

Draw It or Loose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc175070487)

[Table of Contents 2](#_Toc175070488)

[Document Revision History 2](#_Toc175070489)

[Executive Summary 2](#_Toc175070490)

[Design Constraints 3](#_Toc175070491)

[System Architecture View 3](#_Toc175070492)

[Domain Model 3](#_Toc175070493)

[Evaluation 3](#_Toc175070494)

[Recommendations 6](#_Toc175070495)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/11/2024 | Johnny Jones | Make the website accessible on mobile app. User friendly with iPhones & Android. |
| 2.0 | 07/20/2024 | Johnny Jones | We updated the requirements, executive history, design constraints and Domain Model. |
| 3.0 | 08/17/2024 | Johnny Jones | Analyzed the characteristics and techniques specific to various systems and made recommendations. |

## [Executive Summary](#_sbfa50wo7nsh)

“The Gaming Room” company currently owns a game called “Draw It or Loose It”. The game successfully operates on a mobile app, but the company is looking to expand to a web-based environment and needs our help. The objective of the game is for multiple teams to compete to guess what picture is being drawn. The game will consist of four rounds lasting one minuet each.

## [Design Constraints](#_2et92p0)

* Game allows multiple players.
* Each team will have multiple players assigned to it.
* Game & Team names must have unique, this will help users check if a name is in use.
* Only one instance of the game can exist at any given time.

## [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)

As demonstrated in the UML diagram, Entity is the super class in which the Game, Team, and Player class all inherit from. This allows the subclasses to reuse the common attributes without repeating them each time. Encapsulation is present as the attributes in each class is private with access provided. The Game Service class uses the singleton pattern ensuring only one instance of Game Service exists.

**"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.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | MacOS can be used for web development but may not be the best choice to do so. The reason I say this is because MacOS is not made for large high traffic programs. It can however be a great option for medium size projects.  There is no additional license cost, but the hardware can be expensive. | Linux is useful for web hosting due to the performance, stability, and flexibility that it can offer. It supports major web server software and technologies. It supports all major web server software and technologies.  There is no additional cost to use Linux however, there is optional cost for enterprise support. | Windows can be a good option for hosting web applications. It provides good support for integration with Microsoft services.  The license cost is usually based on the usage such as when using cloud. You can also sometimes pay as you go. The makes the pricing flexible. | Mobile platforms usually do not host web applications. There are some limitations that pretty much eliminates this as an option such as limited processing power, issues with the network dropping, and possibly the battery dying.  Cost wise, Android and iOS do not charge for license as they are not designed for serving as a operating system. |
| **Client Side** | The cost to support multiple clients using MacOS would be a higher initial investment which could also require ongoing maintenance.  The time frame could be extensive as we must take into consideration testing and deployment time to ensure compatibility with Apples guidelines.  This would require skilled developers who is familiar with MacOS technologies, and design system. | The cost to support multiple clients using Linux would be a lower initial investment due to the availability of tools.   The timeframe would be shorter than using MacOS but would still require testing and deployment to ensure it is compatible with various Linux environments.  This would require developers experienced with Linux technologies, principals, and design. | The cost to support multiple clients using Windows would be a higher initial investment due to licensing costs.  The timeframe can vary. Again, it would require testing and deployment time to consider compatibility with windows.  The project would require skilled developers proficient in Windows who is familiar with Microsoft design principals. | An initial investment would need to be made for development tools, licenses, hardware, with on going maintenance.  Significant time would be needed for testing and deployment specifically for iOS and Android.  This would require skilled developers in iOS and Android to ensure we keep the project moving along successfully and in a timely manner. |
| **Development Tools** | Languages required to build this type of software are C/C++, Swift, or Python for specific tasks.  Multiple development teams may be needed as the project could require ongoing maintenance.  There is no license costs related to the development tools. | Languages may require C/C++, Python, Java, or JavaScript.  I feel that Linux may be the best option to complete this gaming application on multiple platforms.  These development tools are open source meaning there are no additional fees to use them. | Languages that can be used to complete this task are C/C++, C#, Python, Java, or JavaScript.  All languages will not be needed but you have a variety to choose from. The good thing is these programs are free to use.  No license needed | Languages to complete this task may include Java and Kotlin for android development. Swift and Objective – C for iOS development.  This presents challenges as you must switch platforms to complete the tasks for The Gaming Room. This can cause the project to be delayed.  No license is needed to complete this. |

## 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 research, the operating platform that I recommend for The Gaming Room to use is Linux. Linux is highly stable, secure, and very flexible. It is widely used in gaming environments which means it is already compatible for high memory and storage usage. Since Linux support a wide range of tools and frameworks it can make it easier when it comes to expansion. As we identified in previous research, there is no licensing cost which would be beneficial as the application scales.
2. **Operating Systems Architectures**: Different programming languages, databases, and frameworks can be used to code Draw It or Lose It using Linux. It is compatible with all major cloud platforms and offers well known security features to ensure it is protected. It is easy to manage and reliable.
3. **Storage Management**: The storage management system that I would recommend for this type of game is MongoDB. MongoDB able to handle large amounts of data and multiple players. This is important as The Gaming Room have already identified the player and storage needs as well as the opportunity for growth.
4. **Memory Management**: Linux uses various memory management techniques that is capable to program the game Draw It or Lose It. Memory Mapping or mmap may be used to load large images or files. Stack and Heap management is important for quick performance and loading. Shared memory will allow multiple processes to access the same memory and allow fast communication between them. Garbage collection can automatically get rid of memory that is no longer needed which could also help with storage. There are also others. I like that Linux can use a combination of these techniques to ensure that the game operates at its best performance for users.
5. **Distributed Systems and Networks**: There are different network considerations that will allow Draw It or Lose It to communicate through various platforms. Connectivity wise, Cross-platform communication can be used to ensure protocols are supported on each of the platforms, while latency management may use client-side prediction and lag compensation to ensure the game is able to be played smoothly. When it comes to outages, we can design the game so that it can switch to offline mode in case of a outage using graceful degradation. A retry logic can also be used to prevent the network from automatically trying again which could overwhelm it. Linux also allow the use of redundancy to ensure the availability in case of a server failure.
6. **Security**: To protect user information on and between various platforms, Linux allows us to use data encryption which encrypts sensitive information such as passwords, personal, and financial information. We could also incorporate SMS bases verification codes. Data anonymization could protect users data so even if it was compromised, it could not be traced back to the users. By using a combination of these practices plus others that Linux provides, we can ensure the users of Draw It or Lose It have a secure platform to game on.