

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 | 12/06/2022 | Lindsey Stoughton | Completed Executive Summary, Design Constraints, and Domain Model sections. |
| 1.0 | 12/14/2022 | Lindsey Stoughton | Completed Server Side, Client Side, and Development Tools sections. |
| 1.0 | 12/16/2022 | Lindsey Stoughton | Completed Recommendations sections. |

**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)

The Gaming Room company has requested we develop a web-based version of the Android only game Draw It or Lose It. The software design must include the ability to have one or more teams, each team having multiple players assigned to it, only one instance of the game in memory at any given time, and that each game and team name are unique from one another. By creating game, team, and player classes that hold unique IDs and names in each instance, we can verify upon creation if an entered name is already in use. Then, through association, each team will contain a list of players, and each game a list of teams that will interact with the game service. By restricting the game service class to a singleton design pattern, the game service will be limited to only a singular instance in memory at a time.

## [Design Constraints](#_2et92p0)

To fully function in a web-based environment, the game must be rewritten into a web-based language, such as JavaScript or Python. This restraint may require onboarding of a new programming team or outsourcing if this is not a language or environment the in-house team has experience with, potentially leading to scope adjustments in schedule and budget. Hosting a web-based game also introduces new server constraints. The client will need to decide if this is something they also wish to outsource or if an adjustment in the budget is needed to cover the influx of server activity. With server and network dependencies, additional performance optimizations may also be necessary to reduce latency and user hardware requirements, increasing the potential customer pool.

## [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 UML class diagram provided below demonstrates how we plan to use the advantages of object-oriented programming to fulfill the software requirements of this project most efficiently. Firstly, we use inheritance of the entity class to pass along shared traits, such as the ID and name, and shared functions, such as the constructor for the ID and name and accessors to private variables, to the player, team, and game classes. Then, using zero-to-many associations, the game service and game classes, game and team classes, and team and player classes can communicate their connected information, such as a list of player objects within a team object, a list of team objects within a game object, a list of game objects within a game service object, and the functions needed to build such lists. The game service class acts as a host for generating unique static player, team, and game identifiers, and holds accessors for receiving this information. The game service class follows a singleton design pattern through use of a private constructor and the static method “getInstance()” that has a return type object of this same class. By pointing any new instances to the first instance, the class is restricted to the instantiation of only one object at a 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** | Mac provides a console, command line, prompt, and terminal with an easy-to-use graphical user interface for server configuration and accessibility. Servers tend to be stable and well prepared for hosting. | Being largely open source, Linux is thoroughly and regularly reviewed for stability, efficiency, and security. However, this robust system can cause some hardware compatibility issues out of the box. Linux’s command terminal allows for easy server configuration and accessibility. | Windows has an easy-to-understand graphical user interface out of the box and Microsoft-backed customer support. The command terminal is user friendly but more bare bones than Mac. There are many more options for available and compatible software and hardware for upgrading servers and the server-side experience. | Mobile device specifications vary from provider, make, and model, requiring a more versatile and adaptable approach to software development. Mobile apps often benefit from offline access options, additional monetization, and the ability to target a wider audience. The ever-changing mobile market requires far more maintenance, and the larger audience will require a larger server capacity. However, offline options may alleviate some of the additional stress on the servers. |
| **Client Side** | Mac hardware is often more expensive than the rest but offers consistently high quality. However, Mac hardware is limited to the hardware produced by Apple, so updates will rely heavily on the frequency they become available. | Linux is often less expensive than the rest but more difficult to find distributors or support for. The user interface and command terminal require a steep learning curve for those not already familiar with the system, contributing to time and cost of development. Linux is lightweight, making it faster and more powerful. | Windows hardware varies drastically in quality and cost but is often the most accessible to a wide range of individuals and more configurable than Mac. There’s a high likelihood the client is familiar with the system and support is easy to find. Windows is less secure than Linux, so antivirus will be needed. | Mobile hardware varies drastically in cost and performance capabilities. Extra attention may need to be paid to throttling or reducing processor load on the user. Can take more time and money to implement due to the high skill and experience needed to support frequently updated and diverse mobile devices. |
| **Development Tools** | Programming languages used to build Mac software include HTML, CSS, and JavaScript. Mac also supports Apple Developer Tools, which are a suite of software tools to aid developers in application performance, behavior, and user experience. IDEs for Mac include Visual Studio Code, Xcode, CLion, Ninja-IDE, and AppCode. | Programming languages used to build Linux software include HTML, CSS, and JavaScript. Being open source, there are many open-source development tools created by the community, such as Gedit, Vim, Netbeans, Bluefish, and Geany. IDEs for Linux include Atom, Visual Studio Code, Sublime Text, and Eclipse. | Windows supports a huge range of major languages, tools, and programs compared to the other platforms. These include JavaScript, Python, Android, Ruby, C and C++, C#, PHP, PowerShell, and Rust. The Windows Software Development Kit and Microsoft Visual Studio also cover most all development tools needed. | Programming languages used to build mobile device software include Java, Python, C++, Kotlin, and Rust. Development tools include Xamarin, Sencha, Appcelerator, Ionic, and Framework 7. IDEs for mobile devices include Android Studio, Xcode, and Visual Studio. |

