

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_heading=h.gjdgxs)1

[**Table of Contents**](#_heading=h.30j0zll)2

[**Document Revision History**](#_heading=h.1fob9te)2

[**Executive Summary**](#_heading=h.3znysh7)3

[**Design Constraints**](#_heading=h.2et92p0)3

[**System Architecture View**](#_heading=h.tyjcwt)3

[**Domain Model**](#_heading=h.3dy6vkm)3

[**Evaluation**](#_heading=h.4d34og8)3

[**Recommendations**](#_heading=h.17dp8vu)5

## [Document Revision History](#_heading=h.1fob9te)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/12/21 | Michael Martin | Software design for requested application |

**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](#_heading=h.3znysh7)

The Gaming Room has requested that a web-based version of their Android game Draw It or Lose it be created. They would like the game to allow for multiple instances with unique ID’s and Names. Each game would consist of multiple Teams, with each Team having a unique name and ID

## [Design Constraints](#_heading=h.2et92p0)

Designing a game for the web would use a different software development environment and programming language than for an Android or iOS application.

The game must work across three different platforms

The game must check game and team name and ID across all platforms

Must have dev that knows the required languages(front end – CSS, HTML, Javascript; Back end - Java)

## [System Architecture View](#_heading=h.tyjcwt)

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](#_heading=h.3dy6vkm)

To start, ProgramDriver contains the main method of the java program. The ProgramDriver class then uses Direct Association with the class SingletonTester. This is to check if a game instance (GameService class) is already running. From here, we have the Entity class. This class is the parent of Game, Team, and Player class. Those classes inherit Entity’s variables and methods. The program runs by having GameService call Game, which in return calls Team, which then calls Player. Each of these instances must be unique, and will not be created if the desired name and ID already exist.

"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](#_heading=h.4d34og8)

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 does not offer a server anymore. The one currently being advertised is a device management for Apple devices. | Linux is the most commonly used operating system for servers. It is completely free and relatively easy to set up. | Windows Server is the second most used server operating system and runs out of the box. Setting any service is extremely easy and time efficient. This one costs the most, and usually has a yearly fee. | Mobile devices are not made to run server applications. However, while it should not be used for a live environment, a phone can run a web server. |
| **Client Side** | Macs offer a great development software environment, such as xcode. However, to develop for Mac, you must use a Mac. | Developing for Linux is easier than Windows and Mac. Linux can run most compiled code, if it doesn’t you can install a package to resolve the issue. Linux usually comes with Python, C++, C and more out of the box. | Developing for Windows is the most difficult of the three. Windows can use applications written in .Net, C#, and C++ natively. | When developing for a mobile device, you typically have to use the programming language recommended by the OS. You’ll want to find a developer that specializes in mobile development, as the guidelines are different from desktop applications and web apps. |
| **Development Tools** | MacOS has many development tools, one specific to Mac is xcode. Other IDEs (not usually used for Mac application dev) include Eclipse, NetBeams, Atom, VSCode, etc | With Linux you can develop for most programming languages. A wide spread of IDE’s exist, such as Eclipse, NetBeams, Atom, VSCode, etc | The most common IDE for Windows would be Visual Studio Code, which is developed by Microsoft. As stated above, native applications can be written in C#, C++, and .Net. You can install additional run times such as Java and Python. | The two major mobile platforms are Android and iOS. When developing for Android you’ll want to use Android Studio, and to know Java or Kotlin. Xcode is the IDE for iOS applications, and the language is called Swift. |

## 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**: As someone who uses Linux, I recommend Linux as the operating platform. Specifically the variation called Ubuntu Server. Ubuntu can provide quick setup, as well as an LTS (long term support) option.
2. **Operating Systems Architectures**: I recommend using Docker running within a dynamically adjustable virtual machine, most applications will provide a docker container with the specific application almost ready to use.
3. **Storage Management**: With this, if referring to the physical drive, an m.2 or 2 ½ “ SSD would be recommended. This would remove any physical bottlenecks that may be caused by a harddrive. As for docker, each docker instance creates a container that manages the storage for you. Docker can dynamically adjust the amount of storage a container uses to ensure the container does not run out of storage.
4. **Memory Management**: Docker works directly with the Linux Kernel to ensure that Docker itself will not end up using too much resources. Docker also provides a way to limit the amount of resources a Docker container uses. When looking at the virtual machine, depending on the provider, the virtual machine can adjust the amount of RAM dynamically to ensure you do not over spend on resources and do not run out of resources.
5. **Distributed Systems and Networks**: Docker allows for other containers to start automatically if one is to crash. This would create redundancy, preventing downtime of the web server. Docker also allows for automatic network redirection, where if a container crashes, the network is automatically routed to the new container.
6. **Security**: As with most operating systems, Linux uses a role based platform. This means that each individual user is assigned to a different group, that group is then assigned a role, which is assigned specific access. Such as the role “Root '', “Sudoers”, or “Wheel. Each of these have different access to various parts of the system. The Linux Kernel is natively secure, being open sourced, individuals and companies are always looking for vulnerabilities, and patches are pushed to the stable branch every 2-3 months.