

Alexander Mock

<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 | <09/17/20> | <Alexander Mock> | <update the revision> |

**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 company would like to develop a web-based game that can run on multiple platforms. The game will be called “Draw It or Lose It”.

The purpose of this game is multiple teams consisting of several people going four rounds for a minute each. A picture will be pulled from a library of images. Drawings are rendered at a steady rate and are fully complete at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit.

## [Design Constraints](#_2et92p0)

Since we need this to run on multiple platforms we must find a way to write code in different languages or write the code and have it inherit use for other languages to be able to use.

Other constraints:

* Only one instance of a specific game can exist at once
* Needs to be able to run on multiple platforms
* Must have more than 1 team
* Each team must have more than 1 person
* Games/Team names must be unique

## [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)

Referencing the model below:

**Entity** holds a relationship between **Game**, **Team** and **Player** class. They all inherit information from **Entity**. With the UML diagram we can show this with “inheritance”. Each class shares common references like “name” and “id”. Looking into their relationship, we can see **Team** and **Player** also denoted through aggregation. You can also see **Game** has a Team and **GameService** has Games. When we using aggregation it is an instance of one class and has a reference to an instance to another class. Looking further at this diagram, we see **GameService** has a reference of Games, **Games** a reference of Team, and **Team** a reference of Player.

****

## [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**  **(Host)**  **In my opinion I would host the web app server on which ever OS you are most familiar with.** | Mac, can be known for a flexible terminal. | Linux, also known for its terminal uses | Windows, has lots of available software comparing to mac/linux. | <Mobile server. An interesting concept but most likely not recommended. Servers usually take a lot of resources and power I don’t think this would but sustainable if this web app grows exponentially> |
| **Client Side** | Expertise: Moderate  Cost: equivalent to windows  Time: is dependent on application development | Expertise: High  Cost: Usually less than Mac or Windows  Time: is dependent on application development | Expertise: Low - Moderate  Cost: equivalent to Mac  Time: is dependent on application development | Flexible, as Os’s are variable. Since os is not set in stone this can make it more difficult to implement |
| **Development Tools** | Tools: Swift/Notepad++ HTML/CSS/JavaScript/Python/Ruby/Java | Tools: Visual studio/eclipse/Notepad++/HTML/CSS/JavaScript/ Java/ Python/C++ | Tools: Visual studio/eclipse/Notepad++/HTML/CSS/JavaScript/ Java/ Python/c++ | Tools dependent on mobile os: Visual studio/eclipse/Notepad++/HTML/CSS/JavaScript/ Java/ Python/swift/c++ |

## 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 would recommend The Gaming Room to start on a Windows based platform. Most big businesses run windows environments. Generally speaking you should run whatever you are most familiar with. For the Operating server Platform Windows Server 2019 would be a good place to start. Starting with Windows Server 2019 this platform allows for both vertical and horizontal scaling as needed. As we expand our network horizontally we can upgrade the server vertically with better processors/ram and storage. Since most of the world uses Windows there are many options for development tools we can use. Setups are generally easier as they are wizard based.
2. **Operating Systems Architectures**: Windows is nice as they offer an easy to use GUI (graphical user interface). Most people who have a computer these days have learned on a windows based operating system. Since most people have this degree in understanding, Windows will provide an easy to use experience. On top of this Windows server 2019 is similar to your standard windows 10 but with all of the extra server features. Running windows environments will be easier to manage the network if all devices are on a similar OS. It can be hard to manage multiple OS’s when dealing with updates/issues or troubleshooting, especially if we horizontally scale our network in all. The more machines will require more support to setup and maintain. It is wise to start with what we need and based off of our projections/budget make the architecture scalable as needed. If we do go down the route of having a few outliers using other operating systems they will need to meet our system requirements.
3. **Storage Management**: Depending on if you are the client or server side, Windows 10 and windows server 2019 both include simple and easy to use storage structure. They both have the ability to use a management tool called storage sense. In server 2019 and windows 10 you have to initially enable it. This shows were your space is being taken up and allows for you to easily access and manage what files you’d like to keep clear. As our game progresses we are also able vertically scale our storage as needed.

