

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 | 02/04/24 | William Tunstall | Initial documentation of game Draw it or Lose it |
| 1.0 | 02/18/24 | William Tunstall | Evaluation of different platforms and making recommendations |
|  |  |  |  |

**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)

Our client, “The Gaming Room,” is looking to create a new web-based game called “Draw It or Lose It.” We are to develop this game to be a web-based version of its counterpart.

## Requirements

The game we are developing must include the following: Games will have multiple teams, including multiple players on each team. The game will only allow unique team names that will ensure that players and teams do not have duplicate names and ensure that there is only one instance of the game existing in memory at one time.

## [Design Constraints](#_2et92p0)

* The game is web-based, meaning that it must be coded in a web-based language to be fully functional in any web-based environment.
* The game must allow for multiple teams and players through the use of team and player objects.
* The game must run checks on game and team names to make sure that each game is unique through the use of a singleton pattern when creating new games and teams.
* There must be only one instance of a game in the memory, by using unique IDs for each unique instance of a game, team, or player.

## [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 Game class, Team class, and Player class all inherit the entity class allowing them to inherit the functions and attributes contained in the Entity class. The GameService class, Game class, Team class, and Player class are all associated with each other. The 0…” notation in between the 4 classes represents that each class can use as many instances of the others as desired or no instances at all. Finally, the ProgramDriver class will drive the package while inheriting from the SingletonTester class.

**"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 known to have user friendly commands that allow for easy server access, configuration, and change.  One notable advantage of Mac is the diverse implementation technologies which negates technical restraints.  Possible the greatest downfall is the expensiveness of the technology. | Linux shares the terminal flexibility noticed in Mac, yet Linux is more cost effective. This is shown through cheaper scalability and lack of licensing costs. There are minimal technical restraints associated with Linux. | The greatest advantage Windows have is the overwhelming support and software available. It contains more supportive software than the other operating systems. Windows licensing is less expensive than Mac, but does not compete with the cost effectiveness of Linux. | A notable trait of mobile OS are the extremely high scalability for web hosting. The downside is that this requires a device with internet connection to support DNS conversion. It is also inexpensive to license. |
| **Client Side** | In order to develop for Mac, one must use a Mac computer running Xcode. Requires niche expertise and a high cost, although comes with support of modern browsers, security, and transmission protocols that prevent session hijacking. | Linux requires a lot of expertise, but the fact that it is an open source OS minimizes costs.  Has some restraints due to lack of support for many file structures. | This OS requires the least expertise, and is less expensive than Mac. The OS does impose some technical design restraints due to focused Microsoft web frameworks. | Mobile devices have flexibility for clients and developers to see updates anywhere they are. Have the advantage of having frameworks that aid in development for specific devices (HTML5 or CSS3) |
| **Development Tools** | Like said before, development for Mac OS must be done on a Mac device in Xcode. The language used is called Swift. | The majority of code done for Linux is in Java. The IDE of choice is Eclipse. | C and C++ are the two most common coding languages used in windows. The popular IDE is called Visual Studios. | iOS and Android code in different languages. iOS similar to Mac uses Swift, while Android typically uses Java. |

## 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**: My personal recommendation is for The Gaming Room to use the Windows OS to develop their web-based game, ”Draw it or Lose it.” The main reasons are that this OS requires minimal expertise to use and has overwhelming software support through a vast library, as well as many IDEs that are available for use.
2. **Operating Systems Architectures**: The Windows API or application programming interface provides services used by all Windows-based applications to allow applications to provide a GUI, access system resources, incorporate audio, etc.
3. **Storage Management**: The application can be stored directly on a user’s device allowing faster load speeds. Cloud support allows for plenty of storage space as needed.
4. **Memory Management**: Each process on 32-bit Microsoft Windows has its own virtual address space that enables addressing up to 4 gigabytes of memory. Each process on 64-bit Windows has a virtual address space of 8 terabytes. Windows utilizes paging, a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory. The mapping between virtual and physical addresses normally is done transparently to the running applications by the memory management unit (MMU), which is a dedicated part of the CPU.
5. **Distributed Systems and Networks**: The Windows operating system provides mechanisms for facilitating communications and data sharing between applications. Collectively, the activities enabled by these mechanisms are called interprocess communications (IPC). Some forms of IPC facilitate the division of labor among several specialized processes. Other forms of IPC facilitate the division of labor among computers on a network.
6. **Security**: Windows 10 and 11 include Windows Security, which provides the OS’s latest antivirus protection. Windows has been known to provide some measures of security, virus and spyware detection and threat level assessments. Most users opt to use added protection to secure their data and information. Third party tech security companies. In all fairness, the hardware itself from Windows comes pre-equipped with protection. This system can scan for malware, viruses, and security threats. Real-time algorithms help the security software keep up with environmental and technical changes that occur with technological advancements.