

Gaming room game service

# **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 | <1/24/2021> | <Ali Choudhry> | <Brief description of changes in this revision> |
| 2.0 | <2/6/2021> | <Ali Choudhry> |  |

**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, Gaming room, wants to run a web based game that can run on multiple different operating platforms. The game is called Draw it or lose it and is currently only available on the android store. In order to expand the program must be compatible with different operating systems such as mac and linux. In order to get the device running on mac the program would have to be written in the swift programming language as apples it is compatible with apples hardware. And in order to run on windows devices it should be written in C++.

The game is supposed to run on the web as a web based application, I made a mistake by assuming the application itself was to be created and be compatible with individual operating systems, i.e macOS, windows etc.

I did read through the information that was provided to us but did not interpret it correctly.

## [Design Constraints](#_2et92p0)

1.The game currently only runs on android devices but should be able to run on all sorts of devices with different operating systems.

2.It needs to be written in languages compatible with the end users hardware.

3.Only one instance of the game service class should exist in memory.

4.Within this class is a list of current games being played, no two games can have the same name so before adding a new game to the list an iterator will check to make sure that there does not exist a game with the same name as the one we are trying to add

5.Likewise no two teams within a game can have the same name

6.No team can have players with the same name

## 

## [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)

First we have the entity class which as its name suggest represents an “entity”. This class has two variables. A long labeled id and a string labeled name. This class is also encapsulated and has accessor method that returns these values as well as overloaded constructors. One which takes in no values whilst the other takes in both a long and a string. The entity class also has a to string method to display the id and name. This entity class is inherited by the player, team and game class. All these child classes use the constructor provided to them by their parent class.

The game class owns a team array list. Every instance of the class has this list. The team class has a list of players. The team, game and player class are all encapsulated and have their own getter methods.

Finally we have the game service class. This class uses the singleton design pattern, because of this there will only be one instance of it. Within this class is a static method called getInstance along with a private constructor so the user cannot create new instances of this class. The getInstance method checks to make sure a static variable owned by the class is not null. If it is null a gameinstance is created. If not then this same instance is returned so no new gameservice objects are made.

The game, team and game service class use iterators to iterate over their static lists to make sure all rules are followed. I.e no two games can have the same name, no two teams can have the same name and no two players within a team can have the same name.

****

## [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** | Pros  \*Can be good for hosting a network of Macs  \*Quality hardware  \*Can support java if JVM is installed  Cons  \*Mac OS runs on Apple hardware which is often expensive  \*Hardware on the server side doesn’t scale as much as other operating systems | Pros  \*Linux operating system is open source and free  \*Rarely has security issues. Since it is open source this allows individuals to contribute to it as well and find and patch any issues within the code.  \*Can support some older hardware if the company wants to keep costs down and invest into older/cheaper hardware  \*Does not crash often  \*Supports multitasking  Cons  \*Although it has many loyal users it is not as popular as lets say windows so new developers will have to get accustomed to working with it.  \*Not as user friendly as some other O.s due to heavy use of terminal | Pros  \*Also popular so finding developers to work with it should not be to hard  \*Many business’s  use the windows operating system  \*Supports many third party applications  \*Software updates are easy to install  Cons  \*Licensing fees can rack up causing costs to increase | Pros  \*hand held and mobile, can fit almost anywhere  Cons  \*Would need JVM installed onto the device  \*Limited memory/storage  \*Not really practical |
| **Client Side** | Pros  \*Ease of use for users, navigating the GUI  \*Elegant set up  \*Browser is easy to find, almost always available  \*The web browser can scale quite easily  \*Can download a popular and reliable web browsing application like google chrome and run the web application through it  Cons  \* Although apple hardware is usually expensive the brand is very popular | Pros  \*Sine Linux is open source this allows anyone to download, set it up and use the operating system.  \*Can download different web browser applications  \*Can scale the project  Cons  \*Heavy use of the terminal which can be intimidating to users who are not aware of how to properly use it  \*Not many development tools | Pros  \*Also very popular and easy to use  \*Purchasing a device that uses the windows operating system is quite easy due to the amount of devices that use the O.S  \*Navigating around the GUI is simple  \*Can download multiple web browser applications such as Firefox, google chrome, Bing etc.  \*Frequent updates | Pros  \*Ease of use  \*Accessible to most people with mobile devices  \*Mobile and ready to use  \*It is possible to download different web browsing applications  Cons  \* Sometimes you may not be able to connect to the web if the area you are in does not have internet/service |
| **Development Tools** | Running and developing in java will not be an issue as long as JVM is installed but for MacOS specific applications there is a little challenge.  \*To create applications on Mac OS you will need to use Swift and use the Xcode ide. One can also use the Atom Ide as well as the AppCode ide  Con  \*If you want to create MacOS applications you will need a membership through apple which costs money  \*Will need apples approval to run an app on its store | Pros  Languages for developing Linux applications include C/C++ java, python etc.  \*C++ and java work well with eclipse and NetBeans. Pycharm is great for python | Pros  \*Microsoft has documentation on how to create windows applications using C++ and the visual studio code.NET ide  Cons  \*Can be free to use visual studio code but if you want extended features and other pros it will cost money | There are many tools to help develop mobile applications  \*If one wants to create android applications then they can use android studio and develop using kotlin or java it is also free  \*Creating apps for apple devices will cost some money($99 a year)  \*Can also create both apple and android apps using flutter programing language which is open source and free. You can the xcode or android studios ide |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>

