

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 | 03/17/22 | Jordan Ballard | Added Executive Summary, Design Constraints, and Domain Model. |

**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 has a game available in an Android application and they would like to create a web-based version to serve multiple platforms. The game is called Draw It or Lose It, and to accomplish the goals we will complete the software design document and begin development on the application. There are specific software requirements that must be met: the ability to have one or more teams, multiple players assigned to each team, unique game and team names, and only one instance of the game can exist in memory at any time. In order to proceed with this process, we will need to use common software design patterns.

## [Design Constraints](#_2et92p0)

* Currently only available as an Android application, so the application will need to be optimized for the web.
* The images must be stored together and easily accessible.
* The images must be rendered quickly.
* Security measures need to be taken.
* The program must meet hardware requirements.
* The game must be supported by multiple platforms.

## [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)

As seen by the diagram, these classes all relate to one another. Through inheritance, the Game, Team, and Player classes all extend from the Entity class. This is also an example of abstraction because Entity, the superclass, defines the methods while the subclasses use these methods, thus hiding implementation details. The GameService class also helps to hide implementation details by being responsible for getting and incrementing the ID numbers. This class ensures only one instance of the game exists at a time. There are several examples of encapsulation through the use of private variables and lists which helps protect the data. The program uses method overloading in these subclasses, and each subclass is an instance of themselves and of the parent class, so polymorphism is used as well. Everything is self-contained and organized into its own class, making portability easy. The “0…” next to the classes represents multiplicity, and shows how many instances of the object there can be. In this case, it means zero-to-many.

**"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** | Not as popular an option as Linux or Windows. Apple software is proprietary. Higher costs for equipment and licensing fees. Runs on Apache servers, is UNIX-based. Requires more updates over time. Good support. Secure. Standard operation is GUI. Ease of setup. | Linux is usually the preferred OS for web hosting. Runs on Apache servers. It is free, UNIX-based, open source, and secure. No licensing fees. Standard operation is command line. Support is harder to find. Greater flexibility and control, considered stable. Not always compatible with softwares, can be a challenge to install them. Can run for long periods without reboot. | Microsoft software is proprietary.  Certain applications require Windows server. Higher costs due to licensing fees. Standard operation is GUI and is familiar. Good support and compatible with a lot of software. Requires more updates over time. Required for certain apps. | In theory it may be possible but not practical and would be very limited. This would be considered not suitable for The Gamin Room. Device needs to always be on and this would be very insecure. No static IP. Low processing power and bandwidth. |
| **Client Side** | As a server, medium expertise and time required. Higher costs. To support Mac as a client requires the same as Linux and Windows since app is web-based. Server must send info as JSON for system to parse. No extra cost, expertise, or time to include Mac with others this way. | As a server, requires higher expertise and time, minimal cost. To support Linux as a client requires the same as Mac and Windows since app is web-based. Server must send info as JSON for system to parse. No extra cost, expertise, or time to include Linux with others this way. | As a server, minimal expertise and time. Higher costs. To support Windows as a client requires the same as Mac and  Linux since app is web-based. Server must send info as JSON for system to parse. No extra cost, expertise, or time to include  Windows with others this way. | Depending on the device there may be costs to place in an app store or something similar. More time and expertise if app needs to be cross-platform for all mobiles because of code language differences. Extra cost, expertise, and time may be required if separate framework is used to make app cross-platform for mobile. |
| **Development Tools** | Can support several languages or IDEs but common are Swift, Objective C, VS Code, Xcode, Java, Eclipse, Python, PyCharm, JavaScript, WordPress. There should not be licensing costs here unless tool is not free or a paid tier is chosen over free. Development team needs to be familiar with the chosen language/IDE. | Can support several languages or IDEs but common are C, Notepad ++, Java, Eclipse, Visual Studio, Python, PyCharm, Perl, Ruby, PHP, JavaScript, WordPress. There should not be licensing costs here unless tool is not free or a paid tier is chosen over free. Development team needs to be familiar with the chosen language/IDE. | Can support several languages or IDEs but common are Visual Studio, C++, C#, Visual Basic, Notepad ++, Python, PyCharm, Java, Eclipse, ASP.NET, JavaScript, WordPress. There should not be licensing costs here unless tool is not free or a paid tier is chosen over free. Development team needs to be familiar with the chosen language/IDE. | Although it is technically possible, this platform is unsuitable for development as it would be too difficult to write a program on a mobile device. For deployment *onto* a mobile device, Java and Swift are common. |

