

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

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

[Table of Contents](#bookmark1) 2

[Document Revision History](#bookmark2) 2

[Executive Summary](#bookmark3) 3

[Design Constraints](#bookmark4) 3

[System Architecture View](#bookmark5) 3

[Domain Model](#bookmark6) 3

[Evaluation](#bookmark7) 3

[Recommendations](#bookmark8) 5

## [D](#bookmark9)[ocument Revision History](#bookmark9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 06/18/22 | Maya Neely | Adding to 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.

## [E](#bookmark10)[xecutive Summary](#bookmark10)

The Gaming Room wants to develop a web-based game, available on multiple platforms called Draw It or Lose It. The game is currently only available on an Android app. The game has one or more teams with multiple players assigned to a team. The game consists of four rounds, each lasting one minute in which the guessing team has to guess the puzzle within the minute or the opposing team has the ability to guess with a 15 second time limit. The app will render images from a library of stock drawings as clues and the drawings are rendered at a steady rate and are fully complete at the 30-second mark.

## [Design Constraints](#bookmark11)

* A game will have the ability to have one or more teams involved. -Business constraint
* Each team will have multiple players assigned to it. -Business constraint
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name. -Business constraint
* Only one instance of the game can exist in memory at any given time. -Technical constraint
* Has to run on multiple platforms(with the exception of Android since this app already exists). -Technical constraint

The business constraints of the program are requirements asked of the client. These decisions can be changed by the client so meeting these requirements is a matter of including them into the code. The technical constraints are the most challenging to abide by since these design decisions typically can not be changed. For example to make the app available on all platforms, the code should inherit other languages from the existing code either already completed by the Android app (which we do not have) or by creating our own.

## [System Architecture View](#bookmark12)

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.

## [D](#bookmark13)[omain Model](#bookmark13)

Entity is the superclass or parent class of the program with Game, Team and Player deriving from it, or subclasses. The use of Entity prevents repetition throughout the code meaning each of the subclasses will share common functions inherited through the superclass. This means OOP principles encapsulation and inheritance are present. Aggregation is also found throughout out the diagram since GameService has a reference to 0 or many Games, Games has a reference to 0 or more Teams and with Teams there is a reference to 0 or more players. We also see that ProgramDriver uses class SingletonTester but neither is directly connected to the rest of the code.

**"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.**

## [E](#bookmark14)[valuation](#bookmark14)

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** | Advantages:   * Flexible terminal * Upgradeable * Secure   Disadvantages:   * Least preferred for web hosting services * Limited hardware options * Content must be reviewed by Apple | Advantages:   * Secure, making it most preferred * Stable * Reliable * Open source * Supports most programming languages   Disadvantages:   * Most unfamiliar platform | Advantages:   * Very commonly used platform * Most available software compared to Mac/Linux * Low loading time   Disadvantages:   * Least secure OS * Closed source | Advantages:   * Most commonly used platform, wider reach to audience * Flexible * Compatible   Disadvantages:   * Hardest OS to code * Lengthy * Not very secure * Selective to certain smart phones |
| **Client Side** | Cost:   * Generally the most expensive operating platform   Time:   * Depends on development requirements   Expertise:   * Moderate | Cost:   * Cheapest/ free option   Time:   * Depends on development requirements   Expertise:   * High, most unfamiliar platform | Cost:   * The middle option between the three   Time:   * Depends on development requirements   Expertise:   * Moderate | Cost:   * Cheap   Time:   * Lengthiest Process between all other options   Expertise:   * High, lots of unfamiliar skill required |
| **Development Tools** | * Notepad++ * Swift * Mac OS X * Python * HTML * CSS * Java * JavaScript * Ruby | * Mostly all languages * Visual studio * Eclipse * Notepad++ | * Mostly all languages * Visual studio * Eclipse * notepad++ | * Android * Swift * HTML * CSS * JavaScript * Java * Python * PHP * C++ |

