

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 | 01/20/2023 | Caitlin Samotus | Initial document |
| 1.1 | 02/03/2023 | Caitlin Samotus | Reviewed the Evaluation section. Incorporated feedback from first submission. |
| 1.2 | 2/19/2023 | Caitlin Samotus | Reviewed the Recommendations section. Incorporated feedback from the second submission. |

## [Executive Summary](#_sbfa50wo7nsh)

The client, The Gaming Room, would like to develop a web-based version of their game, Draw It or Lose It. Currently, the game must be played on an Android OS device. The function of the game application is to render stock library drawings for members of a team to guess the clue within 30 seconds. An instance of the game must allow more than one team to participate, though the teams need to be unique. Each team can have multiple players, though each player needs to be unique as well. The goal is to identify the best operating platform to develop the web-based game and ensure the functionality aligns with the client's needs.

## Requirements

* The game must be adapted to a web-based version.
* The game must be able to render stock images to completion in 30 seconds (no more, no less).
* The game must allow for one or more teams to participate in an instance of the game.
* The game must be able to toggle between teams for guesses.
* Teams will be made of multiple players.
* Each game instance must be unique. Users will have the ability to check the Game Name to ensure it is available.
* Each team must be unique per game instance. Users will have the ability to check the Team Name to ensure it is available.
* Each player must be unique per game instance. Users will have the ability to check the Player Name to ensure it is available.

## [Design Constraints](#_2et92p0)

* The game must be web-based.
  + Which operating system has the best capability to develop web-based applications?
  + Is there a programming language that works best for that operating system?
  + Does that programming language have the capabilities to develop all the game requirements?
* Only one instance of a game can exist at a time.
  + The code will need to have a singleton pattern to ensure only one instance of a game can be created based on unique game name.
* Teams must be unique within a game instance.
  + The code will need to have a singleton pattern to ensure only one instance of a team can be created based on unique team name.
* Players must be unique within a game instance.
  + The code will need to have a singleton pattern to ensure only one instance of a player can be created based on unique player name.
* A timeline and budget have not been provided for this work. These are critical elements to ensuring the approach is acceptable.
  + What is the budget? Is it sufficient to cover the level of effort?
  + What is the timeline? Can the work be completed within this constraint?

## [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 driver class is ProgramDriver that runs the program, and it implements the SingletonTester class.
* The Entity class is the parent class. Game, Team, and Player all have an inheritance relationship with the Entity class, meaning that they have access to all of the variables and methods in the Entity class in addition to their unique attributes.
* Each of the classes in the parent/child group also exhibits polymorphism (or operator overloading) with the toString() method.
* The Entity class is also an example of abstraction because it will never be directly called by the ProgramDriver or GameService class. It will be called through one of the child class methods only.
* The GameService class has the private GameService method to implement the singleton pattern and allow only unique instances of Game to occur.
* All private fields and methods (denoted by “-“) are instances of encapsulation. These exist but are not directly accessible by the user or non-inheritance relationships.
* GameService has a 0 to many relationship with Game, meaning GameService can have as many Games as needed or no Game at all.
* Game has a 0 to many relationship with Team, meaning Game can have as many Teams as needed or no Team at all.
* Player has a 0 to many relationship with Player, meaning Team can have as many Players as needed or no Player at all.

