

Draw It or Lose It

# CS 230 Project So昀琀ware Design Template

Version 1.0

**Table** **of** **Contents**

[**CS** **230** **Project** **So昀琀ware** **Design** **Template** 1](#_bookmark0)

[**Table** **of** **Contents** **2**](#_bookmark1) [**Document** **Revision** **History** **2**](#_bookmark2) [**Executive** **Summary** **3**](#_bookmark3) [**Requirements** **3**](#_bookmark4) [**Design** **Constraints** **3**](#_bookmark5) [**System** **Architecture** **View** **3**](#_bookmark6) [**Domain** **Model** **3**](#_bookmark7) [**Evaluation** **4**](#_bookmark8) [**Recommendations** **5**](#_bookmark9)

**Document** **Revision** **History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 04/17/2024 |  | Updated 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

The mission of The Gaming Room is to adapt the existing Android-exclusive game, “Draw It or Lose It,” into a web-based application that caters to a variety of platforms. The game involves multiple teams, each with several players, who participate in a series of four rounds, each lasting one minute.

During each round, an image is selected from an extensive library and one team attempts to guess the image until the time runs out. If the initial guess is not correct, the opposing team is given a 15-second window to guess. This project aims to enhance the gaming experience by making it more accessible and engaging for users across various platforms.

# Requirements

* The game must have the ability to have one or more teams involved.
* Each team must have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time.
* The game must consist of four rounds of play lasting one minute each.
* The game needs to be developed as a web-based application to serve multiple platforms.

# Design Constraints

* With multiple teams playing at the same time, the game should be able to handle concurrent requests at one time.
* The game should be able to handle an increasing number of users without lowering performance.
* Ensuring the security of user data and transactions is crucial in a web-based setting. Appropriate security measures should be put in place to safeguard users.
* The game needs to be compatible with various platforms and browsers.
* The game should be easy to use and engaging for the end-users.

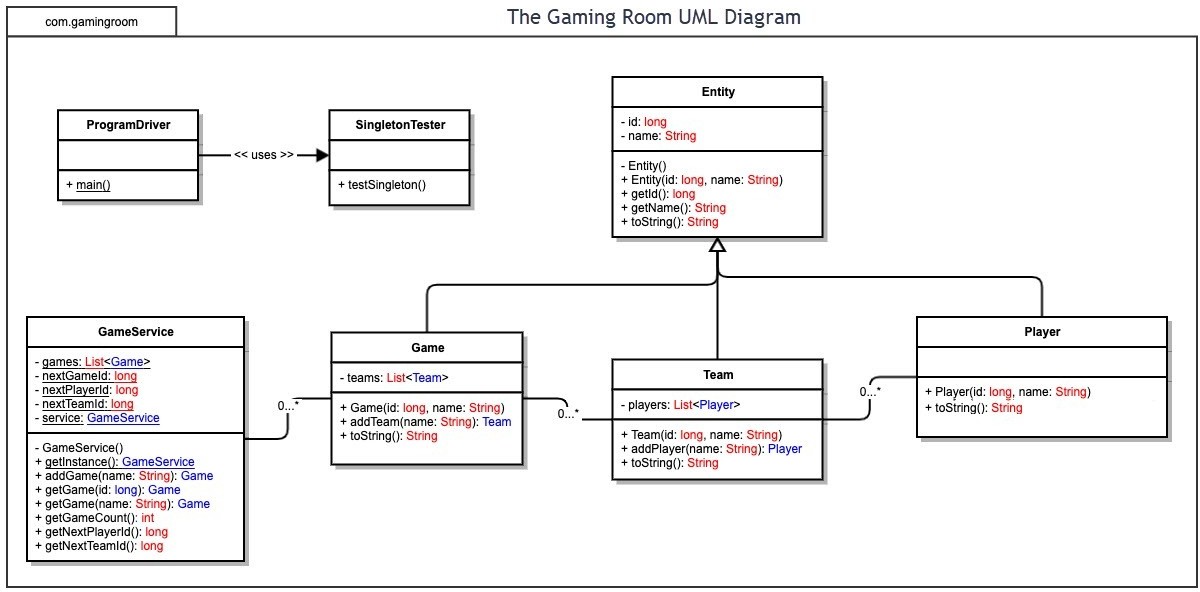
# System Architecture View

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

The UML diagram provides six classes including ProgramDriver, SingletonTester, Entity, GameService, Game, Team, and Player. These classes demonstrate several principles of object-oriented programming. The Game and Team classes are subclasses of the Entity class, representing the principle of inheritance. This principle promotes code reusability as these subclasses can access the Entity class’s attributes and methods. The diagram also shows the principle of aggregation, or the “HAS-A” relationship. This is seen where the GameService class has multiple Game objects, the Game class has multiple Team objects, and the Team class has multiple Player objects. The GameService class seems to implement the Singleton

pattern, indicated by its self-referencing attribute. This pattern ensures that only one instance of the class exists in the application. Lastly, encapsulation is demonstrated where each class has private attributes and public methods. This principle hides the internal states of a class and exposes only necessary operations. These principles together contribute to e昀케cient and maintainable software design.



