

# **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 | 8/17/24 | Ryan Erno | Completion of Document |

**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)

Current Client: The Gaming Room is looking to develop a web-based game that can serve multiple platforms and is designed based on their current game, Draw It or Lose It. To fulfill this, it will be important to maintain most of the original code from the original game and build upon it, so the code won't have to be written several times. This will include adding abstract classes, and specific design patterns like Simpleton and Iterator.

## Requirements

*Business Requirements*

* *Maintain core mechanics from the original game*
* *Needs to serve multiple platforms*
* *A streamlined and efficient design process*
* *Allows multiple players and teams*

Technical Requirements

* Only one instance of the game can exist in memory and will need a system to support this
* Will need to be able to maintain unique identifiers for team and player names per instance.
* Code needs to meet best industry practices(readable and maintainable)
* Team and player names must be unique to check if they are already in use.

## [Design Constraints](#_2et92p0)

* Getting the Game to be compatible across platforms will require a specific design approach to work.
* Simpleton pattern needed to allow only one instance in memory.
* Needs to be able to handle multiple users at once without performance issues.
* Since it is a game it will need to have high maintainability as future updates and improvements are assumed.

## [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)

* Class Entity will contain all the necessary variables and characteristics needed throughout the program (‘’Id’ and “name) it will encapsulate this data and will thus utilize inheritance.
* Game class will inherit from the Entity class, it will represent the instance of the game and contains the team information along with game information like names and Id’s. Utilizes encapsulation by having specific characteristics for teams and inheritance from Entity.
* Team Class inherits from the Entity class and will represent a team and a player list. It encapsulates its class-specific characteristics and utilizes inheritance from the Entity class.
* Player class inherits from the Entity class and will represent the player with names and id. It encapsulates its class-specific data and utilizes inheritance from the Entity class.
* GameService class manages the instance of the game and utilizes the simpleton pattern to ensure one instance is used at a time.
* ProgramDriver Runs the main part of our code, it will manage in sequence the functioning of our classes and methods.
* SingletonTester will simply test the operation of the singleton pattern ensuring in operates as intended.

**"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 good for web-based applications, it is typically stable and secure and integrates particularly well with Apple products however Mac hardware is typically more expensive is less common in usage and has less support. | Linux is widely used for server-side hosting of apps and is cost-effective along with a wide range of support and compatibility between platforms, it tends to be however more complicated to use and learn. | Windows is a widely common platform and is relatively user-friendly along with an extensive support system however Windows is less versatile and is seen as less secure in terms of security. | While portable and easy to use, mobile platforms are not typically used for web-based applications because they have limited capabilities and resources, and usage would require too many design constraints. |
| **Client Side** | Usage of Mac would be higher in costs and would have a less extensive support system therefore expertise in this system would be rare and specialized. Cross-platform ability is also limited. | Linux is cost-effective and is typically versatile between platforms however It is less intuitive than other platforms and will require specific expertise to use. | Windows is the most widely used platform. Expertise will be diverse as will its support system. Server licenses can be expensive to acquire but is much more intuitive to use. | Expertise will be needed as mobile platforms are limited in ability both in resource and processing affecting time efficiency and cost effectiveness. |
| **Development Tools** | Relevant programming languages include Swift and Objective-C. Deployment would use tools such as Xcode, IDEA, and Atom. | Relevant programming languages include Python, C++, and Java. Deployment would use such tools as Eclipse, Emacs, and NetBeans | Relevant programming languages include C#, C++, Java, Python, and a large variety of others. Deployment would use Eclipse, Unity, and PowerShell. | Relevant programming languages include Java and Kotlin. Deployment would use Flutter, React Native, and Android 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. My Recommendation for our client The Gaming Room is Linux, Linux is not only a free open-source system that helps reduce overall costs in licensing, but it is designed to run efficiently and reliably. It is also an immensely flexible system able to be integrated and run on various types of platforms from common hardware to mobile phones and is historically used within servers because of its exceptional flexibility. In addition, it is made up of a multiuser system, allowing multiple users to access its resources simultaneously while allowing for multi-processing and protection between them. With these advantages, Linux is a necessary option for efficiently running a web-based application.
2. I would suggest that this Linux recommendation utilizes a monolithic kernel structure, this means that all systems and services like file management, memory, and use of drivers are run in kernel space meaning all are located within a single address allowing for simplicity and high performance, there is no separation between user level and kernel level processes, calls then can be made directly to the kernel which negates the overhead that is usually received with message passing. That being said, and as mentioned before Linux is also a multiuser-based system meaning that many users are allowed to access its resources at a given time this is done by connecting multiple machines to a mainframe in which the machines act like clients because this is a game application, multiuser access is necessary for its operation.
3. The Linux Ext 4 File system is my recommendation to the client, it is the most recent and extensive version of the Ext family to date. Ext 4 is also among some of the more extensively tested systems therefore giving an advantage in terms of stability and reliability. Ext 4 also is known to perform better than for example(BTFRS) and typically has faster read and write speeds. Lastly, because of its maturity as a tested system it has a large inventory of tool support for any data recovery needs, it also contains a method known as journaling which allows it to identify if and where data is out of order. This is important since the edge goes to BTFRS in this respect. I think Ext 4 will exceptionally meet the client’s needs.
4. Linux has a variety of memory management techniques, one of these is virtual memory this is where a memory address that may be used by a program gets mapped to a physical address in computer memory, this allows a system to compensate for low RAM or Memory shortages. Another is Paging which means the system manages its memory in smaller blocks, not only does this naturally reduce fragmentation but allows for programs that may be larger than memory to run. Lastly, is swapping allowing the system to discard to any unused pages to make room for others to load into memory thus improving efficiency.
5. For the Application to be able to communicate between platforms it could utilize APIs (Application Programming Interface) which will allow clients from various platforms to interact with the application in the same manner, from here the application could be modularized into microservices which communicate through said API’s, allowing certain parts of the application like user authentication and scoreboard to be deployed independently. Connectivity can be done with WebSocket which would allow for a stable and consistent data exchanges between client and servers and then the game could be deployed on a Cloud network which allows for a large global user base to access it with good connection which, could be further improved with added connected for redundancy if case it was to fail.
6. Some ways to provide security to the client is with firewalls which Linux itself has tools included for, these are useful for controlling the specific “in’s” and “outs” of traffic anything else is denied. Encryption is another method, it takes sensitive data and converts it into an unreadable form which can only be converted back through specific means, another useful tool related to this would be use of a VPN which are a popular choice these days, this will allow for encrypted communication between clients and servers, Linux is capable of supporting this tool as well. Authentication methods like multi factor authentication can be used thus making a user need an addition mean of accessing the system like getting a code emailed or messaged to the relevant user. Lastly is the Linux SELinux(Security-Enhanced Linux) feature included within its kernel which allows admins to control access to certain system resources for certain users, Known as MAC(Mandatory Access Control). These methods and Linux features ensure that our client and it application is well protected.