

The Gaming Room

# **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 | <01/18/2021> | Shelayah Robinson | Adding information related to software design |

**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 decided to develop a web-based game. At the moment the client’s game is only available on an android application. The application they are wanted to make is currently called: Draw it, or Lose It. The client is hoping to serve all platforms and not just android. The idea for the game was based off of a 1980s television game: Win, Lose, or Draw.

## [Design Constraints](#_2et92p0)

For the Gaming Room: the design constraints are as follows:

* The client requires the app to be web-based so that more people are able to access it
* The game itself would consist of four rounds, one minute each
* The game concept wants to revolve around the team members to guess part of the puzzle such as a phrase, title, etc.
* Drawings must be rendered at a steady rate and complete at the 30-second mark.
* If the team member doesn’t guess the answer before the time ends, then the remaining teams have 1 guess each to solve the puzzle with a 15 second time limit.
* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [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)

The Entity class is essentially our handler for the following: Game, Team, and Player class. From the three they inherit characteristics of the Entity class. Of the four classes created: Game, Team, Player, and GameService they all reference each other. The ProgramDriver class is how we create or build the project to meet the requirements of our clients. From the ProgramDriver class we can access all the classes we created, and execute them. With that in mind, there's also the SingletonTester class. For each class that is organized, it allows the project to run according to the design constraints: it will allow for there to be more than one teams and players, while having one game session at the same time.

**s**

## [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** | According to the Apple website, in order to purchase a server and run things for IOS, it can be purchased at the rather cheap price of $20 dollars. This inexpensive server will be accessible for creating websites. Their website also includes macOS Server resources to help you get started with using the server. However, there aren't that many people that use Apple as their main OS. | The Linux license is free; however, the price solely depends on whether what the requirements and what needs to be run on the server. So, this requires a bit of research. The only one downside is that not many people are too familiar with how to utilize the Linux interface. | Unlike Linux, you have to pay for Windows license. In order to run a Windows server, it solely depends on the server you are wanting to use. According to the Standard Edition, it is only 20 per month and $972 dollars own it for use. According to the Datacenter Edition it is $125/ month and $6,155 to own for use. This is perhaps the most expensive to deal with of the three. However, Windows is the most commonly used OS. | Mobile Application Server costs between 70$ to 320$ per month. This depends on the how much CPU, memory and disk space. It's fairly inexpensive, but it depends on the project you are looking to run. But if you have good traffic on the app it is then it can easily cover the costs. And a great majority of everyone use mobile to access data. |
| **Client Side** | On the client side, since apple is not an open source, costs would be rather low. There is a 29.2% of the population that uses macOS as their primary system. However, this solely depends on the previous experience of the user; however, some maybe prone to learning faster than others. With that being said, this wouldn't take as long learning as Linux. | Unlike Windows and Apple, Linux is an open source, which allows costs to be low. However, this would also require the client to be educated or have experience in using this OS. Not many are too familiar with how to use the OS because it isn't commonly used as much. Only 25.3% utilize Linux as their operating system. | Costs would be high, and like apple, it isn't an open source. However, because many are quite familiar with using Windows, it wouldn't take too long to learn. About 45.3% of the population use Windows. | The cost would again, be low because mobile is fairly flexible. A good majority of population utilizes mobile devices so there wouldn't be a need for worrying about clients having past experience. However, there is taking into account the different operating systems that can be run on these devices. And while these devices can be split into two categories of Android and Apple, there are other subcategories to consider for primarily Android. |
| **Development Tools** | When working with Mac, they have their own programming language that is compatible which is Swift. Swift allows to work with Apple products when developing applications etc. | When developing for Linux, the development tools that can be best utilized are Visual Studio. While Java is the main language, there is also programming languages such as C+. | When dealing with Windows, the Development tools are commonly Visual Studio which can deal with a lot such as web applications, applications, etc. | The development tools for Mobile devices solely depends on the target audience. If the target audience is just Apple, then it would be appropriate to use just the native language which is Swift. If the target user is just Android, then there are a few languages that can be used to, such as Java, Python, C#, etc. But the main programming language is Java, and it also requires to download the Android Native Development Kit and also getting access to an IDE. If both clients targeted, then both Java and Swift would have to be utilized. |

## 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**

In order to expand the game application, I would recommend utilizing the operating platform, Windows OS. Of the three operating platforms, Windows is the most use, it is currently 45.3% of the population, uses this OS, and since the application is web-based it would be better to utilized windows for the back side of things. Such as handling user requests, troubleshooting, etc. While there are license costs to consider, the platform is a little more flexible.

1. **Operating Systems Architectures**: The Windows OS has a GUI that is rather simplistic to allow for easy navigation. Windows also allows for a wide range of IDE’s that can be used to develop applications. Some of the most commonly used IDE’s include: Visual Studio, Eclipse, IntelliJ, JetBeans, as well as a few others.
2. **Storage Management**:

The windows OS manages storage through simple configuration which allows for those working on the back-end side of things to easily navigate through the settings. While owning a windows cloud server does require another expensive investment, however this is a good investment because this space can be expanded if the client ever has future plans for expanding the game application and adding different assets, etc.

1. **Memory Management**:   
   For Windows OS, there are two options for windows to select from physical and virtual memory. Virtual memory is more suitable because it allows for large programs can be handled effectively. One of the main advantages to owning virtual memory as opposed to physical, it has memory protection, and extended use for physical memory.
2. **Distributed Systems and Networks**: When working with windows OS there are going to be a bit of issues that are going possibly going to coming across. The common issues, includes lagging, recurring queuing problems, and having overloaded servers that may cause simple tasks to in mind, I think these problems can be resolved through communication between the developer team members.
3. **Security**

There are plenty safety features that are set into place. This includes the Windows Defender antivirus program that always runs on the client's computer. This also includes VPN services in the list to add protection to the user information. With this in mind, it's important to implement daily routine security checks to ensure that the users are safe and none of their information is exposed by unauthorized access. This also includes making sure that developers working on the application are well educated in learning how to secure and encrypt user information.