

Draw It or Lose It

# **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 | 05/26/2024 | John Bryson | First Iteration |
| 1.1 | 06/05/2024 | John Bryson | Second Iteration |
| 1.2 | 06/09/2024 | John Bryson | Third Iteration |
| 1.3 | 6/23/2024 | John Bryson | Server Side, Client Side/Development Tools |
| 1.3.1 | 6/23/2024 | John Bryson | Reocmmendations |

**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)

This document outlines the plan to build "Draw It or Lose It," a web-based game inspired by the hilarious competition of "Win, Lose, or Draw." Teams will test their artistic and guessing skills using a library of pre-loaded images. Up to four rounds of frantic drawing and guessing await! By bringing this game to the web, we'll make it playable on any device, making it easier than ever to challenge your friends and family. This design focuses on creating a user-friendly and exciting experience that meets all our client, The Gaming Room's, requirements.

## [Design Constraints](#_2et92p0)

## The Gaming Room has an existing Android-based deployment of the Draw It or Lose It. CTS has been asked to extend this to the web and thus the tech stack needs to be compatible with web-deployment - Java has been selected for this purpose. Java being the native Android SDK language should ease this new deployment effort

## [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 application includes a main driver class responsible for initiating the creation of games, teams, and players. The actual creation is managed by the GameService class, which follows a singleton design pattern to ensure that only one instance of GameService exists in memory at any time.

To prevent the creation of additional instances, the GameService constructor is private. The only way to instantiate a GameService is via the getInstance() method, which checks if an instance already exists and only creates one if it doesn't.

Once GameService is active, the driver class can call the addGame() method. This method uses the iterator pattern to avoid creating duplicate Game objects by name. The new Game object is then added to the List games.

Teams can be added to a game using the addTeam() method, which also uses the iterator pattern to prevent adding teams with duplicate names. The new Team object is then added to the List teams.

Similarly, players can be added to a team using the addPlayer() method. This method ensures that no duplicate Player objects are created by using the iterator pattern. The new Player object is added to the List players.

The Game, Team, and Player classes all inherit from the Entity class, which has two protected attributes: id and name. The default constructor of Entity is protected, preventing the creation of null objects and ensuring that only overloaded constructors are used.

The designed UML demonstrates several object-oriented programming techniques. Polymorphism and inheritance are evident in the extension of the Entity class and constructor overloading. Encapsulation and abstraction are also utilized in the methods for adding teams. A Team object cannot be directly created due to the protected constructor, but the addTeam() method allows the user to add a team without needing to know the details of its creation.

**"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** | It is popular in web hosting, user-friendly, and reliable. It is upgradeable and offers various options for different web hosting requirements, along with strong security features. Although it has a smaller user base compared to Windows, it has moderate licensing costs and supports server-based deployment (macOS Server). | It offers high stability and scalability, with security flaws being caught before they become issues, making it the most preferred choice for web hosting services. Additionally, it is cost-efficient. However, it can be difficult to find applications to support specific web hosting needs, and there is a bit of a learning curve. Licensing costs are lower or free. Server-based deployment is supported, and it is commonly used for servers. | It is predominant over other platforms, offering wide compatibility and user-friendly features. It has high resource requirements but benefits from less loading time, a large user base, and extensive software support. However, it is easily susceptible to viruses, posing security concerns. Licensing costs are moderate to high. Server-based deployment is supported (Windows Server). | It is more popular, convenient, and user-friendly. It has a wider reach, better compatibility, and is cost-effective. However, it is selective to various smart mobile devices, has poor security or lack of support, and adheres to strict app guidelines. Licensing costs are low to moderate. Server-based deployment is not applicable, as applications are deployed on Google Play Store or Apple App Store. |
| **Client Side** | It requires moderate expertise, is user-friendly, and reliable, with costs like Windows. | It requires maximum expertise due to its learning curve but incurs minimal cost. | It requires minimal expertise, is user-friendly, and has costs like macOS. | It provides flexibility for clients to see updates from any location but is slightly more difficult to implement compared to other devices. |
| **Development Tools** | Languages  that are not  limited to  HTML/CSS/Ja  vaScript.  oMac  developers  often use  Xcode, Visual  Studio Code,  or Sublime  Languages that extend beyond HTML/CSS/JavaScript. Mac developers frequently utilize tools like Xcode, Visual Studio Code, or Sublime Text. | Linux supports development environments such as Visual Studio, Eclipse, and Notepad++. Languages commonly used on Linux include Python, Java, C/C++, and many others, extending beyond HTML/CSS/JavaScript. | Languages commonly used by Windows developers, beyond HTML/CSS/JavaScript, include C#, C++, Python, Java, and more. Windows developers often rely on tools like Visual Studio and Visual Studio Code for their development needs. | Languages that extend beyond HTML/CSS/JavaScript include Java, Kotlin, C/C++, and Python, which are commonly used in Android Studio for Android development. For web-based components, developers can also utilize standard web development tools such as Visual Studio Code or other integrated development environments (IDEs). |

Based on the analysis of various system architectures and specific techniques, here are recommendations for The Gaming Room:

1. **Operating Platform:** Start with Windows devices due to the abundance of available software, minimal expertise required to begin projects, and reasonable initial costs. Windows provides a broad range of IDEs suitable for game development, ensuring flexibility and accessibility.
2. **Operating Systems Architectures:** Windows offers robust services essential for gaming applications, including comprehensive support for GUI, system resources management, multimedia, messaging, and web services. It supports both user account-based and server-based applications, providing versatility in deployment options.
3. **Storage Management:** Leverage Windows 10's Storage Sense feature to efficiently manage and evaluate hard drive storage. This includes organizing app save locations and utilizing cloud storage for data backup, ensuring project files are secure and easily accessible across devices.
4. **Memory Management:** Implement a dedicated database or library for storing game assets like pictures. Windows supports customizable memory management to store files outside default directories, facilitating organized and secure project development within IDE environments.
5. **Distributed Systems and Networks:** Utilize game development platforms like Unity and Unreal Engine, which support cross-platform development. These IDEs enable game deployment across various operating systems (including web, iOS, and Android), ensuring broad accessibility and cross-play capabilities. Ensure robust server infrastructure to handle large player volumes and mitigate connectivity issues.
6. **Security:** Windows provides built-in security features such as malware and virus protection, which automatically update to defend against evolving threats. Utilize Windows' user account control settings to safeguard sensitive data and ensure continuous protection of user information and system integrity.

By leveraging Windows' robust ecosystem and features tailored for game development, The Gaming Room can effectively manage storage, memory, network distribution, and security aspects of their projects while maximizing compatibility and accessibility across platforms. These recommendations aim to support efficient and secure game development operations from inception to deployment.