

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**

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 4**](#_Toc115077324)

[**Evaluation 5**](#_Toc115077325)

[**Recommendations 6**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/19/23 | Ian Viguera | Completed missing constructors, Created Entity Class. |

**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 aims to be a mobile game consisting of multiple players where they take turns guessing what is being drawn. Each team will have four one-minute rounds to guess a slowly rendered puzzle under 30 seconds. If the team does not guess the puzzle, the opposing team will have 15 seconds to try to solve it for extra points. The client informed that will only be offered on Android platforms.

## Requirements

The technical requirements for this application include: a 1-minute mark for each round, a slowly rendered puzzle within the 30 second mark, hold more than one player for each team, have more than one team, and have an additional 15 seconds after each round. This can be achieved with constructors that overwrite variables after each loop and counter iterations that end round loops after they reach zero. The app must be primarily focused to be in an Android environment.

## [Design Constraints](#_2et92p0)

* Web server to host the web application.
* How to quickly render images.
* Security

## [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 *ProgramDriver* class will serve as its main, or driver class, and it uses constructors declared in the *SingletonTester* class. The *Entity* class directly inherits variables from the *Game*, *Team* and *Player* classes respectfully. Relationships surrounding the *GameService*, *Game, Team*, and *Player* classes are all set to 0 to many multiplicity relations.

**"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** | Although server licensing is cheap to invest in, IOS is the least popular platform to code and the most unpopular for game-based programs. This could lead to potential financial losses caused by a lack of user traffic. | Linux behaves similar to Windows and Android platforms with the exception that it restricts the access of third-party applications. This is a security advantage, but potentially restrictive if the app depends on other apps. One notable advantage is that the server is free, meaning that no investment will result in a loss. | Windows supports most of the development requirements and security measures. It does not have any risky disadvantage that could affect the program or its development. Server pricing ranges between $20 and $200 monthly. The best option is to invest in the cheapest server until it reaches sizable client traffic. | Server licensing is cheap and easy to work on. The downside of mobile environments is that client traffic is highly volatile and could have too many users acting under a cheap server, or less users acting under an expensive server. For this reason, mobile platforms are recommended to be approached as a last resort. |
| **Client Side** | The main benefit is that IOS is not an open source, meaning that expenses would go solely on development and resources. The only downside is that Mac OS has the lowest client usage. | Linux is, in fact, open source. Even if there’s more expenses to be considered, it is easier to develop and it is a platform that adapts to different programming methods and can increase user traffic. | Windows requires the highest investment cost. The upside for this risk is that Windows is the most popular platform for development and user traffic. | Development cost and difficulty are, by comparison, relatively low when working for mobile devices. The downside comes from the highly volatile user traffic that cannot be measured or predicted. This could be a success, but it is not guaranteed. |
| **Development Tools** | Has the capability to run JavaScript programming but has to be enabled. Swift programming is recommended for better app functionality. Since it is the most secure OS, it does not require other security methods that compromise development time. | Since Linux runs JavaScript with no customization needed, only a coding environment is needed, such as Eclipse or Visual Code. For security measures, other programming languages should be implemented. | Adapts to any programming language and has access to multiple resources that would benefit development. Budget control need to be highlighted because of the expensive resource’s costs. | Multiple cheap resources are available that can benefit development. Supports multiple programming languages, especially JavaScript. Microtransactional programs need to be tied to the main game in order to bring profit to our client. |

## 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**:

The best platform to expand The Game Room’s “Draw it or Lose it” is Windows. Windows supports Android development environments, and it is easier to implement them to other platforms. The main reason why Windows is the best option for this particular app is the development tools that it supports. It has the capability to support other coding languages, such as React, which can be used for multi-platform development. One last advantage is its tight security, run by Microsoft cyber security division, that guarantees a safe development environment and reduces cyberattacks and development leaks.

1. **Operating Systems Architectures**:

Window’s architecture consists of Kernel and User mode. The most basic description of this architecture is that User mode is the output that is meant to be shown to and interacted with the user. Kernel mode is what happens behind that interaction, memory management and component communication are prime examples for this mode. This is good for developers due to its low latency that speeds up development time.

1. **Storage Management**:

Modern consoles and personal computers come equipped with at least 16 GB of RAM. Since the application only adopts 8 MB with 200 images, it is expected to run with no game breaking or notable bugs. If The Gaming Room wishes to expand upon the app, added content and updates should be offered as downloadable content. This way it adapts to any platform and the additional storage management relies on the user. In other words, Microsoft’s Azure cloud saving is the best option for storage management.

1. **Memory Management**:

In reference to Microsoft’s official learning webpage, “Each process on 32-bit Microsoft Windows has its own virtual address space that enables addressing up to 4 gigabytes of memory”. Window’s current memory management strategies sits upon one of the fastest and most efficient for loading memory, compared with other operating systems. This kind of management focuses on automatically allocating a small portion of its memory to both the kernel and the user programs equally. It can make “Draw it or Lose it” fast and efficient.

1. **Distributed Systems and Networks**:

Windows uses Network Load Balancing (NLB), which “combines two or more computers that are running applications into a single virtual cluster, NLB provides reliability and performance for web servers and other mission-critical servers”. The servers can also hold multiple platforms simultaneously, supported by multiple development environments. Since Windows is more focused on user experience and development flexibility, it allows for a smooth network between other platforms with no notable restrictions.

1. **Security**:

Windows security is upon the strongest of our current times. Its latest built-in antivirus software accurately detects and prevents malicious software from harming files, documents, or its operating system. This software is updated constantly with better features, security measures and defenses to provide protection to personal computers and servers altogether. Clients and developers can be relieved that its security measures actively protect sensitive data across multiple platforms as well as its own programs. This means that from the development time to further from app launch time, all of its data and sensitive information are fully protected by its latest security measures, made by Microsoft’s cyber security team.