When utilizing storage sense it has the ability to be configured as needed, such as when there is low disk space or after an allotted amount of time. Storage sense has the ability to delete temp files like when you run disk cleanup. If we are utilizing cloud storage as sometime it can be cheaper it depending on the configuration we can also setup storage sense to clean that up as well. If you don’t want to have it be configured to run at times you can simple run the utility as needed.

Another big topic is running backups/replica’s of our environment. These are two separate methods that hold different purposes. In our case It would be wise to run backups of our servers weekly. This is in case of a few different things. Such as if there were a bad update that messed up our environment we can roll back to a previous week. At this time we can test which patches were likely to cause the issue. I can also be a good idea for the main components to host offline backups. By using offline backups other users/hackers/people with mal-intent won’t be able to reach them on our network.

For Replica’s this more so for our last resort. Security is a big factor in anything technology related these days. By creating backups and restoring them to a secondary network if we were ever to be compromised we can boot up the secondary network and host everything as if nothing had happened.

1. **Memory Management**: In windows operating systems you can choose where you would like to keep your active files. Depending on your storage setup you could keep all of your files/application on a separate drive that can be encrypted. As our game progresses we will also be able to vertically scale our memory as needed.

In Windows server 2019 and Windows 10 you can adjust the virtual memory as needed. Under the advanced tab of “**Adjust the appearance and performance of Windows”** there is a section to set memory adjustments. Depending on what/how many drives we are allocating you can set it by drive as well. Comparing recommended values vs where it stands currently.

1. **Distributed Systems and Networks**: Knowing that the client would like Draw It or Lose It to communicate between various platforms, we need to find a common focal point to where everything can connect together. After some digging development apps such as Iconic will be able to help disperse our system/network.

How these development apps work are to have the app hosted on the server and end users side connects to. Once the application has been loaded and configured on the server side we can then push to the clients side.

By hosting most of the main functions on the server this will greatly reduce the processing power the client could have been responsible for. In multiplayer situations by hosting the processing on the servers environment this will allow for smooth even playing field for the clients that connect to it.

If we were to have the clients individually do the processing every machine would play different. As there are different machines with different specs this would be the determining factor at play for rendering/compiling/running.

1. **Security**: Security is a big factor. Most operating systems have some level of security but it can be considered low level. I would recommend going with third part security such as a physical fire wall (palo-alto/juniper) and or integrated anti-virus(Sophos intercept x). As most operating systems initial protection won’t stop most hackers these days.

By placing a physical firewall between where the internet connection comes in and our routers and switches this will be the first big layer of defense. With this it will allow to set parameters of what traffic is allowed in and out of our internal network. This includes things of certain IP addresses, devices, mac addresses, and websites.

After the first big physical layer we can also implement an intricate sophisticated antivirus program such as Sophos antivirus x. This works on a software level. By being placed on each domain device we will be able to monitor typical antivirus tools such a on access scanning/scanning/email phishing/malware/download protection. Along with the basics this also acts as a physical firewall. It can Block certain traffic access via Device/mac address/ip address. It also back logs all the info in case there is an incidence.

By keeping tight security this allows us to operate at optimum efficiency and not have the hassle/worry as much about external forces. As stated above we can also utilize our backups and if we have the budget can create a replica of our environment to push our backups to in case we are compromised. If in the event that this were to happen we can boot up the secondary environment and continue on after dealing with the threat(s).

**Server Side:** The client has asked you to create a web-based application. This implies a server-style configuration for hosting the website and allowing it to scale up to thousands of players. What does this mean for your ability to host the software application on each operating platform listed above?

-In order to keep up with the vast requirements there a few things that come to mind when dealing with such scaling.

First we would need the server itself to handle the processing load so when streaming it back to the devices there is less load bearing on the actual devices.

Along with this we need to be able to foot the traffic we are taking on. Specifically Network requirements. The more players the more bandwidth we will have to have in order to handle the traffic. If we were to overload this can cause network drops, higher ping, thus causing the game to not run smoothly.

