

Draw It or Lose it

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 10/16/2022 | Jonathan vest | Introduction of software design and constraints |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has requested the development of a web based multi-player, multi-team game loosely based on the game *Win, Lose Or Draw.* The game will be named Draw It or Lose It and will consist of multiple teams with multiple players on each team. Each game and team will require a unique identifier and name to ensure that only one instance of the game is executing at any given time. The game will render drawings at a set rate and a team will have thirty seconds to guess the image. If the team does not correctly guess the image before time expires the remaining teams will have the opportunity to offer one guess each to solve the puzzle within a fifteen second time limit.

## [Design Constraints](#_2et92p0)

* Ability to have one or more teams – The game must be developed in such a way that will allow it to have a single or multiple teams involved with each game. The teams must also be unique, no duplicate teams will be allowed.
* Images and Library - The game must have a large library of stock drawings to choose from. Depending on the platform the game is being designed for we must consider the constraints the platform will place on the images themselves. When using images that will be published on the web we must also consider copyright and licensing issues for the images.
* Only one instance of the game – Game design must allow only one unique instance of a game to exist in memory at any given time.
* Platform – Determining what platform the game will be played on will impact game design. Whether it be an application on a mobile device or from a web browser on a desktop computer, determining what platform the game will be built for will determine memory limitations, frame rates, screen size etc.

## [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)

Our Domain Model will consist of five main Classes outside of the Driver and SingletonTester classes. The Entity class will serve as a base class for our subclasses to inherit from. The Game, Team and Player classes will be derived from the Entity base class and will therefor extend Entities attributes and properties to those classes. The Game, Team and Player classes will share a zero to many relationship, meaning each Game can have from zero to many teams, and each team can consist of zero to many players. Our Domain Model will also consist of a GameService class. This class is not derived from Entity but does share a zero to many relationship with the Game class, meaning that each GameService can have from zero to many games as long as they have a unique identifier. The game service class is a singleton class meaning that it does not contain a public constructor. Instead, it utilizes the getInstance() method. This will ensure that there can only be one instance of the game in memory at any given time.

The Object Oriented principles displayed in this UML diagram include inheritance, encapsulation, polymorphism and portability. Inheritance is demonstrated by the Game, Team and Player classes inheriting attributes and methods from the base Entity class. Encapsulation is demonstrated by the private data members and methods represented in the UML diagram by the minus sign depicted in front of the variable or methods name. Polymorphism is demonstrated though the overloaded constructors depicted in the UML diagram by the same constructor name being passed different parameters. Polymorphism is also represented by the need of the subclass to override its parent toString() method. Finally, Portability is demonstrated by the use of the Entity class. The Entity class can be used as a base class by any number of other objects outside of our gaming room domain.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | User friendly UI. If using other Mac components, makes creating of Mac clients very easy.  Relatively secure.  More expensive Option.  Not the best option for scaling. | Can be difficult navigate the platform if the user is not familiar.  Cost effective option.  Very secure. | User friendly.  Most popular platform on the market.  Scalable.  Large selection of available software.  Expensive licensing.  Not a secure as Linux or Mac. | Inexpensive.  Specifications vary between user devices. Must meet specifications for each type platform(Android, IOS, Windows Phone) |
| **Client Side** | More secure than windows. If using other mac platforms, this makes integration easier.  Very Expensive and proprietary.  Limited flexibility, not ideal for scaling. | Open Source.  Very secure.  Cost effective.  Requires expertise to be efficient with the platform.  Not compatible with all applications made for Mac or Windows. | User friendly, low expertise needed.  Good choice for scaling.  Large selection of available software.  Not as secure as Mac or Linux.  Expensive and proprietary. | Provides flexibility to clients.  Extensive time required for support.  Requires skilled developers, more difficult to implement. |
| **Development Tools** | Mac’s are able to run all available languages but the most popular on Mac is swift. Xcode is Apples IDE for macOS and is compatible with C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit (Rez), and Swift. | Linux is compatible with all major languages including Jave c++, and python and is compatible with a large number of IDE’s including Eclipse, PyCharm, Rider, Code::Blocks and NetBeans to support those languages. | Windows is compatible with all major programming languages including Java, c++, c#, Python, R and many more. There are a large number of IDE’s available for the windows platform including Visual Studio, Eclipse, NetBeans, PyCharm, Code::Blocks and many more. | There are a countless number of applications that can be developed for the mobile platforms Android, IOS and WP. The software for developing an application and supporting these applications can be run on any of the three operation systems Mac OS, Linux and Windows. |

Recommendations

1. **Operating Platform**:

I would recommend a serverless cloud-based architecture running the Linux OS. A serverless architecture offers cost benefits and process agility. This will be ideal for scaling the application as the company grows and the popularity of the game increases.

1. **Operating Systems Architectures**:

With the serverless solution the cloud provider, i.e. google cloud, AWS, Azure, is responsible for dynamically managing the allocation and provisioning of servers. This takes the responsibility off The Gaming Room to manage and ensure the proper functionality of the servers away from the company so they can concentrate on the development of the application. The cost benefits of the serverless architecture are also substantial as The Gaming Room will not need to employ a team to maintain the servers. That cost can now be allocated to other areas. Scaling for a serverless architecture is also automatic and virtually seamless (Bashir, 2019b). Services can be accessed through methods such as API’s or HTTP and allows for distribution operations across varying computing environments. Serverless architectures build off the three-tier model where a middle layer can be leveraged to deliver features such as security services and content delivery.

