[

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 | 03/21/2025 | Tanner Hunt | Wrote executive summary, requirements, design constraints, and domain model. Evaluated the benefit of multiple operating systems. Wrote initial recommendations. |

## [Executive Summary](#_sbfa50wo7nsh)

“Draw It or Lose It” is a Pictionary-esque online game designed for Creative Technology Solutions (CTS). This game is a native application – an app the user installs on their device – for high performance in a competitive, online environment. The game will connect to servers hosted by Creative Technology Solutions.

## Requirements

This game will feature:

* Four rounds of play, about 1 minute each
* Groups of players will build a team
  + Each team will have multiple players
  + Game and team names must be unique
* One or multiple teams may be involved in each game
* An image from a predefined library will be rendered over 30 seconds
* One team should guess what the image is (a phrase, title, or thing)
* If the first team is not able to guess within the time frame, the remaining teams are allowed 15 seconds to provide one guess
* Only one instance of the game may exist at a time
* The game will be hosted on a server by Creative Technology Solutions
* The clients will connect to the game via a native application

## [Design Constraints](#_2et92p0)

* Games must be able to host one or more teams over the internet, with each team hosting multiple players
* The game is a native application
* The game will be server-driven rather than client driven
  + The image to be drawn will be decided by the server. The image data will slowly be sent to the client
  + Wins and losses will be validated by the server
  + Match making will be a service provided by the server

## [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)

Entities are objects that have a both a unique ID and a name for display purposes. Game, Team, and Player objects are all entities. This shows the OOP principle of inheritance, where each object shows an “… is a …” relationship with its parent. These objects also aggregate each other, where a game may have 0 to multiple teams hosted within it, and each team may have 0 to multiple players within it. This fulfills the requirement that a game may have multiple teams competing and each team may have multiple players. Games are managed by the “Game Service” object, which maintains only a single instance of itself using the singleton pattern. A single agent responsible for managing games allows the program to fulfill the requirement of only a single instance of a game existing at one time.

**"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** | Easy to learn (Whatley, 2024)  Resilient to malware and viruses because based on UNIX  (Whatley, 2024)  More expensive than windows  (Whatley, 2024) | Most common server type  (Linux vs Windows, 2023)  Free of Charge  (Linux vs Windows, 2023)  Complex Operation  (Linux vs Windows, 2023)  May not have LTS  (Linux vs Windows, 2023)  Very Secure  (Linux vs Windows, 2023)  Some software does not work with linux  (Linux vs Windows, 2023)  Has managed servers, like RedHat | Large number of settings may make it difficult to learn (Whatley, 2024)  Vulnerable to security errors (Linux vs Windows, 2023)  Less expensive than Mac, but may still have high licensing costs and per-user costs (Whatley, 2024),  (Linux vs Windows, 2023)  Operated through GUI  (Linux vs Windows, 2023)  User error potential  (Linux vs Windows, 2023)  Guaranteed LTS  (Linux vs Windows, 2023)  Has cloud services, like Azure and windows server | Hosting a server on a mobile device is not viable for enterprise applications |
| **Client Side** | Easy to use because of GUI. Developed code will only work on macs, which will increase time and costs of developing for other platforms. Also uses Objective C, which is designed for MacOS and would require additional training for engineers or hiring engineers familiar with the language as it’s otherwise uncommon. $99 publishing fee per year. | Free licensing costs. Compatibility with programs built outside Linux distro has increased, but is still not as compatible as Windows. Training and complexity will depend on which distro is chosen. Least used OS | Easy to use because of GUI.  Has commercial licensing costs. May require developers familiar with .net development, or require additional training for engineers to learn it. | Android studio is a very developer friendly application that is easy to learn and uses Java – a commonly known programming language. This reduces the time and cost associated with training/ hiring developers and reduces project complexity. |
| **Development Tools** | Databases   * PostgreSQ, MariaDB, SQLite   Programming Languages   * Objective C, Swift, Python,   IDE   * Xcode (free) | Databases   * MySQL, MariaDB   (Linux vs Windows, 2023)  Programming languages   * Perl, PHP, Python, Ruby   (Linux vs Windows, 2023)  IDE   * VIM (free), VSCode (), Eclipse, PyCharm | Databases   * MSSQL, Microsoft Access   (Linux vs Windows, 2023)  Programming Languages   * VBScript, ASP.Net, C#, Javascript, Java Licensing cost $10-15 per employee, Python   (Linux vs Windows, 2023)  IDE   * VSCode ($45-500 user/mo), Eclipse (free) | Databases   * SQLite, PostgreSQL, MongoDB, MySQL   Programming Languages   * Java Licensing cost $10-15 per employee, Swift, Objective C, Kotlin   IDE   * Android Studio ($25 one-time fee to publish) * Xcode |

