

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 | 10/7/2024 | Armando Meraz | Fully completed design template |

## [Executive Summary](#_sbfa50wo7nsh)

The client’s software design problem is regarding a web-based game, “Draw It or Lose It”, where teams compete in four rounds attempting to guess the puzzle. They have asked specifically to incorporate the ability to have multiple teams each with assigned players and unique ID’s. There can only be one instance of the game in memory. We will accomplish these requests by ensuring we utilize unique identifiers for every instance of player, game, & team.

## Requirements

*From a business perspective the game must be available in multiple platforms, currently it is available only on Android. The game & team players names must be unique and only one game is allowed to exist in system memory.*

## [Design Constraints](#_2et92p0)

The game application has several design constraints such as:

* Available in multiple platforms (Windows, Android, Mac, Linux, etc.)
* Ability to have at least one or multiple teams.
* Multiple players in each team.
* Both Game & Team Names must be unique.
* Only one instance can exist in system memory.
* Java will be the programming language.

The implications of the game design constraints on the application development are that the client wants their game to be available in any platform to reach as many users as possible. It needs to be compatible with mobile & desktop devices and be able to run on Mac, Windows, Linux, or any operating system. Thus, the game needs to be programmed in a language that is compatible across various OS, like Java. The design goal should be to provide a happy user experience, currently the game is compatible only with Android devices. The code that exists will need to be overhauled & refactored to meet the clients needs for multiple platforms/ OS. This will require experienced developers to assure all software requirements are met.

## [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)

There are a total of seven classes described in the UML diagram below. Inheritance is observed since Game, Player & Team classes inherit some of the same properties from the Entity Class. We can also see that classes share a relationship & multiplicity, such as GameService, Team, Game, & Player which are interconnected and share a relationship of zero to many. (0..\*) Based on the UML Diagram we can also observe the teams can have more than one player and that the names of both players and teams are unique. The GameService class has all the getters and setters including the service singleton method and the ProgramDriver class contains the Main() method.

<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.**

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac has great security and is less vulnerable to malware & viruses. Mac Servers typically are more expensive and have a smaller market share compared to Windows. Unix-based foundation which provides various open-source tools. | Linux is more cost effective & is open source as such as there is no licensing fees. Overall is faster compared to Windows OS and boots up and accomplishes jobs more quickly. Downside is the ongoing support for Linux is more expensive. Not compatible with some of Microsoft’s technology. | Windows is more user friendly compared to Linux, which can be more complicated to utilize. Great Scalability which means Windows can quickly adapt to arising needs and can be sued with UNIX. A weakness is the cost is more compared to Linux and the servers can be overloaded to the point of crashing. | User base is larger compared to other OS. Mobile apps can still be used without internet connection. They enhance accessibility & are cost effective. Updates can be pushed to mobile users much more rapidly. The development aspect is more complex and are dependent on whichever App store its hosted on like Google Play Store or Apple Store. |
| **Client Side** | Mac’s OS is expensive, and the ongoing maintenance cost can be quite costly. Can be quite complex to navigate compared to other OS. The software can be designed to meet specific client requirements. | Linux is usually free of charge due to being open-sourced. Linux is used less often than other OS and requires more technical knowledge & experience to navigate. | Windows tends to be more costly compared to Linux. Time required to become experienced is dependent on individuals but is quiet user friendly once you learn it. | Mobile devices are very inexpensive compared to other OS and are easily available anywhere. They are easy to use & navigate, the downside is since there are many OS available it is more complex to implement. |
| **Development Tools** | Swift is Mac’s most used programming language. IDE’s that are compatible with Swift are Xcode, visual studio, textmate, & Nova. Cost is about 100 per year fir Apple Development program | Linux has IDEs like PyCharm, Eclipse, and Atom. C/C++, Java, Python, JavaScript are all programming languages use in Linux. | Windows has IDEs like Android Studio, NetBeans, IntelliJ IDEA. Windows is implemented primarily in C#, Python & Java | Android mobile devices use Android Studio and NetBeans in mostly Java. MacOS tends to lean more towards Objective C & Swift, IDEs used in MacOS is Xcode. |

## 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**: After careful consideration we are recommending Windows as the operating system best suited to expand the game Draw It or Lose It because of the following:

* One of the largest user- based & popular platform available for the development of games and applications.
* Integrates very well with Android services through Android Studio.
* Provides a very rounded comprehensive API that allows the easy creation of applications.
* Strong security features, and a focus with managing memory management efficiently.

1. **Operating Systems Architectures**: Windows Kernel is considered the core of the OS and it takes care of managing process scheduling, access to hardware, memory allocation and many more vital functions. The Windows OS also separates the user access from the kernel which in turn enhances system security. They also support efficient file system management like the New Technology File System(NTFS).
2. **Storage Management**: With a Microsoft environment we are strongly recommending the Azure Blob Storage which allows the seamless integration between Microsoft services & also amazing cloud storage capabilities. Management of access is done via the Azure Active Directory which gives the control of who can and cant access certain data. It is very cost effective due to only having to pay for the storage you will utilize.
3. **Memory Management**: Windows 11 manages Memory Management through their own technique Paging which allows the efficient allocation and management of the system memory. Demand paging is also implemented when the system only loads the pages that will be used by a process, this reduces the load on the system memory. They also employ their virtual memory system which helps further the management of memory by mapping physical memory to the page table.
4. **Distributed Systems and Networks**: The client’s hosted game of Draw It or Lose It can communicate with other different platforms using distributed software which enables the transfer of data among different systems like iOS & Windows & Android. The connection occurs via the networks and connects people from all over the world regardless of location & can fail during outages.
5. **Security**: Security will be implemented by requiring the use of username/passwords to thwart threats and we can even require a two-step authenticator protocol to further defends against attacks. The encryption of data will also be something we do and making sure we are updating software to the latest version. We will also have firewalls and anti-virus software to make sure no one gets in without the proper authorization in place.