**"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** | Capable of running on all webservers (Chrome, Edge, Safari, etc.).  Able to multitask efficiently. | Linux is used to support the web and Android OS.  Free hosting capabilities.  Linux can be used on many hardware types. | Least capable of working cross-platform.  Works best on Microsoft-based servers.  Data sharing is a concern, as are system bugs. | Can be used as a web server, though the device type would limit the compatibility.  There are also licensing concerns with servers on mobile devices. |
| **Client Side** | MacOS devices are expensive, so unless the client already has these, it would not make sense to develop their game on this platform.  The platform is great for creative projects, though not as flexible for customization. | Steeper learning curve.  Not all devices work with Linux without additional modifications, though Linux is available in both Mac and Windows devices alongside other OS. | Windows OS devices are widely available to all users and are most likely what the client has given that the game already works on Android. | The client likely has mobile devices available for employees.  Easy to work with on the go, but generally not preferred to a traditional laptop or desktop computer for productivity. |
| **Development Tools** | Support is available.  IDEs are readily available.  Works well with C++ and Java, which would be the preferred language to build the web-based game. | Free and flexible for development.  However, not for inexperienced developers as there is little support.  IDEs are available, but not as robust as Mac or Windows.  Works well with C++ and Java, which would be the preferred language to build the web-based game. | Support is available.  IDEs are readily available.  Works well with C++ and Java provided the right software is installed, which would be the preferred language to build the web-based game. | IDEs are available, though limited, and may not be the typical options available on other OS.  Given the small physical parameters of a mobile device, it would be less than ideal for development. |

## 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 recommended operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments is Linux. Given that the existing version of the game is on Android, it was likely developed using Linux to begin with. Linux can be used on devices that have MacOS or Windows OS devices easily, which makes it a versatile option for the client. It is an open-source platform that can also be cost-saving for the client in both the development and hosting aspects of the game.
2. **Operating Systems Architectures**: The main architecture of Linux consists of hardware, kernel, shell, applications, and utilities (Javapoint, n.d.a). The hardware represents the physical computer on which the OS is running. The kernel layer is the fundamental functioning of Linux OS that converts the physical hardware into virtual actions and resources. It also ensures processing conflicts are mitigated. The shell is essentially the user interface. The applications and utilities are functions implemented by the OS.
3. **Storage Management**: Linux operates with block storage and maintains files with a file hierarchy. Linux is available to use with both cloud and physical storage. This will allow for a hybrid solution of both cloud and physical storage for the application, which is the ideal approach (Cleo, 2023).
4. **Memory Management**: Linux uses virtual memory storage in conjunction with demand paging within the kernel layer (temporary storage on the hardware) (Javapoint, n.d.b). Memory is accessed by a user as needed and the physical location is not provided to the user.
5. **Distributed Systems and Networks**: Linux is a primary building block of web servers, so it is the ideal platform to build a web-based game, such as Draw It or Lose It. Mac, Windows, and Android are all compatible with the Linux OS, so the game can be pushed to any platform desired by the client. It is open-source and primarily virtual, which makes it portable. It is highly secure and supports multiprogramming and multi-user functions. For a client who wants to be able to do it all, Linux supports this across many devices.

1. **Security**: Though Linux is an open-source platform, it is considered very secure. Root access is not provided to all users, which protects the core application. Memory access is done virtually, so the physical address is not provided, and therefore is not accessible to the average user. Logs are maintained for administrative purposes to see who touched what and when in a very transparent way. And finally, being open source means that a lot of individuals have access to the code and are invested in it working properly. The community of users works as a neighborhood watch so that malintent rarely slips through the cracks (Emertxe, 2022).

References

Cleo, T. (2023). *Blog: On premise vs. cloud: Key differences, benefits and risks*. Cleo. Retrieved February 12, 2023, from https://www.cleo.com/blog/knowledge-base-on-premise-vs-cloud

Emertxe (2022, January 19). *Why is linux more secure than other operating systems?* Emertxe. Retrieved February 19, 2023, from <https://www.emertxe.com/embedded/why-linux-is-more-secure-than-other-operating-systems/#:~:text=Linux%20systems%20are%20rarely%20infected,many%20reasons%20associated%20with%20it>.

Javapoint (n.d.a). *Architecture of linux - javatpoint*. www.javatpoint.com. Retrieved February 19, 2023, from <https://www.javatpoint.com/architecture-of-linux>

Javapoint (n.d.b). *Linux memory management - javatpoint*. www.javatpoint.com. Retrieved February 19, 2023, from <https://www.javatpoint.com/linux-memory-management#:~:text=The%20subsystem%20of%20Linux%20memory,programs%20and%20kernel%20internal%20structures>.