

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 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/16/2023 | Tyler Humphries | Design document for initial release |
| 1.1 | 04/01/2023 | Tyler Humphries | OS Evaluation and recommendation added |

**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)

Draw It or Lose It presents players with an alternative game similarly based on the television game *Win, Lose, or Draw.* It provides an alternative to drawing a picture by providing stock images so the game can be played on a mobile device.

## Requirements

The client wishes for the game to be compatible with multiple players in one or more teams. The team names should be unique and allow for players to choose them. Only one instance of the game can exist in memory at any time.

## [Design Constraints](#_2et92p0)

There is a financial constraint for development on this project. Because the project owners don’t have much development experience, they will be unable to maintain it themselves. If something goes wrong in the game, or a future update is needed, the company will need to hire additional outside help, which could cost a lot of money, as opposed to maintaining it themselves.

## [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 Entity class, which creates the instance of the game, inherits the Game, Team, and Player classes. The GameService class controls the overall gameplay and controls the instances of the game to delete and add instances when necessary.

**"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 offers server-side software called macOS Server, which allows users to host websites, share files, manage e-mails, manage calendars, and more | Linux offers several server-side deployment methods such as Ubuntu Server, CentOS, Debian, and Fedora. These programs come pre-installed with tools such as Apache, PHP, and MySQL | Windows offers a web server software package called Internet Information Services (IIS). IIS allows users to host a website usings tools such as ASP.NET, PHP, and Microsoft SQL Server | Mobile devices do not offer capabilities to host websites |
| **Client Side** | In order to ensure compatibility with the Apple OS, the developers will need to account for Apple’s Human Interface Guidelines, as Apple products use a unique interface for compatible software. Additionally, App Store guidelines must be adhered to. Something else to consider with this is the fact that Apple’s preferred programming languages for its compatible software is Swift. | To start off with Linux, developers will need to consider the target distribution for their application, such as Ubuntu, Debian, CentOS, Fedora, and others. Each distribution has differences in libraries and packages available. The preferred programming languages for Linux applications are C, C++, Python, and Java. Linux also contains several dependencies for specific libraries and packages, which will need to be installed on the system in order to play the game. | Developers will need to use a Windows-based development environment like Visual Studio. Just like Linux, Windows uses a specific User Interface style for its compatible programs. C#, C++, and Visual Basic are the preferred programming languages for Windows based programs. Developers will also need to follow strict guidelines for Windows App Store, as Microsoft has some guidelines for software listed in the store. | Developers will need to consider the target mobile operating system, such as Apple OS or Android. Each operating system has its own requirements, programming languages, design principles, and development tools. As there are hundreds of devices, developers will also need to consider compatibility and backwards compatibility of devices. Just like Apple, Linux, and Windows, mobile devices will have app store guidelines for each operating system’s respective app store. |
| **Development Tools** | Apple’s preferred programming language for software is Swift. More than one development team is most likely unnecessary to build the program for Apple compatibility. | C, C++, Python, and Java are the preferred programming languages for Linux devices. Due to Linux’s multiple target distributions of Ubuntu, Debian, CentOS, Fedora, and others, it is recommended to have more than one development team so the project can be ported to every distribution of required. | Windows’s preferred programming languages are C#, C++, and Visual Basic. It is most likely unnecessary to have more than one development team to port the program over to Windows. | Due to the many different devices available and compatibility for those devices, more than one development team may be required for the project to work on a mobile device. |

## 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**: Linux is the most capable operating platform to go with in order to expand *Draw It or Lose It* to other computing systems.
2. **Operating Systems Architectures**: Linux offers server-side deployment tools such as Ubuntu Server, CentOS, Debian, and Fedora, that come pre-installed with additional tools meant for server-side hosting such as Apache, MySQL, and PHP.
3. **Storage Management**: For the purpose of the *Draw It or Lose It* game, I would recommend using Linux’s Storage Area Network (SAN) for storage management. SAN is a high-speed network that connects storage devices to servers.
4. **Memory Management**: Linux uses paging, swapping, and memory-mapping to manage memory. This means that memory an app such as the *Win It or Lose It* game remains in RAM storage and doesn’t cause memory strains on the device hosting the game.
5. **Distributed Systems and Networks**: The game will most likely have to utilize a network set up by the game’s host. This can either be a LAN network for players connected to the same network, or an over-the-Internet network for players outside of the host’s home network.
6. **Security**: To protect user-information in an ongoing game, the network should be protected by a specific ID, or “room name,” and a password so only the game’s players can connect to the in-session game. Hosts should only provide the room name and password to trusted players.