

Draw it or Lose it

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 09/21/25 | Cielo Neal | Executive summary, requirements, and constraints written. Domain model described. |
| 2.0 | 10/5/25 | Cielo Neal | Evaluation table filled out |
| 3.0 | 10/19/25 | Cielo Neal | System recommendations provided |

**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 Game Room wants to create a game called Draw it or Lose it, in which teams will compete to guess a theme based on a slowly rendered version of a stock image they are shown. The games will have 4 rounds of play lasting one minute each. The game can be played with a single team but not a single player. The game and team names must be unique and only one instance of the game can be played on the device at a time. The game will be web-hosted and allow online play.

## Requirements

* A game can have one or more teams
* Each team will have multiple players
* Game and team names must be unique and the system must check whether a name is taken
* Only one instance of the game can exist in memory
* Instances of the game, team, and players all have unique IDs
* The game is web-hosted
* The game must have online multiplayer
* The game must draw on a library of stock images, ideally self-hosted
* A timer must be implemented to facilitate game rounds

## [Design Constraints](#_2et92p0)

* **Host server**. Because This is a web-based game, a server-client relationship must be built to facilitate the game
* **Unique IDs**. Each instance of the game, a team, and a player must exist only once, as such unique IDs will be generated to ensure unique instances.
* **Team names**. Because team names can be input by the player, the system must check that the name does is not already in use.
* **Design patterns**. The main design patterns that will be in use for this project are the singleton and iterator patterns, which will be used to implement game, team, and player instances.
* **Cross platform development**. This game is intended to be cross-platform, and as such, development team members should be familiar with cross platform development already.

## [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 Game, Team, and Player subclasses all derive from a superclass, Entity. The Entity class will allow the subclasses to inherit the ID and name fields, the entity constructor which will be overloaded, and the getId, getName, and toString methods. The GameService class will allow the actual creation of game, team and player objects, which can have 0 or many separate instances. Separately the ProgramDriver contains the main method allowing the program to function. ProgramDriver currently calls upon the SingletonTester class to ensure that the singleton pattern is implemented correctly.

**"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** | **Advantages**   * Stable and reliable * Intuitive UI * Malware resistant * Minimal setup from the user * Supported by Apple tech support   **Disadvantages**   * Can only run on Apple computers * High upfront hardware and software costs * Limited software options compared to Windows and Linux * Limited user control | **Advantages**   * Free to use * Most popular server OS, with high community support * Highly secure * Stable and reliable * Highly scalable * High software compatibility   **Disadvantages**   * Controlled with commands * High user setup * Unintuitive for beginners * Distributions vary greatly | **Advantages**   * Familiar to most people * Intuitive GUI * Compatible with Microsoft programs * Supported by Microsoft tech support * Can run virtual machines   **Disadvantages**   * High licensing cost * Not as reliable * Updates frequently * High hardware load | **Advantages**   * Highly available tech * Familiar to most people * Web server apps are available to aid setup * Low to no cost * Allows the server to be run from almost anywhere   **Disadvantages**   * More easily interrupted * Limited comp resources * Easily interrupted * Difficult setup * Difficult to secure |
| **Client Side** | **Advantages**   * Easy and intuitive to use * Minimal user setup * Supported by Apple tech support * Difficult to break via user error * Access to Apple-specific software   **Disadvantages**   * Can only be run on Apple hardware * High hardware cost * Updates are often paid or require new hardware * Limited user control | **Advantages**   * Free and open source * Distribution packages allow for high customization * Can be booted onto any hardware * Low upkeep   **Disadvantages**   * Not widely used for PCs * No official tech support * Updates are handled via command line * Intensive setup * Not prioritized by software developers | **Advantages**   * Familiar with most users, as the most common OS * Compatible with most programs * Simple GUI * Supported by Microsoft tech support * Allows user configuration   **Disadvantages**   * High hardware usage * Unwanted features (such as Copilot) take up CPU * Well understood by attackers | **Advantages**   * Highly accessible * Development tools are highly available * Generally affordable * Familiar with most users   **Disadvantages**   * Cannot use as many resources as a PC * Several different mobile OS’s, which all behave differently * Slow run time compared to computers |
| **Development Tools** | * **Languages**: Objective-C and SWIFT * **IDEs**: Xcode, Xcode cloud, Eclipse * Can run virtual machines for Linux and Windows testing | * **Languages**: C, C++, Java, Python * **IDEs**: Eclipse, PyCharm, VS Code * Can be dual booted onto other hardware for cross testing | * **Languages**: C# and .NET * **IDEs**: Visual Studio, Eclipse * Visual Studio offers many plugins and is a highly popular IDE | **Android**:   * **Languages**: Java * **IDEs**: Eclipse, VS code   **iOS**:   * **Languages**: Objective-C and SWIFT * **IDEs**: 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**: I recommend Linux as the server OS for The Gaming Room. Despite not having official IT support like MacOS or Windows, Linux is widely used for servers and has an active community that can readily help with any issues faced. I also believe the advantages of it being free to use with many different distributions with the capability of being just as “plug and play” as Mac make it the best option.
2. **Operating Systems Architectures**: Linux systems are built around the Linux kernel, a lightweight UNIX-based kernel that allows for Linux’s modularity. Linux has over 600 distributions, which come with specific packages to support different user needs. For The Gaming Room’s server computers, Ubuntu, Debian, and Red Hat are all strong distro contenders.
3. **Storage Management**: Linux makes use of LVM or Logical Volume Manager, which allows for dynamic resizing. Disc space is treated as a pool rather than fixed partitions, which allows for the least amount of storage fragmentation possible. Disc usage can be monitored and optimized with simple command lines, which makes storage management a simple process, even if not automatic by default.
4. **Memory Management**: Linux makes use of virtual memory, allowing for usable memory much greater than its physical memory. One way it allows this is through dynamic loading. Only what is absolutely necessary is loaded into memory at a time, with calls for things outside of current memory having to be pulled from disk. This means that calls outside of memory can take a long time, but memory usage is more efficient overall. The Linux system also uses page swapping in cases where all pages are in use, where a currently loaded page is saved to a swap file and de-loaded from memory based on a least recently used aging system. This dynamic loading should work well for The Gaming Room’s purposes, though it may create server lag during times of intensive, unexpected usage.
5. **Distributed Systems and Networks**: In the case of Draw it or Lose it, REST APIs will be used to facilitate a client-server connection between many different users. HTTPS protocols will be used to give and receive requests from the client systems and exchange data will likely be structured in JSON or XML. Networking will be handled through the internet, as this is a web-service. Multiple client programs will be designed to communicate with different operating platforms, which will all communicate to the same server program through HTTPS protocols.
6. **Security**: The principal of least privilege will be absolutely necessary to maintain the security of the system and the users. Users must log in to access their own accounts and users must have no access to the database storing user and team data. Attacks such as cross-site scripting and injections will be avoided using input validation and limitations, such as whitelisting types of input. Additional system security is provided by Linux, which uses a strict privilege model in which only one user can access all privileges. This superuser role would be given to a system administrator in charge of the server. Additionally, the Linux kernel itself is built with security defenses such as firewalls, Kernel lockdown, Secure Boot, and many other options which can be configured. This combination server-side and client-side security approach will protect the system from attack or misuse as much as technically possible.