## 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**:

The recommended operating platform is Windows. The main reason for this decision is so that it is easier for the Gaming Room to maintain and control their server. Mac as a server is costly for equipment and works best when configuring and monitoring other Apple devices. Windows as a server has a recognizable GUI to be used as opposed to using the command line to operate. The ease and familiarity of Windows will eliminate any time required for admins to acquaint themselves to Linux and to operating through the command line. Windows also supports many applications, both made by Windows and by third-parties, and has more support in terms of troubleshooting and maintenance. This is a good option for a first-time server.

1. **Operating Systems Architectures**:

The different operating system versions of Windows are of the Windows NT operating system family. They use a hybrid kernel. The architecture is modular and consists of a kernel mode layer and a user mode layer. User mode sends I/O requests to the kernel and runs applications written for different types of operating systems. User mode subsystems do not have direct access to the hardware. Kernel mode controls things like memory management, hardware interactions, and scheduling, and has unrestricted access to hardware and resources. Windows versions since Windows XP support both a 32-bit model and a 64-bit model. 64-bit is recommended for the server.

The recommended operating platform pattern is the client-server model using the REST software architectural style. With the client-server model, the server takes requests from the clients and provides services, and following the principles of a RESTful architecture will help to simplify the system and increase interaction visibility.

1. **Storage Management**:

It isn’t necessary to use cloud storage for this application, or to use a third-party storage management software, so the data can be kept on the server and managed with the tools that the computer comes with, like File Explorer. A server using an SSD would be best because of the increased speed, though the cost is higher. Windows 10 comes with features like Disk Management and Storage Sense that help manage files and perform more advanced storage tasks.

1. **Memory Management**:

The server will need enough RAM to ensure the game runs smoothly, at least 16 GB. For the handling of the images in memory, all images can be cached in memory to be shared across each instance of the game. This way, they are available and ready whenever they are needed. Garbage collection is run by the operating system automatically and helps take care of memory leaks, and calls in the code can be included to prompt garbage collection to run at specific times. The code should be checked for memory leaks that will not be caught by garbage collection. If needed, the application can be directed to use memory in ways such as virtual memory, memory-mapped files, heaps, and thread-local storage.

1. **Distributed Systems and Networks**:

A RESTful API pattern will be used for client-server communication, including using HTTP verbs (GET, PUT, POST, DELETE) within the API. The client can connect to the server through an internet connection and receive data which can then be used and manipulated. The client will use their browser to make HTTP requests to the server. To accomplish communicating to different platforms, the server will return requests by using JSON to facilitate the data exchange between different programming languages. JSON is language independent and can easily parsed and generated by most languages, so different types of clients will be able to receive and use the data. Outages and connectivity issues can be combated by making sure there is a backup power source and that server can handle high-volume traffic.

1. **Security**:

Windows 10 has several security features built in. It has periodic software updates from Microsoft that fix bugs or bring new features. It also comes with its own antivirus program called Microsoft Defender Antivirus, as well as Microsoft Defender Firewall. Microsoft Edge has tracking prevention, a password generator, and a password monitor. And the system also of course allows different user accounts to be created that contain data that can’t be accessed from another profile. Windows 10 also has other useful security features such as secure boot, credential guard, SmartScreen, and BitLocker, which can encrypt the entire drive. The server can also provide security through the application with the use of principal object, authenticator, authorizer, and annotation, all of which work together to provide security, validate users, and allow/disallow users to perform certain actions. @Auth and @RolesAllowed annotations can be added to trigger authentication in specific areas and only allow specific roles. This authentication is very basic, so down the line another more powerful version can be used as a replacement.