# 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 昀椀ndings 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 | Available for | O昀昀ers server- | Comes with | Mobile platforms |
|  | Apple’s Mac and is | based | licensing costs | typically do not host |
|  | free with Mac | deployment | that can range in | web-based |
|  | hardware. It is | methods and is | price. It can host | applications. |
|  | capable of hosting | free. It is capable | web-based | Instead, they |
|  | web-based | of hosting web- | applications and | connect to servers |
|  | applications and | based applications | scale up to | hosted on other |
|  | scaling up to | and scaling up to | thousands of | platforms. Licensing |
|  | thousands of | thousands of | players. | costs vary |
|  | players. | players. |  | depending on the |
|  |  |  |  | speci昀椀c Mobile |
|  |  |  |  | Device Management |
|  |  |  |  | solution and service |
|  |  |  |  | provider. |
| **Client** **Side** | A responsive | The application | A responsive | The application |
|  | design is necessary | would need to be | design is crucial to | would need to be |
|  | to ensure | designed | ensure | designed with a |
|  | compatibility | responsively to | compatibility | mobile-昀椀rst |
|  | across di昀昀erent | ensure | across di昀昀erent | approach, ensuring |
|  | browsers and | compatibility | browsers and | it functions correctly |
|  | devices. This might | across di昀昀erent | devices. This | and provides a good |
|  | involve additional | browsers and | might require | user experience on |
|  | time for testing | devices. This | additional | both iOS and |
|  | and optimization | would involve | development time | Android devices. |
|  | to ensure a | considerations for | and expertise in | This might require |
|  | consistent user | di昀昀erent screen | various web | expertise in mobile |
|  | experience across | sizes and browser | technologies to | web development |
|  | di昀昀erent browsers | capabilities. The | ensure the | and testing on |
|  |  | development | application | various devices and |
|  |  | process might | functions | screen sizes. |
|  |  | require expertise | correctly on all |  |
|  |  | in responsive | supported |  |
|  |  | design principles | browsers. |  |
|  |  | and testing across |  |  |
|  |  | multiple |  |  |
|  |  | browsers. |  |  |
| **Development** | Common | Languages like | Commonly used | For Android, Java is |
| **Tools** | languages include | Java, Python, and | languages include | commonly used |
|  | Java, Python, and | JavaScript are | Java, Python, and | with Android Studio |
|  | JavaScript. Popular | commonly used. | JavaScript. Eclipse | as the preferred IDE. |
|  | IDEs include Eclipse | Eclipse and IntelliJ | and Visual Studio | For iOS, Swift is |
|  | and IntelliJ IDEA. | IDEA are popular | are popular IDEs. | commonly used |
|  | Multiple | IDEs. Depending | Depending on the | with Xcode as the |
|  | development | on the platforms | platforms, | preferred IDE. |
|  | teams may be | targeted, multiple | multiple | Depending on the |
|  | needed for | development | development | platforms, multiple |

# Recommendations

Analyze the characteristics of, and techniques, speci昀椀c to various systems architectures and make a recommendation to The Gaming Room. Speci昀椀cally, address the following in the Recommendations section of the software design document:

1. **Operating** **Platform:** I recommend Windows Server for “Draw It or Lose It”. It is a reliable, scalable, and secure platform that can support the game’s expansion to various computing environments. This server provides a wide range of features bene昀椀cial for gaming applications, including support for the latest networking and security protocols. It also o昀昀ers excellent performance and scalability features, crucial for a growing player base. Furthermore, its seamless integration with Microsoft technologies like Azure and support for .NET, commonly used in game development, make it an ideal choice.
2. **Operating** **Systems** **Architectures**: The Windows operating system uses a hybrid architecture. This architecture is divided into two main components: user mode and kernel mode. User Mode is where applications and subsystems run. It has no direct access to hardware or kernel system data structures, which helps protect the system from crashes and security breaches. Kernel Mode is where the core of the operating system runs, including the kernel, device drivers, and system services. It has direct access to system data and hardware. This architecture allows Windows to provide a high level of performance, security, and compatibility with a wide range of hardware and software.
3. Storage Management: Windows incorporates several built-in utilities for managing storage, such as Storage Sense and Disk Cleanup. These tools are designed to help maintain the system’s storage by removing unnecessary 昀椀les, subsequently freeing up space. These tools, integrated within the Windows operating platform, provide a thorough method for managing storage. They guarantee the e昀昀ective use of storage resources, a key factor for the uninterrupted functioning of “Draw It or Lose It” on the chosen platform.
4. Memory Management: Windows uses a Paged Virtual Memory management system. It segments memory into “pages” and assigns each application, like “Draw It or Lose It”, its own virtual address space. This makes the application believe it has access to a large block of memory. When an application needs memory, the operating system maps virtual memory pages to physical ones. If physical memory is full, Windows can swap infrequently used pages to disk storage, freeing up physical memory. This process, along with the isolation of each application’s memory from others, allows “Draw It or Lose It” to run e昀케ciently and securely on Windows.
5. Distributed Systems and Networks: For “Draw It or Lose It”, we’ll be leveraging a client-server distributed system. This approach allows us to customize each client application according to the advantages of its speci昀椀c system. The game’s success relies on the ability of multiple clients to connect to a single server for a cohesive gaming experience. Therefore, a strong server network is crucial. We must also consider factors such as network delays, server outages, scalability, and security. All these considerations are vital in designing a distributed system for our game.
6. **Security**: For security, we can rely on Windows Defender, which is an integral part of any Windows system, providing built-in protection. In addition to Windows Defender, we should also consider implementing additional measures. For instance, we can use secure communication protocols to protect data during transmission. Regular security audits and penetration testing can help us identify and address potential vulnerabilities. Furthermore, in the event of a security breach, having a response plan in place is crucial. This can help us quickly

identify, contain, and recover from the incident, minimizing its impact. These strategies can signi昀椀cantly reduce risk and enhance the overall security of our game.