

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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)4

[**Recommendations**](#_m8aleynsvzvc)6

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 7/24/2022 | Douglas Buckley | Made recommendations to the preferred operating system that The Gaming Room should try to expand the services of the game onto next. Design constraints were discussed as well as an overview of the UML Class diagram showcasing the relationship the classes have with one another. Recommendations for the project are mentioned at the end of the design report. |
| 1.1 | 8/7/2022 | Douglas Buckley | Revised the **Evaluation** section of the report. Went into further detail for the pros and cons of using different operating systems in regard to the Server side, Client side, and the development tools of each operating system. |
| 1.2 | 8/14/2022 | Douglas Buckley | Revised the **Recommendations** section of the design report. |

**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 would like to develop a web-based game titled “Draw It or Lose It” which is loosely based on the 1980’s television game show “Win, Lose, or Draw”. There will be multiple teams participating in the game but there will be a guessing team that has 30 seconds to guess what image is being drawn in front of them. If the guess is incorrect or the team runs out of time, then the opposing teams will have 15 seconds to guess the image to win the points. Currently, “Draw It or Lose It” is available exclusively on Android devices.

## [Design Constraints](#_2et92p0)

As per the requested requirements of The Gaming Room, the following designs must be included in the application:

* **Each game should be able to have one or more teams involved**
* **Each team should have multiple people assigned**
* **Game and team names MUST BE unique to allow other users to check whether a name is in use or not when choosing a team name**
* **Only one instance of the game can exist in memory at any given time. This should be done by creating unique identifiers for each instance of the game, player, or team.**

There are a few of the design constraints that I would like to examine in further detail. For

example, The Gaming Room wants the application to be able to run on all platforms and devices, however at the moment the application is only functional on devices that have the Android operating system. The application still needs to be configured so that it can operate correctly on the other operating systems (Apples, Linux, Windows). This can be done by looking at the source code on a device capable of running it with the proper programming language that can configure it.

## [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)

Entity, the class at the top, has a relationship with the three classes beneath it. The classes are the Game, Team, and Player class. The three listed classes all inherit their data from the Entity class, which seems to be the “id” and “name” attributes in this case. When looking at the relationship between the three classes that inherit from the Entity class, we can see that the classes Player and Team have a connected relationship. This happens to be a “has a” type of relationship, meaning that one of the classes has a reference to the other class. In this case, the Team class references the Player class, the Game class references the Team class, and the GameService class references the Game 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 has to work together.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The Mac OS does offer server-based deployment methods for the website to be hosted. Having a MacOS server will benefit other Mac users greatly since it will have the most support for applications that run on Mac, but it is capable of supporting general admin applications as well.  However, one of the downsides to a Mac OS specific server is that the price of upkeeping the server can be expensive depending on the number of machines connected to the server. In addition, the MacOS wouldn’t be the best option if the companies had to rely on third-party programs or applications for the website to function properly since it relies mostly on its unique Mac software. | One of the benefits that come with using a Linux based server is that it’s a free and open source. Linux is also known for being thoroughly customizable, meaning that the security of it can be designed to be as lenient or restricting as the company desires it to be.  However, one of the biggest cons that come with having a Linux server is the learning curve that comes it. | The Windows Operating System, like the other two mentioned, is also capable of having website hosting capabilities. Unlike Linux, Windows is considered a proprietary software, meaning that the purchase for licensing is required for use. The con to this is that the cost can be expensive, ranging from $500 to about $6,200.  Windows, unlike macOS, is capable of running a large majority of third-party applications and software. Windows makes easy patches to the software and hardware a very easy and streamlined process. | Although this method of hosting the web-based application is not very common, it’s still possible to accomplish it through this method. Oracle is a company that offers the support of mobile server application hosting. It's capable of supporting platforms such as Android, iOS, both mobile and desktop versions of Windows and Linux, and Blackberry. |
| **Client Side** | Moderate amount of expertise and time is required.  Cost happens to be similar to the Windows operating system. | Requires the most amount of expertise and time required. On the plus side, the cost happens to be the lowest out of all the operating systems. | Minimum amount of expertise and time is required. The cost is close to that of the Mac operating system. | Allows for a wide range of flexibility to clients and even developers to see updates at any place.  However, due to the nature of mobile devices, its slightly more difficult to implement than any other device. |
| **Development Tools** | Mac operating system has a popular language known as Swift, while being able to use other popular options such as Notepad++. It should be noted however that Mac is capable of running all programming languages. Some examples are but not limited to CSS, JavaScript, and HTML while supporting libraries that are used to support the frontend and general languages. These languages can be but not limited to Java, Python, PHP, and Ruby. | Linux, while also capable of notepad++, can work with IDEs such as Visual Studio and Eclipse. Similar to the Mac operating system, Linux is capable or working with languages such as HTML, JavaScript, and CSS while supporting the libraries that are used to support the front-end development languages like Java and Python. | Capable of working with development tools such as PyCharm, Eclipse, Visual Studio, VSCode and more. Similar to the previous two languages Windows is capable of running languages such as JavaScript, HTML, and CSS while supporting the libraries used to support the front-end languages such as Java, Python, PHP, or Ruby. | It's possible to create a large variety of applications while using the Android and Swift languages. Both of the languages and their respected software are capable of running on of the three main operating systems. Similar to all the other operating systems, mobile devices are capable of using languages such as HTML, CSS, and JavaScript while supporting libraries that support the front- languages such as Python, Java, Ruby, and PHP. |

## 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**: Based on the evaluation of all the operating systems, it would be advised that The Gaming Room were to use the Linux operating platform to host the game. Reason being is that it’s a free and open source, meaning they would save on money where as the other operating systems would require some form of payment for either server upkeep, licensing, or both.
2. **Operating Systems Architectures**: Most operating systems have similar architectures for their core constructs. They all contain hardware, kernel, and shells. Hardware consists of physical components such as a CPU, hard drive, and so on. A kernel is a form of bridge that connects the data being processed by the hardware to the applications. Lastly, a shell is a type of software that allows for the user to interact with the operating system.
3. **Storage Management**: When it comes to potential storage management, using the Cloud storage could be beneficial to The Gaming Room. IF the player demand for the game were to begin rapidly increasing, it’s a lot easier to increase the storage capacity of the cloud rather than doing it with physical devices would cost more in purchasing equipment and maintenance.
4. **Memory Management**: The application “Draw It or Lose It” will need to have a database full of images to refer to be able to draw for the teams to guess. Windows allows for memory allocation of pictures outside of the default “Pictures” folder that it comes with. This means that The Gaming Room is able to upload and download images into the desired folder to allow all of the contents of the game to be close together and even stored into a secure folder.
5. **Distributed Systems and Networks**: Since The Gaming Room is wanting to make the game capable of running on all platforms, then they will need to make sure that the server that will hose the game is capable of being accessed by all other devices with the varying operating systems. In addition to this, The Gaming Room will need to make sure that the servers they wish to use are capable of hosting large player traffic along with having additional power applied to the server (i.e., a generator or solar battery tank) in the event of outages.
6. **Security**: Linux allows for an insane amount of customization when it comes to the security setup. The Gaming Room could decide how loose or strict their security measures are for the data of the game.