

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 5/21/2022 | Tanner Glazier | Changed all information on all pages |
| 2.0 | 6/12/2022 | Tanner Glazier | Added content to evaluation section |
| 3.0 | 6/19/2022 | Tanner Glazier | 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](#_sbfa50wo7nsh)

The gaming room wants to setup a web-based game that any user can run on any platform they choose. They game name they have decided to go with is called Draw It or Lose it and they already have a version that is on Android devices. The games rules are simple, it requires multiple teams of several people doing four rounds each lasting one minute. A picture is pulled from a library of images and then each team must guess until the team either guesses it or the time runs out. If the first round of people doesn’t get it, it goes to the opposing team member, and they have 15 seconds to guess the picture.

## [Design Constraints](#_2et92p0)

One of the biggest constraints is the ability for it to run on different devices and operating systems. This is challenging because all operation systems are different from each other, and each have different programming requirements. Another constraint is that it requires multiple people to play the game and more than one team involved, this could cause issues with connection, if a player drops connection, it could cause issues with gameplay.

## [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 makes a connection between the Game, Player, and Team classes. What that means is that everything in those classes gets information from the Entity class. Each class will share common references like a name, or ID. We can also see that Game Services directly connects to Games which then also flows back into Entity making Entity a super class.

**"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 must 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** | Has great options for different web hosting requirements.  Has an easy-to-use terminal and commands to configure servers easier.  The main disadvantage is that although good for web hosting it isn’t very popular option for it. | A more cost-effective solution  Very secure and most issues are seen before they ever become a problem. Linux is the most preferred web hosting option.  A disadvantage is that is can be difficult to find applications that support Linux. | Has the most compatible software solutions.  It is dominate to all the other platforms; more devices use Windows.  Disadvantages are it requires more resources to run Windows and can have higher load times and more susceptible to viruses. | Most servers are better suited to not be mobile and stay in one place, along with server specifications, mobile devices aren’t there yet.  Can have poorer security to all the other devices. |
| **Client Side** | Very limited experience needed to get started.  Has a slightly higher cost than Windows devices.  Has compatibility for all major web browsers and mobile devices. | Expertise and time needed to setup.  Cost is very minimum compared to the other operating systems.  Supports all major web browsers and easy to develop on. | Easier to use than Linux and can be compared to Mac OS.  Cost is a little cheaper than Mac devices but more than Linux based devices  Supports all browsers and more software than the other systems. | Gives the user and developers the ability to see the application from any place.  Can be slightly more difficult to implement than the other devices. |
| **Development Tools** | Swift is the most popular options for Mac. We can use either Notepad ++ or Visual Studio can also run-on Mac devices. Mac can run most languages like HTML, CSS, and JavaScript and supports Java, Python, PHP, and Ruby. | Linux can use Visual Studio, Eclipse and Notepad ++ for easy-to-use tools. Like Mac it supports all the same languages. | Easier to setup than Linux it also supports Visual Studio, Eclipse, and Notepad ++. Like the other operating systems, it supports all the same languages for backend and frontend. | Android and Swift give developer’s opportunities to create any app they want and any of the software can be run on all three other operation systems. The languages consist of HTML/CSS/JavaScript covering the frontend and Java, Python, PHP, and Ruby for all backend operations. |

## 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 recommend to the company to start off with Windows based devices, they are the most use operating platform, other than mobile devices and it supports a lot of applications and along with the low cost and easy to use features it will be an easier starting point.
2. **Operating Systems Architectures**: Windows has a graphical user interface (GUI) that allows applications to show what is going on along with system resources and much more.
3. **Storage Management**: Windows 10 allows users to manage storage easily by setting up new drives automatically when attached to the computer and allows for plug-and-play options so if an external drive is connected it automatically shows up for the user. The system also gives users options to connect to external devices by mapping to the user’s computer easily allowing them to access anything on the go.
4. **Memory Management**: While making the game, creating databases and libraries will be done allowing for easier memory allocation outside of storage. Memory can store the information for a period making for image loading faster.
5. **Distributed Systems and Networks**: Each operating system is different, for the most part Linux, Mac and Linux games can run the same on each platform, but porting to phone’s can be tricky and may require programming a whole new thing using Swift.
6. **Security**: Windows has a built-in security feature called Windows Defender; this would protect what is on the user’s machine. If we are talking about logging in users to a web portal, then we would want to have a secure database along with hashing & salting of passwords to make sure that nobody can get user information.