

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 | 09/24/24 | Curtis Hetrick | Updated Executive Summary and Design Constraints. |
| 1.1 | 10/13/24 | Curtis Hetrick | Updated Evaluation Table |
| 1.2 | 10/27/24 | Curtis Hetrick | Updated Recommendations |

**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 needs their existing Android game, Draw It or Lose It, translated into a web-based game that functions from multiple platforms. All team names, players, and games will be stored to ensure that team names and players are always unique, and only one instance of each game can exist at a time.

## Requirements

As a web-based game, Draw It or Lose It must be compatible with all browser types. Each instance of the game must be unique, with its own unique team and player names. All teams must be made to have multiple players on them.

## [Design Constraints](#_2et92p0)

1. Program must operate on all browsers (Technical Constraint)

* For the game to be accessible across multiple platforms, it must be compatible with all mainstream browsers, such as Firefox, Internet Explorer, Google Chrome, Opera GX, etc.

1. Game is restricted to existing library of stock drawings (Technical Constraint)

* Using outside sources for images would likely cause licensing issues, so using the large library of stock images from existing the Android version is the best course of action.

1. Internet connection will be required to participate in game (Technical Constraint)

* Users must be connected to the internet while playing to send guesses and receive results.

1. CTS will cover early stages of game development as requested by client (Business Constraint)

* The Gaming Room has asked for Creative Technology Solutions to streamline the development process by developing the game application based on current requirements.

## [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 first thing to notice about the UML diagram is that a superclass named “Entity” is used to create the *Game, Team,* and *Player* classes, which is an example of inheritance. According to the connections between *GameService, Game, Team,* and *Player*, each *GameService* can have none or many *Game* instances. Every *Game* can have multiple *Teams* and each *Team* can have multiple *Players*. An example of Polymorphism can also be seen in how the *Entity* class gives fields and methods to its subclasses, but each subclass adds its own unique fields and methods on top of the inherited properties. Using that *Entity* superclass makes it so that the subclasses don’t all need to implement fields and methods that are common among them, which makes the subclasses more efficient. Encapsulation can be seen with the “-” symbols beside some methods and fields, which indicates that those are private and cannot be accessed directly from outside that class, adding a layer of security. If we look at the *ProgramDriver* class, it contains a main() method, which indicates that it is where the *GameService* is implemented. From main(), a *GameService* would be created, then that *GameService* would initialize any *Games,* along with the *Teams* and *Players* associated with them. Finally, the *SingletonTester* class would be used to test for duplicate *GameService* instances which should not be possible as only one *GameService* should be able to exist 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 Requirement** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Along with offering built in web hosting, Mac is known to have very good security. Unfortunately hosting on Mac OS comes with less compatibility and isn’t as friendly when it comes to games. | Linux offers potentially the cheapest starting cost for hosting a web app due to needing no licensing. Also, thanks to it being open source, security is easier to set up. The downside is that Linux has less compatibility and isn’t as immediately user friendly. | Windows has the advantage that it is compatible with the majority of web browsers and is the most widely used OS around the world. However, it can be more costly and potentially less secure than some other OS when it comes to hosting web-based applications. | While it is possible to host a web based app from a mobile device, their hardware is not built to withstand that much strain. Unless the web app is private and used by very few users, it would not be a reliable hosting platform. |
| **Client Side** | Familiarity with Mac and Safari are a must for any web app made for Mac. Having experience with Xcode will also be important. Since many of the assets already exist on the Android version, the cost and time should remain lower. | Developers should be familiar with the command line navigation of Linux. Since Linux offers support to many browsers, the game would need to be compatible with most common browsers. | Almost all browsers would need to be supported for Windows clients. Since Windows is so widely used, the familiarity will make development much easier than some other platforms. | Mobile app development is thankfully a thriving industry, which means it is cost effective. Tools for developing are also widely available, making it easy to develop the apps quickly. Existing Android app layout can be reused too. |
| **Development Tools** | For development on Mac, it is common to use languages such as Swift, Java, and Objective-C. Depending on the language, developers may use IDEs like Visual Studio, Eclipse, or Xcode. | Developing on Linux commonly uses Java, Python, and C++ to name a few. For these languages, you may use PyCharm, Visual Studio, Eclipse, or even Atom. | Java, Python, C++, and also JavaScript are common languages for making apps on Windows. These can once again be developed using PyCharm, Eclipse, Visual Studio, or WebStorm. | For mobile devices, common languages range from Java to Swift, but also Kotlin for Android and C#. IDEs to be used may be Eclipse, Xcode, Visual Studio, and IntelliJ. |