## 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**: Expanding to a Windows operating platform will assure the most reliable and user-friendly server- and client-side experiences. The extensive support through Microsoft customer service and integrated development tools makes it easy to learn and adapt to new projects. Windows is also one of the most accessible options with many different quality and price points to choose from and a large variety of compatible hardware and software.
2. **Operating Systems Architectures**: The core of the Windows operating system consists of the Windows kernel and low-level device drivers. The kernel is responsible for taking operating system requests from system services and translating those requests into instructions for computer hardware. Windows operating system services are part of the operating system rather than components that you install after the operating system deploys. System services include various executive services that provide distinct functions within the operating system. At the upper level of the operating system, apps operate by interacting with the computer user, and at a lower level by integrating with the operating system services. You install apps after you install the operating system, and you must start apps manually to use them.
3. **Storage Management**: An appropriate storage management for the Windows operating platform is the cloud service Microsoft Azure. The platform is cost-efficient and is a fully unified delivery pipeline, so testing, integration, and delivery are uninterrupted and compatible with other Azure tools at every stage. Azure storage services offers secure, high-performance, reliable, and scalable cloud storage with unified data management within Microsoft cloud storage solutions, so it can grow with your need.
4. **Memory Management**: The Windows operating platform uses swapping and virtual memory techniques for memory management. Swappingis a memory management technique used in multi-programming to increase the number of processes sharing the CPU. It is a technique of removing a process from the main memory and storing it into secondary memory, and then bringing it back into the main memory for continued execution. This secondary memory, or swap-space, is disk space that is used as an extension of main memory by the virtual memory. Virtual memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of the main memory.
5. **Distributed Systems and Networks**: To communicate Draw It or Lose between various platforms, we must have a distribution system that connects the devices through a network. Using a network operating system, like Microsoft Windows Server, is what allows resource sharing between two or more computers operating different operating systems. Network operating systems allow for highly reliable and secure server access. Distributed operating systems can share their computing resources and I/O files while providing users with virtual machine abstraction. By distributing systems, data processing jobs are distributed between the processors, increasing data availability and the speed of data exchange. Since the system operates independently of one another, if a single system experiences a crash, outage, or data corruption, the system remains operational.

**Security**: The Windows operating platform has built-in advanced encryption and data protection, robust network and system security, and intelligent safeguards against ever-evolving threats. Microsoft Windows Server comes with even more security features, like Host Guardian Service and shielded VMs to withstand internal attacks by encrypting the virtual hard disks. Windows Server also uses SDN (software-defined networking) methodology, that enables central configuration and management of physical and virtual network devices. This adds a security layer allowing application isolation across the network.