* When it came time to decide on an appropriate web application server, we narrowed the choices down to Windows and Linux and ultimately decided to go with Linux. This is because Linux is open source and free to use. In order to use the Windows operating platform we will have licensing fees.
* Linux allows us to implement the client server architecture that we desire so we can get our application running on other computing environments/operating systems.
* We will use Ubuntu which is a Linux kernel-based distribution. Next, we will install the Apache web server on Ubuntu.
* Linux is known to be reliable it is a free, open-source platform will provide us with what we need.

1. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>

* The underlying architecture we will be using is the client server architecture. The client will send requests to our web server and we will provide the proper responses.
* Every computing system requires hardware. We need a way to manage this hardware and have it work to get things done. This is where the operating system comes in.
* The hardware will consist of the CPU, RAM, secondary storage, input/output devices as well as network network interface card.
* On top of this we will have the operating system which includes the Linux kernel, system libraries and kernel modules.
* And finally we will have the user programs or applications running atop the operating system.
* These hardware devices will be managed with the operating system. For example lets say that a certain program is hogging up to much CPU time. A system call will be initiated and the registers, program counter, memory locations will be saved onto a process control block. The process control block is an abstract data structure which represents a process. All of this data will be saved into the operating systems stack memory and a new program will be run. This is called a context switch and is one way a operating system switches processes.
* Overall the operating systems role within a computing device is vital. It provides so many important features that we need.

1. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>

* For storing user information such as username, password we can use a linux database sever. So when a user tries to log in this sever will process their request.
* All of this data will be stored in secondary memory. Once a new user is created their information will reside on disk this way if a power outage was to happen and all of our severs are down we can be assured that the user information will be safe.
* We can use the oracle database on this server.

1. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>

* Linux uses virtual memory. In other words there will be “virtual addresses” that map to real locations within memory. This is done to avoid memory fragmentation. Lets suppose we did not use virtual memory and instead mapped all processes straight to real memory locations. This will work fine for a bit, as long as RAM is not overloaded. But suppose we have 4 programs running. Process’s A to D, they are all arranged in order of high-low memory consumption. They can also be thought of as being right on top of each other. Lets say process B is done, now we have space to run a different process lets call it E. Process E is half the size of process B and is now sitting in the same spot in memory where process b was just running in. But since it is not using the same amount of memory there will be some leftover bytes of RAM that are not being used. Do this swap a few more times with processes of different memory needs and we will end up with memory fragmentation.
* This is not optimal. So Linux uses virtual memory to fill these holes. Also if the ram is being overloaded a process can be loaded into secondary memory for the time being to free up some space in RAM.
* Specifically for the game room we will keep our images in secondary storage and the game itself will be running in ram. The data that makes up the images will be read by the disk and transferred over to ram where it can be processed and sent to the user. There are other ways of handling memory management which are better in terms of keeping memory usage low but we will stick with this routine.

1. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>

* For the gaming room we will use the client server architecture pattern. The client will make a request using a web browser and it will be the severs job, which is running our java code, to respond and provide the proper resources to our user.
* This is all done using the network, which provides a way for our users to communicate and receive information from our severs.
* In order to handle outages I believe we should have some backup servers running near our original severs. This will cost extra but they provide a backup plan in case of an emergency.

1. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>

* On our platform we must ensure security and keep ourselves and our clients safe. Ubuntu, the Linux kernel-based distribution, provides this for us by giving user programs to have limited/low privileges. This is done by using the principle of least privilege. Which means that the user program can only access information that it needs to perform its routine, or other information related to the user such as their name, team name etc. It cannot access anything outside of this domain
* This way to user program can attempt to alter kernel code or go into the kernel. The underlying operating system ensures strict rules for user programs, so they do not access the kernel mode.
* On top of this ubuntu also has a built-in firewall that we can configure. This ensures that certain, most likely malicious IP addresses cannot enter our system. On top of this already existing firewall, we will add an extra network-based firewall to provide maximum safety to us and our user. With this set up our network is sure to be secure
* Since this platform is open source it allows for other users to quickly find any bugs if they exist and report them.
* To protect our users from getting their accounts stolen we ask them to put in a proper password when they sign up. Specifically, one that has a combination of lower and uppercase letters as well as numbers and special characters. This way if someone were attempting to get into their account from using username it would be nearly impossible due to the complexity of the password.