

Draw it or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/22/23 | Zahlen J. | Added Initial Documentation. |
| 1.1 | 02/05/23 | Zahlen J. | Added information regarding Evaluation. |
| 1.2 | 02/19/23 | Zahlen J. | Added 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](#_heading=h.35nkun2)

Client *The Gaming Room* seeks to develop a web-based game to expand the reach of their game *Draw It or Lose It,* based on the 80s’ game, *Win, Lose or Draw*; which is currently only available on android platforms.

## [Design Constraints](#_heading=h.1ksv4uv) // I removed requirements as it felt redundant being followed by this.

The following are the design constraints laid out by the client:

* The App must be web-based to expand availability across different platforms.
* Must have the ability to have one or more teams involved.
* Each Team must have multiple players.
* Game as well as team names must be unique to easily distinguish between them.
* Only one instance of a game can exist in memory at a time. - Accomplished by creating unique identifiers for each instance of a game, team, or player.

The following are the rules on the game:

* The game consists of four rounds, each lasting one minute.
* The application renders images from a large library of stock drawings as clues.
* Each drawing is rendered at a steady pace until being fully revealed 30 seconds into each round.
* If the team does not guess the puzzle before the timer expires, the remaining teams each have 15 seconds to come up with one guess to solve the puzzle.

## [Domain Model](#_heading=h.2jxsxqh)

The Game, Team, and Player classes are all handled by the Entity class from which they all inherit characteristics. The Game, Team, Player, and GameService classes all reference each other. The ProgramDriver class acts as our main driver for the application and accesses all of our classes to execute them as well as uses our SingletonTester class to ensure each instance of our application runs according to its design constraints.

**"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](#_heading=h.z337ya)

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 is very accessible and its servers are fairly easy to configure.  It’s GUI is also fairly easy to use and understand and it has flexible terminal commands. | Linux is very cost friendly due to its status as openSource. However, it is fairly difficult to navigate without prior experience/knowledge. Utilizes Linux shell for simple configuration of the server. | Windows is very expensive on the server side of things. The most expensive by far, but like Apple, its GUI is very user-friendly and common-spread enough that most people will have had prior experience using it. | Mobile devices vary device to device and user to user and honestly, the optimization of other platforms is better. They are fairly user-friendly though (by design of course) |
| **Client Side** | As macOS is not openSource, there are costs to incur and can be fairly expensive for end-users. Moderate time and expertise is needed to navigate the OS but it’s been a main-stay since the 90’s and a decent portion of the population have the required experience. | Massive MASSIVE amounts of experience and time needed to fully utilize the OS. | Moderate costs and accessibility. Easy to learn and like macOS, most of the population already has a baseline understanding of utilizing it. | Very flexible for both clients and developers, allowing you to see and make updates from basically anywhere. Can be a bit more difficult implementation-wise than other platforms. |
| **Development Tools** | Supports most commonly used programming languages. HTML, CSS, Java, etc. Extensive libraries to support frontend development. Access to a plethora of IDEs and tools such as GitHub, etc. | As most desktop based platforms, it shares most of its development tools with both windows and macOS. May not have access to certain tools created for windows/mac | Probably the most accessible of all. Has all the previously aforementioned, as well as tools such as CMD, and other windows features that are excellent additions when it comes to development. | Most limited. Becoming more accessible and more in-line with desktop, but obviously the platform’s age and use-cases are its limitation. (but has most of the same tools as the rest) |

**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 definitely see Windows being the best platform to allow our client to expand their game to other platforms. With the largest market share of any, you’d hardly ever run into any true constraints especially with the application being web-based.
2. **Operating Systems Architectures**: Windows, Developed and published by tech giant, Microsoft, provides an excellent way to store files, run software, play games, consume content, and browse the internet. It’s GUI is super simplistic and allows for easy navigation. Not to mention its expansive user-base comes with support for some of the most commonly used IDEs like VStudio, JetBeans, etc.
3. **Storage Management**: WindowsOS manages storage with simple configuration allowing for those working the back-end to navigate easily through the settings. While owning a windows server carries a hefty cost, it’s well worth the investment with the accessibility of expansion if the need ever arises for our client.
4. **Memory Management**: Working with Windows, there are two options of memory to utilize. Physical and VRAM. VRAm is the optimal choice and allows for large programs to be handled effectively and efficiently. One of the primary advantages is its memory protection.
5. **Distributed Systems and Networks**: There are definitely issues when it comes to working with windows. There’s often issues with lag, queuing issues, overloaded serves. Most of these problems are fairly avoidable through solid communication between development team members.
6. **Security**: Windows has a host of security features in place. The primary one that comes to mind is the included Windows Defender antivirus program that comes installed on every Windows device. There’s also access to an extensive list of VPN services.