

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 9/14/2023 | David King | * Changes made to the cover page. * Executive summary updated * Requirements updated * Design constraints updated |
| 1.1 | 9/27/2023 | David King | * Evaluation updated |
| 1.2 | 10/13/2023 | David King | * Recommendations updated |

**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)

The purpose of the gaming room project is to expand the browser-based game Draw it or Lose it to platforms beyond android. Draw it or Lose it is a multiplayer game in which several teams compete to guess an answer based on generated images.

## Requirements

* A game will have the ability to have one or more teams involved.
* Each team will 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. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

In addition to the requirements listed above:

* The game must be capable of running on multiple platforms
* The game must be web-based

Code requires adherence to best practices to retain scalability and allow for ease of distribution across various platforms.

## [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 UML features the classes Entity, Team, GameService, Game, Player, Program Diver, and SingletonTester. Entity acts as a parent class, and is inherited by the Game, Team, and player child classes. The GameService, Game, Team, and Player have an associative relationship. GameService and Game have an aggregate (has-a) relationship, Game has an instance of Team, and Team has an instance of Player. With this transitive aggregation, active instances (team, game, and player) remain singular. To ensure that this remains true, the SingletonTester class provides an affirmative output.

**"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** | Mac is built with ease of server access and configuration in mind. GUIs on Mac tend to prioritize user friendliness over complexity. Zsh acts a simple and flexible terminal shell for Mac devices. That said, Mac is a less common platform for server hosting. | Very customizable and virtually no-cost, albeit not very user friendly. Bash command shell allows for significant server configuration. While Linux is a widely-used platform for server hosting, advanced technical knowledge may be required to maintain a secure server. | Windows is built with a user-friendly GUI, and has a more robust command prompt system than Mac. Retail prices for windows server licensing may be more expensive than some other platforms. Windows also contains bloatware and frequent updates, which may impact performance or security. | Specifications vary significantly between mobile devices. Even a web-based application approach should keep this variability in mind. Servers should make use of cross-platform frameworks to avoid nativizing. |
| **Client Side** | Client-side Mac UIs are well known for their focus on user-friendliness. Mac as a platform tends to be exclusive, and cross-platform applications with multiple client types may require extra knowledge on the part of developers. | Linux is a very diverse OS, with a myriad of client-side configurations for all sorts of different uses. Compatibility between clients is important, so cross-testing the most popular Linux configurations may be required. | Windows maintains a large user base, and so many users are familiar with the platform. Unlike Mac, however, Windows supports legacy versions. Like Linux, cross-version testing may be required to ensure compatibility. | Different UIs may be required between mobile platforms such as Apple and Android. The same cross-platform framework from the server-side should be applied to the client-side, with variations made based on individual device support. Testing on older supported devices will also ensure a wider user base. |
| **Development Tools** | macOS applications are typically coded in Swift, though older builds might also include code from Swift's predecessor, Objective-C. Java can also be used in macOS applications. | Linux is written in C/C++. However, applications can be written in most languages, including Java. | Windows is written in C++. Like Linux, Windows applications can be written in most languages, including Java. | Mobile devices vary significantly in what languages they accept. However, the most common platforms such as Apple and Android tend to accept languages such as Javascript, Java, Python, and C++. |

## 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**: Windows is the optimal platform to develop Draw It or Lose It. Windows provides the following advantages:

* A wide range of cross-platform development applications and emulators for the command lines of other operating systems.
* It is the largest OS platform based on market share and number of users.
* A set of user-friendly features that are designed compatible with everything from gaming to application development.

1. **Operating Systems Architectures**: The Windows architecture consists of two components, user mode and kernel mode. While in user mode, access to system resources by programs is limited. In kernel mode, access is unlimited. User mode could be considered the front-end of windows, ensuring a seamless user experience while the kernel manages background processes such as memory allocation.
2. **Storage Management**: Draw it or Lose it is a fairly lightweight game with minimal storage requirements. The game itself should be stored locally on an HDD or SSD device. However, online save data could be saved via a cloud solution to prevent a user from tampering with it in their local files. Cloud services such as IMB cloud or Windows Azure provide scalable and cost-effective solutions to online storage needs.
3. **Memory Management**: Windows memory management is admittedly poor compared to platforms such as Linux, due to the infamous amount of bloatware present on the platform. However, most modern Windows devices still provide far more physical memory than the game likely requires. The platform’s use of virtual memory further expands these capabilities. Newer versions of Windows such as Windows 10 and 11 can also reserve portions of hard disks and use them to provide extra ram.
4. **Distributed Systems and Networks**: For the purposes of the game, a client-server distribution will be best. Centralizing connections allows for easier cross play compatibility than something like peer-to-peer. It would also allow for the game to be hosted on a service such as Azure, which already has multi-platform integration features.
5. **Security**: Windows has a built-in and robust security suite premised around user authorization and authentication. Using a service like Azure would also allow for automatic whitelisting of connections. Finally, Azure includes VPN-standard features like double encryption and data obfuscation. It is worth noting that Azure is a highly rated cloud hosting service, but the Azure VPN client is poorly regarded.