

The Gaming Room

# **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 | 11/12/23 | Benjamin Lambert | Initial iteration of software design. |
| 1.1 | 11/26/23 | Benjamin Lambert | Conducted evaluation of Mac, Linux, Windows and mobile devices. |
| 1.2 | 12/10/23 | Benjamin Lambert | Recommendation added based on evaluation of Mac, Linux, Windows and mobile devices. |

**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)

Draw It or Lose It is a game by The Gaming Room that is currently available on Android only. The project The Gaming Room wants accomplished is to move their current game from Android to Web-based so that multiple operating systems are supported. Each game will consist of four rounds lasting approximately one minute each and will be composed of unique team names with multiple players assigned to each team. The application will render an image at a steady rate until fully-rendered at the 30-second mark. The present team will have the ability to guess the puzzle before the time expires. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit.

## Requirements

* Web-based
* 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)

* Web-based version of the gaming app
* Each Team name must be unique
* Only One instance of the game at any given time
* Each team will have multiple players assigned

## [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)

Entity is a superclass for classes Game, Team, and Player. This means that they all inherit attributes from class Entity. GameService, Game, Team, and Player are all associated with one another. In this association, there can be zero-to-many Games to GameService, zero-to-many Teams to Games, and zero-to-many Players to Teams. This means that there is no cap for these classes. A closer look can show how each class has functions that affect one another such as GameService with function addGame() and Team with function with addPlayer(). The ProgramDriver contains the main() method that runs the entire program and calls to each class & function as applicable.

**"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** | MacOS is a Unix-based operating system known for its ease of use, quality hardware, security, and developer friendly environment. However, the drawbacks of this system come mainly from it not typically being used in this setting. Cost will be significantly higher when compared to a Linux operating system. | Linux is great for Server Side/web-based employment. Its Advantages are cost-effectiveness, flexibility (find what fits your needs), and extensive software repository and community support. The only disadvantages of using this system is that there may be a significant learning curve that may be encountered. Those unfamiliar with the Linux environment will have some challenges. | Windows is great for Server Side/web-based employment. Its Advantages are ease of use, supported software, and the .NET Development stack. The disadvantages of using this system is that ther licensing costs may be high, the windows CLI is not as powerful as Linux, and Windows is often more resource-intensive. | Mobile Devices are not typically used in hosting a web-based software application. This is because of the lack of hardware/resources. However, they hold a siginifcant role for clients accessing web-based applications. |
| **Client Side** | Software development for multiple client types using Mac requires that we use a Mac and that could be significant in cost. Also, knowledge of swift is a must to develop for Mac. | Software development for multiple client types using Linux requires a high level of expertise, and time. Cost is not necessarily inflated for this system. | Software development for multiple client types using Windows requires a consideration of cost. Windows, being easy to learn should make setup and development easy. | Software development for multiple client types using Mobile Devices requires finding someone with expertise in mobile app development. Also, depending on the size of the screen of smartphone, the GUI will need to adapt to dufferent aspect ratios than a computer. |
| **Development Tools** | Programing Languages: Swift, Objective-C  Tools: Interface Builder, CocoaPods, Carthage | Programming Languages: C, C++, Python, Java, Rust  Tools: Visual Studio Code, Eclipse, Atom, Make, Git | Programming Languages: C#, C++, Visual Basic .NET (VB.NET), Java  Tools: Visual Studio, Visual Studio Code, Eclipse, .NET Framework/.NET Core | Programming Languages: Swift, Objective-C, Java, JavaScript  Tools: Xcode, Android Studio, Eclipse, Visual Studio |

## 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 recommend using a Linux server to host the game “Draw It or Lose It”. Linux has strong advantages as stated previously that far outweigh it’s disadvantages. Cost-effectiveness, flexibility, and extensive software repository are key to ensuring the most support for “Draw It or Lose It”.
2. **Operating Systems Architectures**: The Linux operating systems is secure, open-source, and can be ran on a wide range of devices. The flexibility and scalability of Linux make it a preferred choice for a wide range of hardware platforms.
3. **Storage Management**: For Linux, one popular and versatile storage management system is Logical Volume Management (LVM). LVM provides a flexible and scalable way to manage storage by abstracting physical storage devices into logical volumes.
4. **Memory Management**: Linux's memory management techniques are designed to provide efficient utilization of resources, adapt to varying workloads, and ensure stable system performance even in resource-constrained environments. The dynamic nature of these techniques allows Linux to handle diverse workloads and maintain responsiveness in different scenarios.
5. **Distributed Systems and Networks**: We can implement a client-server architecture where multiple clients connect to a central server to communicate between various platforms. The server will manage the game state, coordinate interactions, and facilitate communication between clients. Furthermore, Mobile and web compatibility should be at the forefront of our focus. Designing the frontend of the application to include mobile devices and web browsers will provide the client with a father reach which was the initial end goal.
6. **Security**: User information can be protected in several ways on the Linux server such as a third-party security tools, enabling a firewall, secure SSH access, or by implementing access controls. A combination of these measures will help ensure that the system and its data stays secure.