

# **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 |
| --- | --- | --- | --- |
| 1.3 | 06/18/24 | Jonathan Skop | Third Iteration  Evaluate the pros and cons of the development process on Windows, Linux, Mac, and mobile devices. The evaluation section aims to pinpoint the specific strengths and weaknesses of each platform regarding the application architecture recommendations for different operating systems. |

## 

## [Executive Summary](#_sbfa50wo7nsh)

The proposed software design document presents my plan for creating the web-based game "Draw It or Lose It" for our client, The Gaming Room. This game is based on the popular TV show "Win, Lose, or Draw," where teams guess drawings. Our solution involves using stock drawings as clues and allowing multiple teams to play four rounds. By bringing the game to the web, we aim to make it accessible across different devices and provide an enhanced gaming experience. The proposed design considers the client's needs and seeks to deliver a user-friendly and engaging gaming application.

## 

## Requirements

1. The game must be accessible via a web-based platform to ensure compatibility across different devices and operating systems. Each game should accommodate the participation of one or more teams, with the capability to assign multiple players to each team. Game and team names must be unique to avoid conflicts and enable users to verify name availability during team creation.
2. The application should restrict the number of game instances in memory, allowing only one game instance at any time. This can be accomplished by implementing unique identifiers for games, teams, and players. Furthermore, game rounds should be subject to specific time limits, such as one minute per round, with drawings progressively revealed until fully completed at 30 seconds.
3. If a team fails to guess the puzzle within the allotted time, the remaining teams should be allowed to submit one guess each within a 15-second time limit to solve the puzzle.

## [Design Constraints](#_2et92p0)

* The game application must be developed for a web-based platform, introducing constraints related to network communication, security, and compatibility with various web browsers and devices.
* The system must enforce the uniqueness of game, team, and player names to prevent naming conflicts and provide an optimal user experience when creating and joining games. Additionally, the design must consider the limitation of only one instance of the game service being active in memory at any given time to ensure the game operates as intended.

## [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.**

## 

## The diagram illustrates the Draw It or Lose It game application system classes. The Entity class functions as the base superclass for all entities in the application, encompassing common attributes such as 'id' and 'name' shared by all subclasses. This implies that every entity in the system will possess an identifier and a name. Subsequently, the Game, Team, and Player classes, serving as subclasses, extend the Entity superclass and represent the primary entities within the game application: a Game comprises multiple Teams, and each Team encompasses multiple Players.

## The GameService class maintains a composition relationship with the Game class, signifying its role in managing the lifecycle of Game instances and holding references to multiple Game objects. Similarly, the Game class exhibits a composition relationship with the Team class, and the Team class, in turn, maintains a composition relationship with the Player class.

## The ProgramDriver class serves as the primary function location, functioning as the application's entry point. Within the ProgramDriver class, the creation of a GameService singleton instance indicates the presence of only one instance of the GameService class throughout the application. The ProgramDriver class is responsible for adding games, teams, and players utilizing the GameService instance. Additionally, the ProgramDriver class demonstrates a dependency on the SingletonTester class, denoted by the <<uses>> arrow.

## The UML class diagram effectively portrays several object-oriented programming principles: inheritance, encapsulation, and abstraction. The diagram prominently illustrates the inheritance relationship between the Entity superclass and its subclasses (Game, Team, and Player), allowing subclasses to inherit attributes and behaviors from the superclass. For instance, the subclasses utilize the super keyword to access the Entity's constructor function, effectively reducing code duplication and ensuring consistency. Furthermore, the GameService class encapsulates its attributes, such as the constructor and the list of existing games, providing methods solely for interacting with the data. This approach ensures data privacy, facilitates the creation of only one instance, and promotes abstraction. Additionally, abstraction is exemplified through the classes' management of games, teams, and players, effectively concealing unnecessary implementation details and emphasizing essential interactions.

## [Evaluation](#_2o15spng8stw)

Using your experience to evaluate each operating platform's characteristics, advantages, and weaknesses (Linux, Mac, and Windows) and mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, consider your client’s requirements and look at the situation holistically; it all must work together.

