implementing efficient data structures, to minimize memory usage and enhance performance.

**Distributed Systems and Networks:** To facilitate communication between various platforms, the application will use RESTful APIs and WebSockets. RESTful APIs provide a standardized way for clients to interact with the server, ensuring interoperability and ease of integration. They support HTTP methods (GET, POST, PUT, DELETE) and can be secured using OAuth for authentication and JWT for session management. WebSockets enable real-time communication, allowing the server to push updates to clients instantly. This is crucial for features like live gameplay updates. The network infrastructure will include load balancers to distribute traffic evenly across servers and failover mechanisms to maintain connectivity during outages. Monitoring tools will track network performance and detect issues early.

**Security:** Security measures will encompass data encryption, secure authentication, and authorization mechanisms. Data encryption will be implemented using strong algorithms like AES-256 for data at rest and TLS for data in transit. Secure authentication mechanisms, such as OAuth 2.0, will manage user identities, while JWT will handle session tokens securely. Regular security audits will identify vulnerabilities, and patches will be applied promptly. The Linux platform's built-in security features, such as SELinux and AppArmor, will enforce mandatory access controls and limit the capabilities of processes, reducing the attack surface. Secure coding practices, like input validation and output encoding, will prevent common vulnerabilities like SQL injection and XSS.

#### References

AWS. (n.d.). Amazon S3. Retrieved from<https://aws.amazon.com/s3/>

Brighthub. (n.d.). A look at the advantages and disadvantages of Mac OS. Retrieved from <https://www.brighthub.com/computing/mac-platform/topics/mac-os/>

Google Cloud. (n.d.). Google Cloud Storage. Retrieved from <https://cloud.google.com/storage>

HubSpot. (2023). Top tier choices: The 10 best Linux web hosting providers of 2023. Retrieved from<https://blog.hubspot.com/website/best-linux-web-hosting>

Javatpoint. (n.d.). Linux Memory Management. Retrieved from<https://www.javatpoint.com/linux-memory-management>

LinuxForDevices. (2020). Memory Management in Linux – How to Manage Linux Memory. Retrieved from<https://www.linuxfordevices.com/tutorials/linux/memory-management-linux>

LinuxHall. (2024). Understanding Memory Management for Better Linux Performance. Retrieved from <https://www.linuxhall.org/memory-management>

LogicMonitor. (2023, February 28). 9 reasons Linux is a popular choice for servers. Retrieved from<https://www.logicmonitor.com/blog/9-reasons-linux-is-a-popular-choice-for-servers>

Oryon Networks. (n.d.). Why experts choose Linux: 15 benefits of Linux servers. Retrieved from<https://oryon.net/blog/why-experts-choose-linux-15-benefits-of-linux-servers/>

RedSwitches. (n.d.). Linux vs Mac: How they compare and contrast. Retrieved from <https://www.redswitches.com/blog/linux-vs-mac-how-they-compare-and-contrast/>

SoftwareKeep. (n.d.). Choosing the right operating system: Windows, macOS, or Linux? Retrieved from <https://softwarekeep.com/help-center/windows-macos-or-linux-choosing-the-right-operating-system>

Tutorialspoint. (n.d.). Process Memory Management in Linux. Retrieved from<https://www.tutorialspoint.com/process-memory-management-in-linux>

WebsitePlanet. (2024). 7 best Linux hosting services (fast & secure) - Tested in 2024. Retrieved from <https://www.websiteplanet.com/blog/best-linux-hosting-services/>

Wipro. (2021, June). How vertical integration boosts security in Apple products. Retrieved from<https://www.wipro.com/innovation/how-vertical-integration-boosts-security-in-apple-products/>