

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 | 07-12-2021  07-15-2021 | Carlie White | Executive Summary, Design Constraints, Domain Model |
| 1.1 | 07-31-2021 | Carlie White | Evaluation |
| 1.2 | 08-15-2021 | Carlie White | Final Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The client wishes to create a game where teams will compete to guess the puzzle given certain stock drawings as clues. The game will consist of 4 rounds that each last one minute, and there may only be one instance of the game running at once, although there must be multiple teams made up of multiple players and team names may not repeat.

## [Design Constraints](#_2et92p0)

The client wishes to develop a web-based version of the gaming app so that they can serve multiple platforms based on their current game that is only available for Android devicesz.

## [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)

All classes represented are in the package “com.gamingroom”. The parent class, “Entity”, inherits to the child classes “Game”, “Team”, and “Player”. By allowing the child classes to inherit from the parent class, polymorphism is used to keep the unique identity of each class. The multiplicities between the child classes allow for the “GameService” class to have multiple teams and multiple players. “GameService” holds the bulk of the information and uses the singleton pattern to ensure there is only one instance of the game running. Finally, the game is implemented in the class “ProgramDriver” which allows for a level of abstraction while also using the “SingletonTester” to ensure the singleton pattern in “GameService” is running properly. All the instance variables are encapsulated throughout the program which also adds another level of security.

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac webhosting is the most secure out of the operating system options listed due to the safety nets Apple puts in place to protect their servers. The terminal allows for easy access to manipulate and control the server as well. | Linux is the second most secure operating system because it rarely ever gets updated which makes it so that the security measures you put in place won’t get compromised due to an update. | Windows servers have the most clients which can allow for a higher risk of security threats. Windows also tends to be slower or choppy with programs not made specifically for them, but Windows also has application databases to choose from. | You could potentially use a mobile device as a webserver, although it is not ideal since they are pretty “low-powered”. The device would always have to be powered on and connected to the internet in order to keep the server up and could not be used for any other purposes. |
| **Client Side** | Although Mac is the most expensive, the hardware is top-of-the-line and is well known for being very user-friendly on the client side. Furthermore, Apple has created an environment that is well-protected against attacks so little time and expertise will have to be spent developing. | Linux is an open-source operating system which would make it one of the cheaper options and it can also run on almost any hardware. The performance of Linux isn’t as great as Mac, but it is the cheaper option. Expertise would be needed in order to create a “safe” environment. | The cost of using Windows can range from very high to very low. Windows is fairly easy to develop on which would require minimal expertise. Windows will run slower with certain cross-platform software, which could increase the amount of time spent. | It would be very time consuming to create the program on a mobile device, although it might be one of the cheaper options. It probably wouldn’t be able to handle all the tasks requested of it because of how low-powered it is. |
| **Development Tools** | HTML, CSS, Java, JavaScript, Python, Ruby, WordPress, vBulletin, cPanel, VisualStudioCode, Eclipse, PyCharm, BlueJ. Webhosting from Mac is often more expensive. | HTML, CSS, Java, JavaScript, Python, RubY VisualStudio, Eclipse, PyCharm, BlueJ, WordPress, forum software. Webhosting for Linux is cheap. | HTML, CSS, Java, JavaScript, Python, Ruby, ASP, .NET, Microsoft Access, MSSQL, VisualStudio, GoDaddy. Microsoft offers webhosting. | HTML, CSS, Java, JavaScript, Python, Ruby, AIDE, Xcode, eclipse. Webhosting from mobile devices is limited but is usually cheap due to low feature options. |

## 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**: I believe that a cloud-based operating platform server best suit the needs of The Gaming Room. This is because the cloud-based server offers better security, scalability, convenience, and can be a lot cheaper. The cloud server I would choose is Microsoft Azure.
2. **Operating Systems Architectures**: depends on virtualization, which is essentially creating computer hardware in a software setting. The virtual system is then able to execute processes using physical servers that cloud providers maintain. Since there is no physical hardware on the customers end, the cloud providers upgrade and maintain the physical hardware as needed, taking away a lot of issues for the user.
3. **Storage Management**: In order to store over 200 images that the user needs to see, I believe that it is necessary to store the data on a virtual hard drive, which is also known as a cloud-based storage. This allows for the app itself to be much smaller than if the images were to be downloaded with the app itself. It also ensures that the developers can change the images at any time they would like without having the end-user even notice.
4. **Memory Management**: Using Microsoft Azure, memory can easily be managed with a virtual memory manager, also known as a VMM. The virtual memory systems allow for larger programs to run on computers with various amounts of physical memory. The memory management system works dynamically with what it is offered. If there is more free memory, then the management may allocate more memory. Likewise, if less memory is from, the server may release memory to the OS.
5. **Distributed Systems and Networks**: Using a cloud-based system for development and production, it is necessary that all parties are always connected to the internet. Then, using the client-server model, your single application can provide to any platform you wish. The server can host multiple clients at once, and you can also have multiple servers on the same virtual machine if needed. All the different servers are interconnected without a centralized database among the servers and can share information. Each platform having their own server allows for each server to cater to the needs of its client. Using REST API, which is independent of platform and language, the clients and servers can easily communicate cross platform.
6. **Security**: In order for this application to be secure, it is necessary to require authentication and have multiple servers (segmentation). All ‘important’ information should be on its own server, with limited access (access control). Furthermore, using an access matrix organizes each domain and each object and its relationships. This allows for the operating system to stop unauthorized access since it is aware of the domain rights with respect to each object. Users should only have access to what is needed to use the app and nothing more. If there is a suspected breach, shut down the virtual machine until you find the cause and solve the issue.