

# Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 3/4/2021 | Maggee Duprey | Filled information within the headers in the executive summary, design constraints and summarized the domain. |
| 1.1 | 3/21/2021 | Maggee Duprey | Filled out the boxes for the evaluation table. |
| 1.2 | 4/22/2021 | Maggee Duprey | Input personal recommendations for the client. |

**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 a web-based-game that can serve multiple platforms based on its popular

Android game, Draw It or Lose It. This game would be like Win, Lose or Draw, a popular game show from

the 1980s.

## [Design Constraints](#_2et92p0)

* Game needs to be run on multiple platforms.
* Multiple teams of multiple people
* Checks for unique team/game names must be present.
* Only one instance of the game must be allowed at any one time.

These constraints are what the game itself needs. In terms of development, The Gaming Room would

like this game to be run on multiple platforms, so we would need to take the working Android code and

port it over to the other platforms, which may require more developers to write this code in other languages.

## [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 is our super class, and the Game, Team and Player classes all inherit directly from it. The

GameService has a reference to the Game class, which has a reference to the Team class which has a

reference to the Player class. All these classes having a reference to another is done through aggregation. Also included are our SingletonTester and our ProgramDriver class. Our ProgramDriver class is where any executions of our application take place, and it uses our SingletonTester class. The

SingletonTester class also has a use relationship with the ProgramDriver class. The way these classes are set up allows for one game to go on at a time with multiple teams (from a list) with multiple players (from a list).

****

## [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** | Adaptive terminal  commands to  configure the  server or make changes, as well as  a fantastic GUI for  ease of use. | Adaptive terminal  commands to  configure the  server or make changes in a more  cost effective manner. Though  while cost  effective, it is more difficult to  navigate, but like  Mac, its command  shell makes for  excellent server  configuration. | More software  available  compared to  other operating  systems, and like  Mac servers,  Windows servers  are expensive, but  the GUI is very user friendly with  plenty of options  for developers via  the command  prompt. | It is better to keep the server stable and immobile for  easier tracking, and  this overall does  vary by user. The  challenge is making the game compatible across  all mobile platforms  and services. |
| **Client Side** | Mac users require average amount of  time. Accurate  skills needed to  navigate os.  Expensive option  for users. | Linux users need maximum time and proficiency to  support Linux  systems.  Minimum cost for  Linux users. Linux  knowledge is required to use this operating system. | User does not  need a lot of time to understand how to support a  Windows setup.  Cost is more expensive than  that of Linux  systems. | Users need to  commit a lot of time and skill to support mobile devices.  Mobile platforms  are difficult to  perform applications created for other platforms. |
| **Development Tools** | HTML/CSS/JavaScri  pt with supporting  libraries to support  frontend and other  languages like  Python, Java, php,  Ruby.  Macs can use Swift specifically, as it is Apple’s designated language while adding something like notepad++ for a tool. | HTML/CSS/JavaScript with supporting  libraries to  support frontend  and other  languages like  Python, Java, php,  Ruby.  Linux runs great with Visual Studio.  Specifically, with  notepad++. | HTML/CSS/JavaScript with  supporting  libraries to  support frontend  and other  languages like  Python, Java, php,  Ruby.  Can do the same as Linux and is easier to use. | HTML/CSS/JavaScript with supporting  libraries to support  frontend and other  languages like  Python, Java, php,  Ruby.  Java, Android, and  Swift can be used to  create thousands of  apps, very efficient  languages  depending on the  designated OS. |

## 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:** For the Gaming Room’s “Draw It or Lose It” application, the Windows OS would be best suited for the expansion. Windows OS has a protected supervisor mode, and this can ensure security and reliability. Windows also has likely the largest user base when it comes to desktop operating systems. There will be more ease for developers to create the application for windows because many more software packages are available to it exclusively. This will also give the development team more room to work with different web servers, file servers, application servers, mail servers, database servers, print servers, and a minimum expectation of expertise to manage and maintain the project as well as future projects due to various software.
2. **Operating Systems Architectures**: Windows OS utilizes an architecture that caters to a system with multiple processors. Users can store information in a hard drive or solid-state drive and have random access memory. Windows allows for a safe mode login as well to safely navigate the systems storage without possibly harming anything else on the computer.
3. **Storage Management:** For storage management, Windows systems can have a variety of different options. The type I would typically recommend would be a solid-state drive (SSD). This is as apposed to an HDD. SSDs allow for screens and applications to startup faster. It seems lie “Draw it or Lose it” will mostly deal in loading different screens and images o this would be the best option.
4. **Memory Management:** Since the game will deal in retrieving and presenting from a plethora of different images, I believe that a decent amount of RAM would be suffice for memory management. Like I have studied before, the estimated photos would take up about 1.6 gb, and that is not including any other features of the game itself. A system of at least 4gb would be the safest bet. Luckily, most premade systems today usually have at least that much if not more.
5. **Distributed Systems and Networks:** To look into distributing the game to different systems, I believe the best bet would be to investigate a multi-platform cloud service and develop from there. That way it is less work in backend development of the bones of the game as it will be runnable from different types of systems. This in the long run would be great for the popularity of the game as it would draw more players.
6. **Security:** Windows OS offers a built-in security feature called “Windows Defender”. It is good when the operating system itself has security features built in because the system itself should know itself better and how to find issues better than most other resources. There are also outside parties that make security programs such as McAfee and Norton. It is good to have connections with those companies if you would like the added advertisement of showing your application is protected by one of them. It can work to make a user feel safe using it and that it is a legitimate application.