## 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**: From my evaluation of all platforms, the operating platform I recommend for The Gaming Room is Linux. Linux is the most secure, stable, and reliable platform while also being the most successful open-source operating system. The production cost is fairly low or nonexistent while also supporting most programming languages.
2. **Operating Systems Architectures**: For the creation of Draw It or Lose It, a cloud-based, server less, architecture should be used. A server less architecture allows developers to focus more on the software and features of a game rather than its server and infrastructure. “The servers are maintained and managed by cloud providers, not developers, allowing developers to focus more on goals and bettering the application. Server less also reduces the cost of development, so you’re only charged for the number of executions made (Bashir ,2019).”
3. **Storage Management**: Since most of Linux storage options are hardware, the most appropriate storage management system would be cloud services. Storing the application on hardware can be costly and if the hardware were to malfunction or fail, storage to the application will most likely be lost. “Cloud storage stores apps on the internet that can easily be retrieved through any device with the proper login giving the user access to these apps “anytime, anywhere” (Eaton, 2021)”. The user can be reassured that their data is protected on the cloud’s server. Also, the amount of storage on the user's cloud is easily upgradable by paying more in a subscription a month or by deleting data periodically.
4. **Memory Management**: “Virtual memory and demand paging are memory management techniques used in Linux (Jithin, 2016).” Virtual memory adds to available memory , in turn, making the system appear to have more memory. When Draw It or Lose it is executed using the Linux os, an instance of the app will be loaded into the RAM. “This is the process of allocating the memory for the instructions to execute. When the program demands more RAM than available, it will be allocated to the virtual memory. This prevents the program from lacking the necessary RAM to execute (Jithin, 2016).” Demand paging on the other hand, generates page faults that impact performance when not managed effectively. “Only the demanded pages of memory are being swapped from the secondary storage to the main memory (Jithin, 2016)” which keeps thrashing low.
5. **Distributed Systems and Networks**: Distributed systems is an efficient way to allow Draw It or Lose It to be able to communicate between different platforms. This allows “a group of computers working together to appear as a single computer to the end-user while sharing the same state (Kozlovski, 2018).” These multiple nodes can easily share data with one another, allowing for easier scalability. Another advantage of distributed systems is that “Failure of one node does not lead to the failure of the entire distributed system and other nodes will still be able to communicate with each other(Distributed).”

A RESTful API should also be considered. RESTful API’s are stateless just as HTTP’s which are both used via internet connection for client-server interaction. Component interfaces and the messages shared between them are initiated by the client and remain the same when delivered to every different client. This allows for session data to easily be supported. Load balancing in distributed systems allows for overall better connectivity. If one of the multiple servers is down, the load balancer directs clients to functioning servers ultimately preventing little to no downtime.

1. **Security**: The most common approach to preventing unauthorized access is passwords. A correct username and password must match the one stored in the system for the system to identify the user as the owner of the account. The problem with common passwords is that they can easily be guessed or exposed, making the computer system more vulnerable to an attack. In order to counter this, encrypted, one-time passwords or two-step authentication could be used. “A firewall could also be used to limit network access between two security domains and monitors and logs all connections.”

Linux OS has built in firewalls and a variety of other features making it more secure than other platforms. One being its superior user privilege model meaning “Linux users have low automatic access rights and require additional permissions to open attachments, access files, or adjust kernel options, it is harder to spread malware and rootkits on a Linux system (Day, 2021).” Their open-source code is another advantage allowing world-wide user developers to eliminate threats and vulnerabilities quicker than any other platform that keeps its code hidden from unauthorized personnel.

Citations

Bashir, Faizan. (2019, July 5). What is Serverless Architecture? What are its Pros and Cons? Retrieved June 19, 2022, [https://www.freecodecamp.org/news/what-is-serverless-architecture-what-are-its-pros-and-cons/](https://www.freecodecamp.org/news/what-is-serverless-architecture-what-are-its-).

Day, B. (2021, October 25). *How secure is linux?* Linux Security. Retrieved June 19, 2022, from <https://linuxsecurity.com/features/how-secure-is-linux>

*Distributed Systems*. Tutorials Point. (n.d.). Retrieved June 19, 2022, from <https://www.tutorialspoint.com/Distributed-Systems#>

Eaton, K., & Wendel, J. A. (2021). *What is cloud storage*. Amazon. Retrieved June 12, 2022, from <https://aws.amazon.com/what-is-cloud-storage/>

Jithin. (2016, November 9). *What is virtual memory and demand paging?* interserver.net. Retrieved June 19, 2022, from https://www.interserver.net/tips/kb/virtual-memory-demand-paging/

Kozlovski, Stanislav. (2018, April 27). A Thorough Introduction to Distributed Systems. Retrieved June 19, 2022, <https://www.freecodecamp.org/news/a-thorough-introduction-> to-distributed-systems-3b91562c9b3c/