

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 03/24/20 | Hunter Blunt | Added executive summary, design constraints, and domain model |
| 1.1 | 04/04/20 | Hunter Blunt |  |
| 1.2 | 04/19/20 | Hunter Blunt | Recommendations section updated |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants a web-based game that operates on multiple platforms. The game will be based on their current app, Draw It or Lose It, which is only available in Android for now. To facilitate this, we will redesign the game to work across multiple platforms and operating systems.

## [Design Constraints](#_2et92p0)

The website will initially have some language constraints. Luckily the android application is written in Java, so we only need to create the website in Java as well. After that we can integrate the two.   
Another major fix we will need to work on is getting the website to work on all operating systems. Not to get into too much detail here, but as an example Safari on Apple OS no longer supports Java.   
Finally, the last main design constraint we will be faced with is time. Depending on how long it takes to come up with solutions to our current fixes, it could push our timeline back.

## [Domain Model](#_8h2ehzxfam4o)

****

Team and Player all are inherited from Entity. Game, Team and Entity, all 3 have common attributes as id and name. Therefore, it is defined at Super class Entity.

Relationship between Team and Player is of "has a" relationship. Team has Players. Similarly, Game has Teams and GamesService organize (has) Games. In UML it can be represented by Aggregation. Has-A relationship means that an instance of one class has a reference to an instance of another class. In above diagram, GamesService has reference of Games, Games has reference of Team and Team has reference of Player. One GameService can have multiple Games, Each Game have multiple Teams and Teams comprises of multiple Players.

## [Evaluation](#_2o15spng8stw)

Below is and evaluation of each operating system and the pros and cons to each in the fields of server, client, and development tools. (NOTE: The dev tools are all the same because they can be used on all platforms. The only exception is Mobile devices which is the same with additional options)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac has a more intuitive terminal commands to make changes, access, and create the server. | Has flexible terminal commands to make changes, access, and create server. It is also much more cost efficient. | Windows has a lot more software available to utilize to create, access and change a server. | It would be best if the server had a single location and wasn’t based on a mobile device. Specifications are much better elsewhere. |
| **Client Side** | Mac requires a moderate amount of skill and time. It has a medium range in cost that would be similar to Windows. | While Linux is much more flexible in what you can do, it requires much more expertise and time than other options. It is however the most minimal in cost. | The cost for Windows would be in the medium range like Mac. It does however require less expertise and time to use than both Linus and Mac. | It can be more difficult to implement; however, it does provide a lot more flexibility. So, it can have a higher cost, and lower expertise and time. |
| **Development Tools** | Languages: Jscript, CSS, HTML, Python, Java, PHP, Ruby, SQL Tools:  PyCharms, Eclipse, Visual Studio, Github, Cassandra | Languages: Jscript, CSS, HTML, Python, Java, PHP, Ruby, SQL Tools:  PyCharms, Eclipse, Visual Studio, Github, Cassandra | Languages: Jscript, CSS, HTML, Python, Java, PHP, Ruby, SQL Tools:  PyCharms, Eclipse, Visual Studio, Github, Cassandra | Languages: Jscript, CSS, HTML, Python, Java, PHP, Ruby, SQL, Swift Tools:  PyCharms, Eclipse, Visual Studio, Android Studio, XCode, Github, Cassandra |

## 

## 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**: We should use Microsoft as the operating platform for the game. It has the widest available support and has a lot more support tools than any other platform.
2. **Operating Systems Architectures**: The main architecture of windows is layered between “user-mode” and “kernel-mode”. The difference between these is that kernal’s executing code has unrestricted access to the underlying hardware. User-mode’s executing code can not directly access hardware or memory.
3. **Storage Management**: For storage management I would recommend the decentralized cloud storage platform Sia. It allows you to exploit the benefits of the decentralized cloud storage environment. It cost on average about 1-2 USD a month per 1TB. On top of these benefits siacoins are used to pay for space, which can be mined and traded to maximize cost benefits.
4. **Memory Management**: 64-bit Windows OS memory management provides up to 16TB of addressable memory space divided equally between kernel and user applications. The 16TB of physical memory is divided with 8TB of virtual memory to kernel and 8TB to user.
5. **Distributed Systems and Networks**: Using a cloud-based storage management system the components of the distributed systems and networks becomes much more manageable. Using Windows as the operating platform will allow us to easily program the game for multiple platform capability. From there pieces of the applications data will be scattered across the cloud’s servers. Then whenever a user sends a request via their OS’s internet browser, the data across multiple servers will be sent to the user. This decentralized approach makes it very easy to get around all the issues of a centralized system because if one server goes down another can be used with the same data to be sent to the user. The practically gets rid of issues dealing with connectivity and outages.
6. **Security**: With modern internet browsers and a decentralized cloud system, security is easy. Using Sia or most other decentralized cloud storage management platforms come with data encryption they use across their entire decentralized network. Besides encryption anther thing we can do is to get a security certificate for the games website as proof of the website’s identity to the browser.