The Linux OS has the largest installed base of all general-purpose operating systems (OpenSource.com, 2018). The Linux OS uses integrated control features to deliver resources with security and modularity. The Linux kernel contains the core components while offering separate modules and services to promote expansion.

1. **Storage Management**

For the fastest load times, best virtualization, and overall best user experience I would recommend the use of direct access storage. Direct access storage allocates each physical record with its own discrete location and unique address. This makes access of the data stored in that location much quicker over a sequential storage device as there is no need to read through intervening records until you find the record required.

Linux offers a file system structure that uses mulit-tiered indexed allocation. This provides each memory location a unique identifier (Unix/Linux..., n.d). These identifiers are referenced across multiple index tiers to conserve memory space by avoiding large, single indexes (Silberschatz, 2008). Linux looks to store files on blocks near their index to reduce the amount of time needed to find the file on disk once the reference is known. (Silberschatz, 2008).

1. **Memory Management**:

Paging is a function of memory management where the computer stores and retrieves data from a device’s secondary memory into the device’s main memory. The basic method for implementing paging breaks physical memory into fixed size frames and breaking logical memory into fixed sized blocks called pages. When the process executes its pages are loaded in any available memory frame from the secondary memory location. Linux employs partial memory execution by taking advantage of virtual memory and demand paging. This eliminates physical memory size constraints and allows multiple programs to run in parallel. Partial memory execution also loads and runs the app faster.

1. **Distributed Systems and Networks**:

For Draw it or lose it to meet the clients’ requirements of operating on various platforms they are going to need to implement load balancing and replication. As the application grows and sees increased client use load balancing will ensure the even distribution of client access across the servers. Deploying load balancers will also provide an extra layer of security. Most cloud providers also offer load balancing and there are several algorithms that a network administrator can choose from that will best meet the systems needs.

Load balancing on a distributed system will also help guard against connectivity issues. If there would to be an outage such as application server going down or a network related issue that renders the application or database server unreachable, the load balancer will automatically recognize the outage and direct traffic to the other available resources.

Running on a distributed system means that the application and database will be running on several different machines simultaneously. Replication will be vital to ensure that the databases remain in synch. Replication refers to the process of copying data from a primary or master database to one or more secondary or slave database. This process usually happens in real time and is an ongoing process (What Is Database Replication? 3 Main Types., n.d.).

1. **Security**:

As with any system that wishes to protect its user and client information authentication and authorization will be key. A model of protection should include client side and server-side authentication. For example, each user of Draw it or lose it will be given a username and be required to create a unique password meeting certain criteria. With those users’ credentials that user will only be able to access their own information and no other users. We can also take it one step further and require a secondary form of authentication such as multi factor authentication to help guard against unwanted access in the event the first form of authentication is compromised.

From the server side, say with The Gaming Room Employees, they will also be given a username and be required to set up unique passwords meeting specific requirements as well as implementing MFA. We can also use a structure such as a Domain Structure to facilitate what aspects of the system this user is able to access as well as what process and operations the user is able to perform. Granting the user with only the permission required to perform their duties as an employee with help to protect both clients and users against malicious or accidental sabotage. Linux promotes security through the principle of least privilege. Giving users and systems just enough privileges to perform their tasks. Linux also employs process independence. Each process in the system has its own independent virtual address space (The Linux Kernel, n.d.). This prevents a process running one application from affecting another while also protecting certain memory areas against being overwritten by rogue applications (The Linux Kernel, n.d.).

Recommending a cloud based architecture does mean that we are surrendering some aspects of overall control of the system and this does also include security. Security of the server ultimately falls on the host.

References

Bashir, F. (2019b, July 13). *What is Serverless Architecture? What are its Pros and Cons?* freeCodeCamp.org. Retrieved October 16, 2022, from https://www.freecodecamp.org/news/what-is-serverless-architecture-what-are-its-pros-and-cons/

OpenSource.com. (2018, May 31). *What is a Linux Server and Why Does Your Business Need One?* Linux.com. Retrieved October 16, 2022, from <https://www.linux.com/news/what-linux-server-and-why-does-your-business-need-one-0/>

Silberschatz, Abraham, Peter Baer Galvin, & Greg Gagne.(2008).Operating System Concepts, EighthEdition.John Wiley & Sons, Inc.https://learning.oreilly.com/library/view/operating-system-concepts/9780470128725/.

Unix/Linux - File System Basics.(Not Dated.).Tutorials point.Retrieved fromhttps://www.tutorialspoint.com/unix/unix-file-system.htm

*What is Database Replication? 3 Main Types.* (n.d.). Qlik. Retrieved October 16, 2022, from <https://www.qlik.com/us/data-replication/database-replication>

*The Linux Kernel*. (n.d.). Retrieved October 16, 2022, from https://tldp.org/LDP/tlk/tlk.html