In order to keep up there are methods known as Auto scaling(Horizontal and Vertical)

horizontal - increase or decrease our processing load such as with more servers or database servers. When we horizontally scale, we are scaling out.

Veritcal - Vertical scaling, also referred to as scaling up, involves adding increasing capacity with additional hardware. Examples include faster or more processors and adding memory capacity. Another example of vertical scaling would be to increase a database instance with larger RAM allocation.

* **Evaluate various platforms for their characteristics, advantages, and weaknesses for hosting a web-based software application.** Consider the following in your evaluation and articulate your findings in the software design template:
  + Does each of the operating platforms offer a server-based deployment method where the website will be hosted?
  + What are the potential licensing costs to the client, The Gaming Room, for the server operating system?

-Ideally we can host a server on all the above listed operating systems except for a mobile OS. I wouldn’t generally recommend even trying because of the resources needed to compensate. Mac, Linux, And Windows are all considered suitable to setup a server – based deployment method.

For Licensing, Mac and Windows would be more expensive but can be considered to be easier to setup. Most things revolved around Linux are free or open source and has a great niche community surrounding it for resource. Where Linux is free it can struggle with the ability to setup, sometimes seen as tedious if you aren’t used to the environment. For Mac and Windows you will have to purchase a server license to run the server along with what ever deployment application you decide to use. Linux you could setup the server and only have to purchase licensing for the deployment application. Where you save money with linux you could be fighting for setup, configuration, and sometimes compatibility.

**Client Side:** The client wishes to move beyond their current Android-only application to supporting players on iOS and Android mobile platforms, as well as traditional desktop-based operating systems. The application must be delivered as a modern, responsive HTML interface running inside the web browser for desktop clients (Linux, Mac, Windows), as well as on mobile platforms. Each will be capable of communicating with the back-end web application running on the server.

* **Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients.** Consider the following in your evaluation and articulate your findings in the software design template:
  + What is required of the application development process to ensure the application is compatible with all web browser platforms and mobile devices?
* When supporting multiple types of clients we must have the staff and general resources to back up the different clients.

Generally speaking it would be a good idea to have individual support for each client type(windows linux mac, android, ios). By doing so you can have a more focused support that will allow to be specific for the client. In doing this problems solved for the corresponding client will be easier to deal with as they are set to the client, Over time becoming more familiar with all of the nuances. Granted having focused support can be more expensive to maintain as opposed to having one broad support.

One of the most important parts to process is the testing of all web browser platforms and or mobile devices.

Starting out it can be seen as wise to keep things as simple as possible, the more implemented the more can go wrong. So starting out with the bare bones necessary to run the application for testing purposes. After getting the basics the next is how do we ensure this will work on the desired clients? Test, Test, Test! There are lots of ways to go about this. One of my favorite approaches is the use of virtual machines. More often than not you can find and run whatever operating system you’d like. Using virtual machines will allow you to install all you need in one convent place as well.

The last idea would to also provide fallbacks incase you were to run into issues. Say after testing you know a certain device or client does not support what the others do. By providing a fall back or another method for that device to use what the others cant this acts as a sort of catch in case things don’t work right. “If this option doesn’t work use this one instead”.

**Development Tools**

* **Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on each operating platform.** Consider the following and articulate your findings in the software design template:
  + What impact do these technical requirements have on a development team? Consider whether multiple development teams may be needed.
  + Are there licensing costs related to the development tools?

As stated before with supporting multiple clients it would be good to have individual support specializing in that client. There are two factors I think of when deciding on developmental tools. From the above listed tools we could use the same tool across the board such as HTML, Javascript,CSS. The question lies in how these interact with the clients OS. If after testing they run smoothly on all then we can keep things simple and run with it across the board. If however a certain tool may run more optimal on another client then it might be relevant to split things up to keep running smoothly. However if we were to go down that road it would be ideal to have separate teams to specialize in them. When splitting this can directly effect the teams and will require more communication on what needs to be done and how it will do so. Once done testing will have to commence to verify smooth functionality.

Depending on the choice of tool you go with there may be a license fee attached for such tools as HTML or CSS.