

<Draw It Or Lose It>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

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

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

[**Document Revision History**](#_3znysh7)2

[**Executive Summary**](#_2et92p0)3

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

[**System Architecture View**](#_3dy6vkm)3

[**Domain Model**](#_1t3h5sf)3

[**Evaluation**](#_2s8eyo1)3

[**Recommendations**](#_3rdcrjn)5

## [Document Revision History](#_3znysh7)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/26/2022 | Walker Karnes | Completed the design document. |
| 1.1 | 06/05/2022 | Walker Karnes | Reviewed the design document and ensured that I completed the sections required for Project Two.. |
| 1.2 | 06/21/2022 | Walker Karnes | Revised the recommendation section for Project Three. |

**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](#_2et92p0)

The Gaming Room needs a web-based-game that can serve multiple platforms based on its popularAndroid game, Draw It or Lose It. This game would be like Win, Lose or Draw, a popular game show from the 1980s.

**Design Constraints**

* There is only one instance of the game allowed at a time.
* Many teams are made up of several players.
* Checks for unique names must be present.
* The game must be able to run on multiple different platforms.

## [System Architecture View](#_3dy6vkm)

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](#_1t3h5sf)

There are several classes found within the UML Class diagram below. Among these are the entity, game, team, player, game service, program driver, and singleton tester classes. The game, team, and player classes are all children classes to the entity class. They all inherit the basic components of the entity class. An object from the team class or a team can be made up of zero or many player objects. The game object can be made up of zero or many teams. And the game service can be made of zero or many games. The game, team, and player class are all related through aggregation because they can exist independently but also are made up of each other.

**"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](#_2s8eyo1)

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** | Mac is well known for being safe for users. The safari browser comes with built-in protection. But, Mac does not support lots of software that other operating systems do. So this may be a limiting factor for development. | Linux is extremely customizable. You could achieve basically anything you wanted from developing a web-based software using Linux. And you are not likely to run into compatibility issues here. | Windows is the best operating system from my experience as far as processing speed and reliability goes. However, their base virus protection could leave users vulnerable when interacting with a web-based application. | A good web-based application will likely still run smoothly on a mobile device even though they don’t have the processing power of computer. The main thing to take in to consideration is the smaller scale on the screen. The screen has to adapt to fit well in a mobile device orientation and this could be a lot of trouble. |
| **Client Side** | As I said on the server side, macs do not support all of the software that other machines do. So, in development we will have to spend a little more time and money to ensure that our software is available to mac users as well. | Linux users will have to pay less because Linux is the cheapest of the four OS. However, maximum expertise and time would be required to support the Linux systems. | Windows is slightly more expensive than Linux and cheaper than Mac. Most people are familiar with Windows OS so, a moderate amount of expertise and time would be required to support a Windows system. | Mobile devices would be the most difficult OS to choose as everything doesn’t always scale down for mobile devices. You would need to put maximum expertise and time into developing the application to ensure it ran smoothly on mobile devices. |
| **Development Tools** | Some relevant programming languages would be Swift, C++, Java, Python, and many more. There are several popular IDEs for software on the Mac but the most prevalent is Visual Studio Code, Atom, Netbeans, pyCharm, and Eclipse. | The most relevant languages for development on Linux would be C, Java, Python, JavaScript, Shell, and many more. The most popular IDEs for Linux development are Eclipse, Netbeans, Atom, and Code::Blocks. | There are many popular languages for Windows development and these include but are not limited to C, C++, Java, Python, JavaScript, and Swift. The most popular IDEs for Windows are Visual Studio, Eclipse, PyCharm, and Netbeans. | The most relevant languages for mobile application development would include Java, Swift, Kotlin, Python, and React-Native. Among the most popular IDEs for mobile development are Android Studio, Xcode, QT IDE, Eclipse, and Visual Studio. |

## 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 recommend that Gaming Room uses Windows for the development for “Draw It or Lose It”. There are vast resources and IDEs for development on Windows. Windows supports minimal expertise requirements as most people use Windows and would be very cost effective as most good software development tools are free for windows.
2. **Operating Systems Architectures**: Windows comes with a very in depth file management system that allows users to manage large amounts of files and implement them together. This allows for large applications to be developed in several programming languages if need be.
3. **Storage Management:** Most windows machines come set up with large amounts of storage and it is very straightforward to upgrade storage if the need arises. Also windows supports the use of cloud servers so you could even extend storage without having to physically add another SSD or something similar. Windows also come with a very handy Disk Management tool. This allows you to manage your storage at the next level by giving you several options such as extending partitions, creating virtual hard drives, and converting from MBR to GPT. These are not features that will be used much by an everyday user but it does allow for that next level of storage management.
4. **Memory Management**: Windows machines come with a memory management system that gives you a detailed view of how much memory is being used up by all of your tasks. All you have to do is click Ctrl + Alt + Delete and click the task manager. This brings up a detailed view of every task that is on your computer at that moment and you can manually go through and shut down any process that is using too much memory.
5. **Distributed Systems and Networks**: Using windows servers allows for simple communication between several different platforms and computers concurrently. Implementing networking support will allow the Gaming Room to ensure that there are no issues with communication between the servers. If the game picks up a large user base there could be issues with congestion on the servers but this can be solved with a team of skilled networking specialists.
6. **Security**: Windows allows users to control what information is being collected or not. It also has a free virus protection that alerts you of potential risky sites and packages that users may encounter. Also, windows supports many different kinds of VPNs that could keep user information protected from hackers. Windows has also implemented a very in depth login for their OS. They have a biometric scan that scans the users face and this is a very simple way to make sure that no one else can use your computer. Not only does this increase security but it is also faster to let the computer scan your face as opposed to entering a passcode every time. While someone could steal your password, no one is going to be able to bypass your face scan.