## 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 best operating platform for the game servers is Linux. Linux is an industry standard for server operating systems – Linux operating 63% of servers worldwide (Server Operating System Market, 2025). Linux also has the benefits of being an open-source project, being highly secure, and having a lightweight kernel. Multiple service providers also have Linux servers, like AWS, should the company decide to use a serverless architecture.
2. **Operating Systems Architectures**: Linux is built on the UNIX architecture. While Linux has a monolithic architecture, there are important modules of code: the kernel, the system libraries, and the system utilities (Silberschatz, Galvin, & Gagne. 2015). Between these three modules, the Linux operating system manages memory, creates an API for user programs, and manages devices using daemons. The Kernel is also able to load sections of kernel code on demand. This makes developing custom kernel code easier and minimizes the size of the kernel (Silberschatz, Galvin, & Gagne. 2015).
3. **Storage Management**: An appropriate storage solution for this game must host servers in a secure location. These servers require fast read and write times with direct memory access. A database using multiple disk drives in parallel fulfills both these requirements while being cost effective. Solid state drive (SSD) servers also fulfill the previous storage requirements but come at an increased cost and should not be used in this project.
4. **Memory Management**: The Linux operating system abstracts physical memory away from the developers into virtual memory. Virtual memory is a technique where ranges of physical memory are converted into memory pages. A program can then reference the memory page, which then references back to a physical memory address. Referencing memory pages instead of memory addresses makes programs more secure and reliable. Processes likely need multiple pages of memory to keep track of their data, so Linux tracks a processes pages using indexing (Concepts Overview, n.d.). An index table is a list of pages a program has access to, giving processes fast access to their memory. Pages are also cached (Concepts overview, n.d.), a process that makes data retrieval faster for the program, thus increasing the speed and reliability of the program.

The default way Linux accesses memory from storage on a disk is by Budget Fair Queueing. This is a scheduler that assigns weights to requests and enqueues them appropriately (Budget Fair Queueing, n.d.). The disk scheduler for Linux is configurable.

1. **Distributed Systems and Networks**: Even though the game “Draw It or Lose It” is hosted on a Linux server, the game client can still work on any other platform using the server-client pattern. A user interface is built for each operating platform (Windows, Mac, iOS, Android), which then sends encrypted requests to the server for data. The server receives the requests, finds the appropriate data, then sends a response. There are several benefits to this design. Firstly, this increases the security of the system. Allowing users to connect directly to each other could expose them to malware. It also mitigates cheating by hiding data from users behind a server. This design also makes code more modular, extensible, and easy to maintain. A detriment to this architecture is that an outage in the server or an internet provider loses players access to the game. Players far away from a server may also experience increased latency in their gameplay. Outages in individual storage devices on the server may also lead to lost player data, which could be a frustrating experience for users.
2. **Security**: There are many layers of security to implement when creating a secure application. Users connecting over the internet should use the HTTPS protocol, which encrypts their data for additional protection. The site the servers are on should also be secured from strangers by a locked door. Employees should regularly be trained in the dangers of phishing attacks. All API calls to sensitive data and functions should implement a security check. The databases should log requests in the case of a security breach to create an audit trail. Users should be given the minimum amount of permissions necessary. User inputs should be validated to protect against SQL injections and buffer overflow attacks. A firewall can also be used for added protection against certain attacks. Finally, users should use two-factor authentication and be frequently reminded about security practices – like not sharing their password, even with official-looking accounts.

Linux is a secure operating system, especially compared to that of Windows or Mac. Linux tracks user privileges, restricting them from sensitive data and functions. The open-source nature of Linux also means that the source code is reviewed and tested often. Package managers also vet downloads and keep installations up to date (Kaspersky, 2025).

References:

Budget Fair Queueing. (n.d.) *Linux Kernel Docs.* <https://docs.kernel.org/block/bfq-iosched.html>

Comparing Mac and Windows Servers for Web Hosting. *MamboServer.* Retrieved from <https://www.mamboserver.com/blog/comparing-mac-and-windows-servers-for-web-hosting/>

Concepts Overview. (n.d.). *Linux Kernel Docs.* Retrieved from <https://docs.kernel.org/admin-guide/mm/concepts.html>

Kaspersky, A.O. (2025). What is Linux and is it really Secure? Retrieved from <https://usa.kaspersky.com/resource-center/definitions/linux?srsltid=AfmBOopusfe6gBr5413AVoPGmPQ92FpRQzBqX7aQ9sQtWZNqDyW--bQR>

Linux Vs. Windows Servers (2023). Retrieved from <https://www.ionos.com/digitalguide/server/know-how/linux-vs-windows-the-big-server-check/>

Server Operating System Market Volume, Share & Industry Analysis, By Operating System (Windows, Linux, UNIX, and Others), By Virtualization Status (Virtual Machine, Physical, and Virtualized), By Subscription Model (Non-paid Subscription and Paid Subscription), By Enterprise Type (Large Enterprises and Small & Medium Enterprises), and Regional Forecast, 2025-2032. (2025). Retrieved from <https://www.fortunebusinessinsights.com/server-operating-system-market-106601>

Silberschatz, A., Galvin, P., Gagne, G. (2015). Operating System Concepts, 8th edition. *Wiley.*