| **Development Requirement** | **Mac**  **Advantages**  High reliability, strong security, excellent graphics and multimedia capabilities.  **Weaknesses**  The available hardware options are limited, and the licensing costs are higher in comparison to Linux. Additionally, this server operating system is less widely used. | **Linux**  **Advantages**  The system possesses capabilities such as low cost, high reliability, robust security, and the ability to manage high server loads.  **Weaknesses**  New users may encounter a substantial learning curve, and potential compatibility issues with specific hardware may arise. | **Windows**  Advantages  The system has a high level of user familiarity, is supported by extensive documentation, and offers a wide array of hardware options.  **Weaknesses**  Windows incurs higher licensing costs in comparison to Linux, and is more susceptible to security vulnerabilities, necessitating frequent updates and maintenance. Higher licensing costs compared to Linux, more prone to security vulnerabilities, requires frequent updates and maintenance | **Mobile Devices**  **Advantages**  **Android** boasts a substantial user base, an open-source platform, and compatibility with a wide array of devices and screen sizes.  The **iOS** platform is characterized by robust security features, a streamlined user interface, and enhanced optimization tailored specifically for Apple devices.  **Weaknesses:**  **Android:** Fragmentation arises due to the existence of various versions of the Android OS, leading to potential security vulnerabilities attributed to its open-source nature.  **iOS:** Due to Apple’s stringent guidelines and App Store fees, limited hardware options and elevated development costs are encountered. |
| --- | --- | --- | --- | --- |
| **Server Side**  **Yes for all**  **Advantages** | **Mac**  Unix-based system provides a stable and secure foundation for hosting web applications. It has a robust ecosystem and is developer-friendly, but it comes with a relatively higher cost for hardware and has limited scalability compared to Linux and Windows.  High reliability, strong security, excellent graphics and multimedia capabilities. | **Linux**  Open-source operating systems allow for customization, flexibility, and a wide range of software and tools. They are highly scalable, known for their stability and security, but may have some GUI limitations and hardware compatibility issues. | **Windows**  Wide range of software compatibility, strong developer ecosystem, broad hardware support, and robust documentation. However, it has more known security vulnerabilities. | **Mobile Devices**  "Portability, touchscreen, and gesture-based interaction with limited screen size, as well as varied hardware capabilities." |
| **Client Side** | Intuitive and user-friendly interface reduces the learning curve. Developing and maintaining multiple clients increases development costs, time, and may require diverse expertise. | Free to use and distribute, but consider additional costs (hardware, tools). Steeper learning curve. Diverse expertise required for different clients. | The costs of obtaining a license may be higher when compared to open-source alternatives. | Important considerations include responsive design, connectivity limitations, and native features such as camera, GPS, and push notifications. |
| **Developmen**  **t Tools** | Node.js and JavaScript are commonly used. IDEs such as VSCode and Xcode. | Rich ecosystem includes integrated development environments (IDEs) such as VSCode, Atom, and Sublime Text, as well as robust command-line interfaces and package management systems like apt or yum. | C# and .NET framework are popular languages for developing Windows-based web applications using Visual Studio and JetBrains IDEs. | Kotlin, Swift, Objective-C, Java, and JavaScript. Proficient in Android Studio, XCode, and device emulators. |

**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:** I highly recommend using Linux Ubuntu Server to host Draw It or Lose It on a Kubernetes cloud setup.

**Reason for the recommendation:**

There are several reasons to recommend Linux Ubuntu Server in Kubernetes. It is faster, cheaper, and lighter than Windows. The Linux system is also more organized in the file system (Akash, 2023). Moreover, Linux Ubuntu is a free and open-source operating system, making it more secure and capable of running without installation. Furthermore, with Kubernetes, you can utilize your public, on-premises, and hybrid cloud infrastructure for free to enhance your organization’s deployment efforts (Java point).

**Why I Recommend:**

I highly recommend using Linux Ubuntu Server in a Kubernetes cloud environment due to its numerous benefits. Linux offers superior gaming performance to Windows due to its efficient resource utilization. It also accommodates major GPU brands, including AMD, Intel, and Nvidia, ensuring hardware compatibility and an optimal gaming experience. From a security standpoint, Linux surpasses most other operating systems and facilitates the seamless configuration of client-server systems. Moreover, Linux equips users with command-line tools, enabling faster network backup and connectivity with other systems and servers.

**Below are specific recommendations for the setup:**

1. **Operating Systems** Architectures: The stable and secure Linux kernel enables easy system and hardware requirements segregation within Kubernetes clusters.
2. **Storage Management:** SSD storage solutions are favored over HDD for their expedited asset access, which enhances user experience. Implementing a Kubernetes node for file storage and a NoSQL node for game data and user management is advised.
3. **Memory Management:** A system load watcher is recommended to optimize memory allocation based on usage, thereby effectively managing costs.
4. **Distributed Systems and Networks:** Cloud-based hosting facilitates seamless node migration and automatic server start-up, bolstering the organization and management of system functions.
5. **Security:** It is advisable to employ a role-based security architecture to categorize administrative, game, team, player, and user access, curtailing unauthorized information retrieval.

* **References**
* **Advantages of Linux - Java point https://www.javatpoint.com › advantages of Linux (Linux).**
* **Linux vs. Windows: Which Operating System is the Best? (Akash, 2023)**
* **https://www.analyticsinsight.net/linux-vs-windows-which-operating-system-is-the-best/. Is Linux good for Gaming? – Linux For Devices (Shaw, 2022)**
* **https://www.linuxfordevices.com/tutorials/linux/linux-for-gaming. Ubuntu Features (Java point) https://www.javatpoint.com/ubuntu-features**