

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 | <05/22/22> | Madison Lopert | Added a summary and design constraints. Described the provided UML diagram. Compared the server side, client side, and development tools between Mac, Android, Windows, and mobile devices. Provided my recommendation and details about that operating system. |

**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)

Draw It or Lose It is a current Android mobile game. The Gaming Room is looking to develop this game as a web-based game that is available on multiple platforms.

## [Design Constraints](#_2et92p0)

* Needs to have the ability to have one or more teams involves
* Each team should have multiple players
* Game and team names should be unique
* Only one instance of the game can exist in memory at a given time

## [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)

Game, Team, and Player are all linked to Entity through the inheritance symbol. This means they inherit the same attributes, id and name, from Entity. The classes GameService, Game, Team, and Player are all connected with zero or more association symbols. This means that GameService has a reference to Game, Game has a reference to Team, and Team has a reference to Player. This makes sense as games have teams, teams have players, etc. This is what will give the program the ability to assign more than one team to a game, or more than one player to a team. In the top left of the diagram, you can see the SingletonTester. This isn’t directly associated to any of the classes but can be run to test the singleton classes functionality. This is important to meet the unique names and one instance of the game design restraints.

**"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** | Mac has flexible command line terminal, like Linux, that is rated highly by users. Mac hosts also run-on Apache servers that allows for basic web code software like WordPress to run smoothly. Mac hardware is often expensive though. | Linux is great option for developers on a budget as it is free. Their command line is flexible, Like Mac, they are also run-on Apache servers. Linux isn’t often considered as secure as Mac, but it greatly depends on the security measures offered by the web host. | Windows command line isn’t like Mac or Linux and requires a learning curve. Windows is used often and there are many software’s available for use. Web code software such as WordPress is often choppier than Mac and Linux. | Mobile devices are in a different format than the others and require a much different perspective from the developer to make. They are popular and great for expanding outreach to a wider audience. Backend servers are available to buy to run the application. |
| **Client Side** | Mac requires a level of familiarity with the OS to create an application but not too much. It is quite easy to get used to and the web host will make it easy to view on multiple platforms.  It costs about the same as Windows to get running due to hardware and subscription costs. | Linux is the cheapest since it is free and open source. Unlike the Mac and Windows, it requires the most expertise to use. It isn’t user-friendly and has a large learning curve. | Windows probably requires the least amount of expertise to use. It is widely used and there are many resources to help guide a developer in the creation of an application. Like Mac, it can be expensive due to hardware and subscription costs. | Mobile apps aren’t easy to develop. There is a lot to consider when created a mobile app such as screen size, battery usage, and data. This would require a good amount of expertise to understand. It is more expensive than Linux due to the price of software, subscriptions, etc. It is often not as expensive as Windows and Mac. |
| **Development Tools** | Mac has many IDE’s available such as Visual Studio Code, Atom, and Eclipse. It does have its own language though called Swift that is available for use. For web-based apps we would likely use HTML, CSS, and JavaScript. | Linux provides access to some cool IDE’s such as Eclipse, Bluefish, IntelliJ IDEA, etc. For web-based applications we will likely use HTML, CSS, and JavaScript. | Windows provides many IDE’s such as Visual Studio, NetBeans, Eclipse, WebStorm, etc. For web-based applications we will likely use HTML, CSS, and JavaScript. For all OS systems there are available libraries that allow the use of other programming languages such as Java. | The IDE for mobile devices greatly depends on which OS it is built for. Android apps can use Android Studio. IOS apps can use XCode. There are other IDE’s available though such as Qt IDE, JetBrains, and Eclipse. Like the others, it will likely use HTML, CSS, or JavaScript unless the IDE requires a specific alternative. |

## 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**: Overall I would choose the Mac operating server platform to create a web-based application. It has great security and reliable hardware behind it. It is more user friendly than Linux and more practical than a mobile operating system starting out. It costs about the same as Windows, but it seems to rate higher in security and has a more flexible command terminal.
2. **Operating Systems Architectures**: Mac’s architecture is made up of four main parts. The interface, applications, graphics, and core OS. Mac is built on a Unix core called Darwin that focuses on data flowing to and from the CPU, memory use, etc. The graphics subsystem has three main components Quartz Extreme, OpenGL, and QuickTime that focus on 2D graphics, fonts, and other elements. There is also an application subsystem that allows you to run application environments such as Carbon and Java 2. The interface for Mac is called Aqua and is built to be a user-friendly experience full of color and texture.
3. **Storage Management**: The storage management can depend on the web host’s drives and SSD’s. If memory is stored locally on the Mac, it can be stored in the Finder application. If a new SSD is needed it can be installed to keep up with all the data. Most of the storage management can be managed by the web host but some companies find it more secure and trustworthy to manage the servers themselves. Web hosting companies can differ a lot in price range depending on the services they offer, finding a host that specializes in security and a large amount of storage space would be ideal for most projects.

1. **Memory Management**: A Mac’s RAM is used by its OS, applications, and other software elements. It separates the RAM into sections, the system partition and application partition. This allows the user to run multiple applications at once and optimize the memory being used. The system partition is for when the operating system first starts up and the application partition is used when an application is launched. Each hold different sets of memory and only allot what is necessary to run.
2. **Distributed Systems and Networks**: There are many cross-platform tools that allow a game to spread among many different platforms. Since it is web-based, it could be accessible through a cloud host that utilizes many servers. This way if an outage was to happen it can locate the resources needed from another server, minimizing down time. I know Mac has many options to choose from like BlueHost, WPEngine, and SiteGround. Mostly every operating system can access the web, being held on the cloud, it can be accessed and utilized on almost every platform. Java itself, which was used for developing this application, can be compiled and ran on all platforms so there shouldn’t be any compatibility issues.
3. **Security**: Mac has many built-in security settings that allows you to customize what is enabled or disabled. For example, they don’t allow Flash games and you can disable JavaScript if needed. This is because these two things are known to potentially let in malware or leave vulnerabilities in the system. Each web host has their own set of security guidelines that they follow to ensure that all your data is protected from threats and attacks. The code for the application is also built using password and role protection to further prevent any vulnerabilities in the system and application.