

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 7**

**References\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.2 | 06/18/2023 | Tabitha Tallent | Made various recommendations. |
| 1.1 | 06/04/2023 | Tabitha Tallent | Chart for operating systems comparison completed. |
| 1.0 | 05/21/2023 | Tabitha Tallent | Initial summary of project and it’s design requirements. |

**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 company is creating a web-based application for their game called Draw It or Lost It, the idea is to guess the image being rendered on screen. To complete the game, it needs to have the ability to have either one or multiple teams with multiple players on each team. The teams must have unique names, which should be checked when creation is attempted. Additionally multiple games can exist, but each game must also have a unique name which also needs to be checked on creation, and only one instance of any unique game can run at a time. Each game consists of four rounds of 60 seconds. Each image is completely rendered after 30 seconds, and if the first team has not successfully guessed the image, the remaining teams have 15 seconds to offer a guess.

## Requirements

Provided from prompt:

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

Additional note:

* Web-based for multiple platforms

## [Design Constraints](#_2et92p0)

* Needs to function cross platform (Windows, Linux, MacOS)
* User interface probably needs to be similar, if not identical to the already existing android app
* Potentially budget and/or timeframe for completion

## [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)

There is a lot of inheritance present in the UML, with Entity being the base class, which can also be referred to as a superclass. Entity as two private attributes, one private method, and four public methods. The GameService class provides the basis for the game instances, including five private attributes that store the current list of games, and the IDs associated with each game, player, and team. It also possesses several public methods that keep track of and retrieve the current game instance, and retrieve the nest games, players, and teams. This class makes use of the singleton design pattern. The classes Game, Team, and Player are all children classes to Entity, from which they inherit attributes, as well as adding their own. Note that a team can have players, and a game can have teams. Overall, the SingletonTester in the main driver double checks to make sure no more than one instance of any one game is active 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** | * MacOs can be used as a server * Likely have to use Apple proprietary software which can be pricey and limiting * Apple server is being deprecated | * Linux is known for being secure * Open-source, so this is a reasonable cost option * This may be unfamiliar to the staff/client | * Special system for file management could be useful * Costs could be reasonable as Windows is often already in use * Can be difficult to contact tech support | * Cloud servers useful here * Could be costly depending on the amount of server space needed |
| **Client Side** | * Easy to integrate with other Apple devices * Commands are similar to Linux commands which can boost familiarity | * Opensource, but often have to self-install Linux * Most common operating system for websites | * Good for virtualization to test across multiple platforms * Lots of client-side compatible web technologies | * Needs to be compatible with Android, ios, and potentially Windows * Easy-to-use interfaces, users are generally familiar |
| **Development Tools** | * May need developers familiar with Swift * Java and other desired languages may need to be installed * VS is an IDE option, as well as ones that may be less familiar | * Need developers familiar with Linux commands * Everything has to be installed, easiest through the terminal * Larger selection of common IDEs | * Potentially less training is required here * Lots of available frameworks/libraries for easy development * Almost all software is natively compatible with Windows and installations tend to be easy | * Developers need to be familiar with available browsers to keep compatibility across operating systems * programs should scale appropriately to screen size, may even consider building a “mobile version” * Easy for quickly checking things through GitHub, can be difficult to program on |

## 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**: Microsoft Windows for familiarity, virtualization convenience, and ease of use.
2. **Operating Systems Architectures**: Windows is the most frequently used operating system which can reduce training time and costs. Windows file server options already come with things like security, backups, and quotas built in. Additionally, it has easy options for virtualization making testing simpler, and the interface is user-friendly and generally familiar to navigate. Their flexible application platform would be particularly useful for the number of images included in Draw It or Lose It (Microsoft, n.d.).
3. **Storage Management**: A linked allocation storage management system would be convenient for Draw It or Lose It. Although this option does take up a bit more space, it allows for direct access to the stored assets of the program in an organized manner of lists of storage blocks on the drive which reduces load times versus an option that must travel a sequential path (Silberschatz et al., 2008, chapter 11).
4. **Memory Management**: Microsoft Azure is Window’s cloud services option and it handles memory management fairly well on its own with services such as Azure App Service. Additionally, it has services that allow for easy analysis of memory usage as well as a huge selection of additionally related memory management services options to select as necessary (Microsoft Azure, n.d.).
5. **Distributed Systems and Networks**: Distributed systems and networks allow for data to be shared through a collection of individual processors that do not have the same memory or clock. This requires that the system be able to communicate via some sort of communication line, have a way to determine the order and outcome of processes, and possess a certain robustness. Ideally, the system needs to be able to perform a check between links for functionality, have the ability to reroute information as necessary, and switch back to normal operation when a failed link is repaired. Draw It or Lose It has a lot of information and should be built with a fault-tolerant network to minimize outages (Silberschatz et al., 2008, chapter 16).
6. **Security**: Windows and Azure, its cloud server architecture, come with a variety of security options, and built-in features to assist with protecting user data and the program itself. Additionally, Draw It or Lose It includes role-based access for data protection, and should enact some form of encryption when handling user data. Two-factor authentication is ideal and maintaining basic security measures such as anti-virus software, system logging, and monitoring is essential (Silberschatz et al., 2008, chapter 15).

Resources

Ardalis. (2023, April 26). *Common client-side web technologies*. Microsoft Learn. <https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-client-side-web-technologies>

*Directory of Azure Cloud Services | Microsoft Azure*. (n.d.). Microsoft Azure. <https://azure.microsoft.com/en-us/products/>

Joel. (2022, September 19). *Windows vs macOS vs Linux for Web Design & Development - PixemWeb*. PixemWeb. <https://www.pixemweb.com/blog/windows-vs-macos-vs-linux-for-web-design-development/>

*Windows Server 2022 | Microsoft*. (n.d.). <https://www.microsoft.com/en-us/windows-server#tabxbc8497093e944431b9063f495d465c22>

Ritter, P. (2023, June 3). *Best Server OS in 2023 - Which Operating System Is Right for Your Server?* Cloudzy. <https://cloudzy.com/blog/best-server-os/#:~:text=Microsoft%20Windows%20Servers%F0%9F%A5%88,-Microsoft%20Windows%20servers&text=The%20best%20thing%20about%20Windows,its%20unique%20file%20management%20system>.

Silberschatz, A., Gagne, G., & Galvin, P. (2008). Operating System Concepts 8th Edition. In *Wiley Publishing eBooks*. <https://learning.oreilly.com/library/view/operating-system-concepts/9780470128725/>

TrustRadius. (n.d.). *Windows Server Reviews & Ratings 2023*. <https://www.trustradius.com/products/windows-server/reviews?qs=pros-and-cons#reviews>

*Use Mac mini as a server*. (n.d.). Apple Support. <https://support.apple.com/guide/mac-mini/use-mac-mini-as-a-server-apd05a94454f/mac#:~:text=The%20most%20popular%20server%20features,more%20of%20the%20options%20below>.

*What is Linux Server? - Sisense*. (2019, June 26). Sisense. <https://www.sisense.com/glossary/linux-server/>