

# 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 | 08/15/2022 | Michael Thero | evaluation. |

**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 client wants to transfer their game from an android app to a web-based app so that it can be played across platforms including Mac, Linux, Windows and on mobile devices. Each game will need to have the ability to have one or more teams involved. Each team will have 1 or more players. Each game and team name must be unique. Only one instance of the game can exist at a time. Each game, team and player must have unique identifiers.

## [Design Constraints](#_2et92p0)

Will need to be designed for many different screen sizes, varying input devices i.e., mouse, keyboard, touchscreen. Will need to be able to run on limited memory and compute power to allow for mobile devices.

## [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)

Below the UML class diagram provides a quick glance at the intricacies of the design of the program. It shows the different classes and the information that they will store and provide. Each entity will contain all other classes. The GameService class will track all games. Each Game will contain teams and each team will contain players. Setting up the classes this way allows for abstraction of the data.it compartmentalizes the components so they can be moved or used as need arises.

**"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 a server OS but it has a license fee and most cloud services don’t support having a mac OS server. | Linux is free and it has a Server version that are low utilization because they do not have a UI in them just a command line. | Windows has a server version as well but there is a license fee. Most cloud providers offer support for windows servers. | Mobile devices do not make good servers. All though I have run servers on raspberry pi’s which are mobile to a point. |
| **Client Side** | Mac has safari as their default web browser. It is cable of running JavaScript, HTML 5 and CSS3 natively. There are a few little quirks in safari that make some JavaScript act funky I have noticed. | Linux has fire fox as its native web browser. Fire fox up until the lates release did not support webp files which would have made displaying the images for this app slower. | Windows IE was the worst when it came to supporting JavaScript. You could not use foreach loops which I found out the hard way. It also did not support webp images. But the news edge browser is as good as chrome. Because it is a chromium browser. | Most browsers on modern mobile devices are either safari or chromium browsers. They are good but mobile internet can cause them to be slow and cause loading error. |
| **Development Tools** | Mac has an array of choices when it comes to development tools the most common one is Atom I think. | Most development tools that are on mac and windows is also made for Linux because Linux is the most common server in use. | Microsoft visual studios has a whole sweat of tools for developing in any language and is my favorite for development. | Mobile devices do not have many development tools and are hard to develop on because they are so small. |

## 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 Operating Platform that I would recommend is the Linux operating platform. I would recommend a Debian version like ubuntu 18. Linux is an open-source operating system that most production servers run. Because it is open source there is no license fee to use it so there are cost savings there. It comes in a lightweight server version that can be run on relatively cheap servers because it does not require a lot of hard drive or memory space to run the operating system.

1. **Operating Systems Architectures**:

There are many versions of the Linux including ubuntu, centos, fedora, RedHat and many others. The biggest difference between them is that Debian distributions tend to use older stable kernels where fedora uses the most up to date kernels. The reason I recommend Debian versions of Lenox is that they use more stable kernels that are tested and proven over time. They have had most of the bugs worked out and are easier to maintain.

1. **Storage Management**:

I recommend using the default file storge management that comes with ubuntu for the server itself. But I would recommend using wasabi to manage and store the images for the game. Wasabi provides hot storage and a great price with no egress fees. This will allow the server to remain small and cheap will allowing expansion of the game by adding more images and levels.

1. **Memory Management**: For memory management for the client side of the application. I would recommend lazy loading images and using WEBP images. This will allow the smaller mobile devices to access the game faster and provide a better user experience. WEBP is supported by all modern browsers and is a lossless compression algorithm that allows for up to 80% if file size savings. For the server-side memory Management I would recommend using a varnish cache server in front of the web server. The cache server will load the most requested images, HTML, and CSS files in memory allowing them to be served quickly while being able to use a smaller memory bank on the webserver.
2. **Distributed Systems and Networks**: distributed systems would include hosting the web servers on google cloud. They have a robust platform that is about 10x cheaper than amazon. This will also come with built in firewalls that will protect your servers from the outside world. You will get faster network speed and lower cost by hosting in the cloud as well. This will reduce latency and give better page load times for the user. As mentioned earlier I would also recommend hosting the images on wasabi for faster response times and cheaper easy to expand storage.
3. **Security**: As I have stated previously hosting on google cloud will come with built in security for the servers themselves. They have a robust firewall that your servers will sit behind allowing only the http and https ports open to the outside world. I would also recommend using https over http so that all messages from the server to the client are encrypted and protected against man in the middle attacks as well as packet sniffing. You can force https on all request by redirecting http traffic for port 80 to https routed over port 443.