

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 | 05/22/25 | Elijah Dames | <Brief description of changes in this revision> |

**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)

Creative Technology Solutions is focused on expanding upon the original Android exclusive version of “Draw It or Lose It” which shares similar functions to the late 1900’s TV show Win, Lose or Draw. The objective of the game is to guess the drawings using only loose drawings of specific things within a one-minute time frame in a four-round structure. The client used should be scalable and support many different platforms to ensure maximum inclusivity. In game names will be unique and one of a kind so that only one instance of each name exists. By leveraging object-oriented programming, the aim is to design a web-based application that implements a singleton pattern or a singleton pattern. This pattern ensures that only one instance is applied across the application and each user has a unique identifier that can be discerned from other users.

## Requirements

*In order to accomplish the desired vision, the following would be required:*

* *A singular game instance can be present*
* *Multiple platform support*
* *Unique names that are only allowed one instance*
* *Multiple teams consisting of multiple players in each team*

## [Design Constraints](#_2et92p0)

1. Single game instance in memory

* The constraint that exists in this Is that it specifically requires a singleton pattern or a closely similar process to managing the game limiting the possibilities immensely if more factors are introduced to the game.
* By implementing so many unique identifiers and lone instances it can be a tedious process trying to manage the memory and state. Using the preferred singleton pattern to implement these specifics is crucial. Implementing a centralized server to keep the game state clean and ensure a stable connection that promotes synchronization of client interactions.

1. Scalability

* Constraint- The game must be able to scale to multiple platforms and the users on the platforms
* This design calls for the inclusion of load balancing and efficient database queries so that a sudden spike in traffic doesn’t crash the application and overwhelm it. Using an appropriate and proven server is integral in this process to minimize the strain a crash can put on the project.

1. Unique In game team and player names

* Constraint- IGN’s and team names must be original and 1 of 1 which calls for a strict and precise mechanism that checks the database for every entry that the user makes when selecting a name.
* A process must be implemented to search the database for the unique name input into the search box by the user. This requires server-side processing and querying so that the performance is not bottlenecked when given the command to search the database.

## [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.

**ProgramDriver-**Contains the method +main(), It is essentially the starting point for the application and directly interacts with SingletonTester class to check the singleton implementation.

**SingletonTester-**Contains method +testSingleton to authenticate the singleton pattern for GameService

**Entity-** An abstract base class that utilizes the attributes -id:long and -name:String. The constructors -Entity() and +Entity(id:long, name, String) along with the methods +getid(): long +getName(): String and +toString(): String in order to implement a unique identifier and name

**GameService-** Game instances are controlled by the attributes-games: List<Game>

-nextGameId: long

-nextPlayerId: long

-nextTeamId: long

-service: GameService

A singleton pattern is enforced using+getInstance(): GameService. Methods used in this class include +addGame(name:String): Game

+getGame(id:long): Game

+getGame(name:String): Game

+getGameCount(): int

+getNextPlayerId(): long

+getNextTeamId(): long

**Game-** Game is **a** class that holds the game instance and inherits from Entity with the attribute -teams: List<Team>. The constructor +Game(id: long, name: String) and the methods +addTeam(name:String): Team

+toString(): String

**Team-**A team class that also inherits from Entity with the attribute-players: List<Player>. The constructor +Team(id: long, name: String)

And the methods +addPlayer(name:String): Player

+toString(): String

**Player-**Represents an active user or “player” by inheriting Entity with constructor +Player(id: long, name: String)

And method +toString(): String

**OOP principles that are demonstrated in this model include:**

**Encapsulation-** Every class uses this on its data like GameService does by hiding the games list and ID only exposing a small amount of methods

**Inheritance-** Entity is inherited by multiple classes in this model including Game, Team and Playerreusing the id and name attributes among others.

**Singleton Pattern and Composition-** The Singleton Pattern is reflected in the GameService class through the+getInstance(): GameService relationship which ensures that only one instance exists which satisfies the Singleton pattern requirements. The GameService to Game, Game and Team and Team to Player relationship shows a composition link by ensuring that they are all held together through parent entities

The software requirements are met through this process such as the Single Game instance in the nextGameId and nextTeamId identifiers.

Unique names are also implemented through the Entity class by providing a name attribute and methods through GameService addGame(name:String) and getGame(name:String) this opens up the ability to ensure only unique names are used when new team or Ign are created.

