

# **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)4

[**Evaluation**](#_2o15spng8stw)4

[**Recommendations**](#_m8aleynsvzvc)8

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 3.0 | 04/12/2022 | Breunna Bingham | Updated the 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 develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which currently is only available for Android. The game will consist of several teams playing a game that has four rounds lasting for a minute each. An image appears from a large library of stock drawings and the team has 30 seconds to guess the puzzle. If the puzzle is not chosen correctly by the current team within the 30 seconds, the remaining teams have 15 seconds to make one guess each to solve the puzzle.

## [Design Constraints](#_2et92p0)

1. Game needs to run on multiple platforms
   1. This is a requirement from The Gaming Room to be able to run on more then just Android devices. This will take more manpower and time to complete since we must work on several separate coding projects for Apple, Windows, and Linux. This also means considering what languages will be best to write the code in for each platform.
2. We need to ensure only one gaming instance is taking place at any time.
   1. As per the requirements from The Gaming Room, it’s important that we only have one instance of a game at any time. This means checking for pre-existing games saved under a specific name.
3. Code that is written needs to check via iterator pattern if a prior Game or Team name is in use already or available to be used.
   1. Same as stated before. An iterator pattern will help to weed out if there is any pre-existing name for Game or Team that has a saved game. If it is in use, the game will simply pick up where it left off, if it is not in use, it will create a new saved file under the new name.
4. The game needs to allow for multiple people and teams to be implemented at the same time.
   1. As per the requirements of The Gaming Room, they want multiple people to make up a team and multiple teams to make up a game, for this game to work properly.

## [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)

We have a new class Entity that is linked to three subclasses, Game, Team and Player. Game, Team and Player all inherit from Entity and each have a ‘is a’ relationship with Entity. It is also important to note that each of these three classes also all share the attribute of ‘id’ and the attribute of ‘name’ from Entity which is the Superclass. If we look at GameService we can see that it has a reference to Game which then has a reference to Team and Team has a reference to Player. Each of these relationships is a ‘has a’ type of relationship because if you look at it in other words, each Team has Players, and each Game has Teams and each GameService has Games.

**"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 Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The servers for MacOS can be very beneficial, as they have a series of advantages. Mac is more streamlined and flexible for development, especially if the entire team is within the same network. A weakness is that the entire team would have to operate on MacOS and that can be costly. | Linux has a lot of advantages that MacOS does, but in addition to that it is very strong and reliable. A weakness is Linux doesn’t have the wide range of software available that windows and even Mac has, and it is also less supportive. | Windows is very easy and simple to set up and has a wide range of available software that is compatible with lots of support. A weakness is that the reliability isn’t all that great. Linux almost never needs rebooting, but with Windows, rebooting is necessary to make sure it is reliable. | Mobile devices used for hosting a web-based software isn’t something that I’ve heard a lot of people doing. Normally people go with a stationary server to avoid loss of internet connection due to location. While the idea could have some potential benefits such as being readily accessible around the world. However, there are way more cons to this then pros. The first weakness that comes to mind, is lack of internet connection. If the mobile device can’t connect to the web, then how will it be able to host a web-based software application? |
| **Client Side** | Like the server side, the team would have to all be on the same network and use MacOS for development, which can prove to be costly. Also, the development team would have to excel in the same coding language that Mac uses which is Objective -C. Time frame would be about the same for development as Windows. | Linux is a very strong learning curve for most programmers. This specific OS would take a lot more time to develop then Windows or Mac. The advantage here is that it is extremely cost saving to use this OS as compared to Windows or MacOS. | Windows would be easy for developers to develop a program, as most programmers have experience in one or more languages supported on Windows. There is a wide range of coding languages that are available, and the completion time would be similar to MacOS. Windows would also be comparable to MacOS for cost although it may prove to be slightly cheaper then MacOS, but not nearly as inexpensive as Linux. | For software development of the Mobile devices, A benefit would be accessibility at any time and place to clients and developers if needed, but it would be costly. We would have to have two separate teams one for Android and one for iOS. Each team would have to excel in the language chosen to develop on each platform and have experience in it. It would take more time developing for two separate programs then developing on a single iOS such as Windows or Mac OS. |
| **Development Tools** | The coding language that is used most commly in MacOS is Objective-C as I stated above. The IDE that can be used would be Xcode for Objective-C language. A fantastic tool that could be used to develop this web application is CodeKit. | Depending on what language is chosen, there is a nice selection, Java, Python, C++, C to name a few. IDE’s and other tools could be PyCharm, Eclipse, and Visual Studio, Rad Studio | Also again, a wide variety of coding languages to choose from, similar to Linux. Java, Python, C++, C to name a few. IDE’s and other tools could be PyCharm, Eclipse, and Visual Studio, Rad Studio, Jira | Development would be split into iOS and Android.  For Android most commonly programmers will use Java. Eclipse would be a great IDE.  For iOS, most commonly programmers will use Swift or Objective-C. XCode and AppCode would be a great IDE/ tools to use. |

## 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 the Windows operating (server) platform for this specific program. I believe that it houses everything necessary for developing a reliable server-based application that can be expanded across several computing environments.
2. **Operating Systems Architectures**: There are two options for the architecture of the windows operating system. One being the x86 which is very powerful. It has different levels to allow certain users access only to specific levels to protect data that is stored within. There is also the Windows NT kernel mode which has several layers and more layers that is utilized then x86. The only problem with x86 is that we may run into compatibility issues with the different computing environments. Therefore, I would recommend Windows NT kernel mode for the architecture choice for this program.
3. **Storage Management**: For this program to run efficiently we need to consider the storage space that will be utilized. Are we using 200 high-definition photos that will be recycled or are we using 200 high-definition photos with the expanding ability to add up to 400 photos? For this reason, I would recommend we use an SRM or Storage Resource Management tool such to aid in providing the necessary storage space needed for the game to run smoothly and efficiently.
4. **Memory Management**: For this program we need to consider how much RAM will be needed to run this program across several different computing environments. We can consider using an SSD for the program to run on. It can give a seamless experience to the user, but we also must note that SSDs are not always available. In this case we should be utilizing our cache memory to deliver a seamless experience to the user without overloading the RAM and causing lag on the user end. Therefore, I recommend either using an SSD or strategically using cache memory to our advantage while developing this program on windows.
5. **Distributed Systems and Networks**: As we know the game Draw It or Lose It is meant to be played across several platforms. This means various software that can communicate with one another. There are multiple coding languages that we can use to communicate between different platforms and connect them all via a single server that can handle the multiple platforms. Since we are using a windows-based server platform, we know that outages and connectivity problems may arise more then say if we have chosen Linux. We also need to pay close attention to lag between networks as the game progresses. If lag is seen it can inhibit the actual anticipated and expected gameplay considering that a new image is shown live to everyone every 30 seconds. And within those 30 seconds people are allowed to make live guesses. It is imperative that we ensure superior interconnectivity between all platforms for a seamless gaming experience for everyone.
6. **Security**: Security is a very important aspect of this program. Especially if our server will store any user information. Since we are developing on windows, we can use some of their more prominent security features. We can run constant scans to detect any viruses and/or threats. Microsoft Windows Defender is a great tool that we can use. We can also utilize a firewall. As stated, before the architecture of Windows NT kernel mode which has several layers, also helps to lock out any unauthorized users from different layers. It helps to prevent hackers from getting to the deepest layers which store the most protected information. Perhaps using two factor authorizations when gaining access to security levels would be a great addition to improving security measures, as well as only giving certain people access to individual levels of the OS to further protect data from being stolen or over written.