## 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 would suggest using Windows as the base for constructing Draw It or Lose It. Thanks to it being the most overwhelmingly used OS in the world, there is a plentiful supply of resources to aid in development and distribution. Windows also offers compatibility with the majority of programs, especially anything game related. All around the world, companies and organizations have used Windows on their systems for hosting games and servers reliably while remaining cost effective.
2. **Operating Systems Architectures**: With Windows, The Gaming Room would be using Windows Architecture. This is made up of multiple layers known as User Mode and Kernel Mode. These layers communicate through an input/output manager after it has been decided what access level is allowed. Anything which only needs limited access to the computer’s resources belongs to the User Mode layer, while processes which need full access to the computer must go through the Kernel layer (Wikipedia contributors, 2024).
3. **Storage Management**: Windows features a very easy to use storage system which will make managing the necessary files a smooth process. Using features like Storage Sense to filter out unneeded processes and potentially cloud storage like OneDrive, Draw It or Lose It can rely on Windows to maintain proper organization and backups for its files. OneDrive could provide a good cloud storage solution seeing as it would give both mobile and desktop users quick access to game files.
4. **Memory Management**: Using 64-bit Windows, which I would recommend over 32-bit, would have Draw It or Lose It utilizing the virtual address space system that comes standard. This means that each process is assigned a virtual address space which represents the range of memory a process can use. Within this system, processes are not allowed to access the memory of existing processes so that they cannot corrupt each other. I recommend using a 64-bit Windows OS because 32-bit only allocates 4 gigabytes of memory, while 64-bit can allocate up to 16 terabytes between the kernel and user programs (adichavan095, 2022).
5. **Distributed Systems and Networks**: I’d highly recommend The Gaming Room use a cloud server provider such as Amazon’s AWS or Microsoft Azure to help make the game accessible to all users around the world. This will also make outages and scaling easier to handle as the cloud server provider has systems already built to deal with both. Finally, Draw It or Lose It should implement a RESTful API to handle requests and responses between users and the game’s back end.
6. **Security**: For security, I’d suggest using role-based access control. This way, users of the game can be given a certain number of permissions that are just enough to interact with the game, but not enough to risk compromising the game’s security. Similar to how programs like Discord use roles, Draw It or Lose It can have a role system programmed into the game as a form of the principle of least privilege. Standard users who create an account can be given a “default” player role upon account creation. Meanwhile, we could create a pre-built list of accounts which have an “administrator” role or “tester” role. That way anytime someone from the dev team needs to make adjustments or a test-patch needs tested, those pre-built accounts can be logged into. Since this is a light game, I’d also recommend players be able to play without accounts as a streamlined way of allowing people to play. These could just be considered temporary “guest” accounts with the same bare minimum permissions of a default player. Overall, I believe role-based access is the most convenient and adaptable security system for the game.

**References**

adichavan095. (2022, February 1). *Windows Memory Management*. GeeksforGeeks. <https://www.geeksforgeeks.org/windows-memory-managment/>

Wikipedia contributors. (2024, September 23). *Architecture of Windows NT*. Wikipedia. <https://en.wikipedia.org/wiki/Architecture_of_Windows_NT#:~:text=The%20architecture%20of%20Windows%20NT,system%20resources%20of%20the%20computer>.