**"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 has the benefit of making life on the developer significantly easier by having a feature that enables the 3 major systems the ability to run apps simultaneously and side by side. Mac has encouraged continuity in its Operating system structure over the year as well making it easy to integrate. A negative of Mac is that It is known for being very pricey and not very conforming to an upgradeable hardware system | The benefits of Linux is its ability to be widely customizable and open source. The freedom to bend the system to the creative will of the user is unique to other systems, making it popular with developers and mod enthusiasts. Negative aspects are very minimal but include compatibility issues with file formatting. | The advantage that Windows has includes is its overwhelming popularity over the other systems. Another advantage is its simplicity and leniency towards developers who are still growing and looking for experience. Disadvantages include disappointing security protocols that pale in comparison to competing OS’s. | Although mobile devices are not the ideal system to host web-based applications with a low amount of users and a small-scale application it can be a system that pulls encouraging advantages. Low-cost web server applications are an example of an advantage. A disadvantage of mobile devices is that there is limited options for hosting and the main options have a very low level of security. |
| **Client Side** | A pro on the client side of MacOS is its ability to be easily accessible and simple interface options. A con of MacOS would be its lack of versatility on devices that rent apple | Pros of the Linux OS include cheap pricing and the total control of maintenance of the OS via the systems inclination towards freedom and open-source visibility. But with the freedom of open source also comes the cons of being easier to manipulate the security of the system and abuse certain aspects. | Windows availability and overall popularity are significant pros. In comparison to its counterparts Windows puts their best foot forward when it comes to security and support availability. A con is that you would need someone specifically well versed in Windows OS to configure and navigate this OS. | Mobile has slowly inched its way up to the position it is now in making more sophisticated and developed phones that are closer to PCs. Despite the efforts in situations like these, mobile devices lack the specs and features needed to develop a sufficient product. A pro that comes out of using mobile devices is that they are usually more cost efficient than the alternatives. |
| **Development Tools** | MacOS uses Swift as the main programming language. The tool Xcode is used by developers to test and build models in an isolated and secure space. | Linux has perhaps the widest range of tools. A tool that is significantly useful in Linux is Visual Studio Code which is very developer friendly and includes features such as strong debugging, code completion and high customization ability. Docker is another tool used to speed up the development process and ease developers into their concentrated work. | Insomnia is a great tool primarily used to develop REST API’s and secure a project properly. WireShark is also a well-known commodity that is used to scan networks and devices to unveil possible threats and vulnerabilities. | Xcode and Xcode cloud are prime examples of developer tools that are used in mobile environments. These tools are used to develop application for apple products and work seamlessly with eachother to test, code and distribute app and app components across devices. |

## 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 operating platform that I recommend strongly is Linux because of its versatility, cost effectiveness and extensive list of open capabilities. This project needs a server that is highly responsive to scalability and maintaining a volatile level of user traffic. Linux accurately fits these requirements quite well with its ability to support cloud-based deployments, especially being highlighted in this selection.
2. **Operating Systems Architectures**: The Linux system works around the kernel of the computer as its centerpiece that is the brain that controls many main functions of the computer such as memory management and drivers. This would enable the game components to run independently.
3. **Storage Management**: PostgreSQL would make the most sense as it provides open-source management of the database. This option also supports the singleton pattern used by providing indexed tables to handle it through. PostgreSQL supports scalability by table partitioning and bolstering performance when high user loads present themselves.
4. **Memory Management**: Linux tactically uses virtual memory system and its swapping features to keep track of internal memory for the game Draw It or Lose It. Methods that Linux would use to manage the projects memory would include copy -on-write and demand paging to lessen the immediate load and strains that the system might be put under by minimizing the among of things being loaded to only the completely necessary data. To clean up efficiency in the game environment the feature Out-Of-Memory killer prioritizes the most urgent processes to prevent crashes during frantic and high-volume time frames.
5. **Distributed Systems and Networks:** To ensure that the system allows for cross platform communication using microservices architecture gives the game the best chance to carry out that task on Linux. Kubernetes would be used to automate the processes involved to ensure everything internally runs as planned including configuration, deploying and scaling applications. A load balancer would manage the amount of work within the traffic of the application to prevent bottlenecks from occurring and overheating the system. Kubernetes works synonymously with this to redirect workloads to less busy paths during outages to remedy the situation at hand and increase uptime efficiency.
6. **Security**: It is of the utmost importance and especially in a cross-platform environment that the most valuable and effective security procedures are followed as well as the most vaunted Linux security features. All user data should take priority number 1 when securing the projects assets using encryption methods such as TLS/SSL for network purposes and PostgreSQL for storage encryption. Linux enlists tools such as AppArmor in the base version of its system to enforce RBAC and strict authorization security for game services. OAuth 2.0 provides the best authentication practices for login security purposes. These measures aim to cover security on all fronts of the application to ensure a safe and secure environment for all potential consumers.

**REFERENCES**

Apple Inc. (n.d.). *XCode - Apple Developer*. Apple Developer. https://developer.apple.com/xcode/

Çintay, E. (2024, February 25). Linux for developers: tools, tips, and why it’s the preferred OS for development. *Medium*. <https://medium.com/@Chinacolt/linux-for-developers-tools-tips-and-why-its-the-preferred-os-for-development-ce7cb3cb218f>

Hanselman, S. (n.d.). Scott Hanselman's 2021 Ultimate Developer and Power Users Tool List for Windows. *Scott Hanselman’s Blog*. <https://www.hanselman.com/blog/scott-hanselmans-2021-ultimate-developer-and-power-users-tool-list-for-windows>

Prabhakaran, P. (2023, January 12). Getting started with PostgreSQL on Linux. *Red Hat Blog*. https://www.redhat.com/en/blog/postgresql-setup-use-cases

*Understanding microservices*. (n.d.). https://www.redhat.com/en/topics/microservices

*What is Kubernetes?* (n.d.). https://www.redhat.com/en/topics/containers/what-is-kubernetes