

Draw It or Lose It Game

# **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 |
| --- | --- | --- | --- |
| 1.2 | 08/01/2021 | Noah Archibald | Completed recommendations section. |

**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 would like to develop a web-based game that will be able to run on a variety of operating platforms and web browsers. The game is called *Draw It or Lose It* and the aim of the game is for multiple teams to compete in four rounds each consisting of one minute each. Drawings are rendered incrementally and fully completed after the first 30 seconds of each round. Contestants then guess what they think the drawing or picture is of. If no one guesses correctly from the first team, members of the opposing team each get a guess.

## [Design Constraints](#_2et92p0)

Based on what features The Gaming Room would like to have built into the app, the following constraints are:

* Game creation involving more than one team.
* Each team holds many different members.
* Allow for unique game and team name creation and verification of names.
* Allow only one game instance to occur in memory at any given time.
* Able to run on several operating platforms.

Since the game is already available on Android, the game needs to be adapted to operate within a virtual web-based environment. There should already be the underlying logic and foundation for the code which can then be “translated” to run on other operating platforms, like windows, linux, and mac OS. Most likely then, many development teams will be needed to ensure that the final programs are rigorously tested for bugs and consistency.

## [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 creates a sort of “link” through which the Game, Team and Player classes all inherit Entities attributes. The Entity class could now be considered a parent or superclass and the Game, Team and Player classes are now all its child or subclasses. Common references shared between the classes can now be referenced and shared by all classes in the program. The GameService, Game, Team, and Player classes all represent an aggregation relationship. This means that each of these classes has an instance of one class with a relation to another instance in a separate class. In essence, they are one way references to another classes instances. In the diagram we can see that GameService references Game, Game references Team, and Team references the Player class, all of which inherit from the Entity superclass.

****

## [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** | Mac offers some advantages when it comes to terminal command flexibility when configuring and accessing or changing data on the servers. However, Mac hardware is very limited and the software that supports it, even more so. | Linux was designed and based on the Unix shell, so it boasts a powerful, reliable, and stable operating environment, especially for configuration of servers. Linux is ubiquitous for server hosting and therefore there is abundant technical support for operating a Linux based server. | Windows offers a lot of unique software features developed by Microsoft exclusively for Windows based servers that can be used to streamline efficiency or troubleshoot issues. Windows is also one of the most ubiquitous OS’s so there will also be abundant technical support, like Linux. | Although mobile devices are weak, in computing power, compared to other devices like computers, they are able to run locally and only need to ping the server when they need to access, verify, or alter information. So, the server costs for supporting mobile devices would be minimal. Also, everyone has a phone nowadays, so continuing to take advantage of this market is vital. |
| **Client Side** | Developing an application to run solely on Mac ultimately requires Mac hardware, or some version of developmental MacOS, which can be costly. It may also be necessary to recruit a new development or maintenance team solely for MacOS. | Linux was essentially built to be lightweight for the purpose of development and programming. It is also open source, so there are many different solutions to many different problems. There is not much expertise required to work with Linux, depending on the technical background of the person. | Windows offers a plethora of unique software capabilities and functionalities that allows for efficient development of windows-based applications and web sites. Its prevalence as an operating system also means the expertise to use this OS is minimal. | Developing for phones and tablets is very similar but designing the layout of applications and web sites for them is completely different. Consider the difference in screen sizes between a mobile phone and a tablet for example. More development time will have to be spent to design separate layouts for the web site. May also have to invest in mobile development teams, increasing cost. |
| **Development Tools** | The relevant programming languages for MacOS are Swift and Objective-C, which is based on the standard C language. Mac supports a wide array of development tools like iTerm2, XCode, Homebrew and many more. | Linux supports pretty much any programming language or IDE. C, C++, Java, HTML, are among the most popular that are supported. Linux also offers a bevy of IDE’s as well, the most popular of which is Eclipse. | Windows applications are primarily developed in the C++ language, but offer support, upon installation, for many other languages as well. Microsoft’s visual studio is a very well reputed and widely used IDE that runs on windows as well. | Development languages vary from device to device. iOS primarily operates on the Swift language whereas other OS’, like Android or Tizen, use Java or C++ respectively. For working on iOS devices, XCode is the most prevalent IDE. Android Studio is the most prevalent IDE for Android devices. |

## 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**: Primarily due to its ubiquity and prevalence among most devices, I would recommend windows as the operating platform that the Gaming Room focus on developing first for their expansion into a browser-based version of “Draw It or Lose It”. Windows also offers a variety of software packages that can help with deployment and streamlining the game later in the development process. Windows also supports a wide variety of IDE’s that could lower overhead costs for development as well.
2. **Operating Systems Architectures**: Since Windows is so ubiquitous, it is unlikely that any developer working on porting the game would be unfamiliar with it’s GUI or technical specifications. For server development and implementation, utilizing Window’s power shell utilization makes configuring and deploying server settings much easier than other operating systems. Also, Windows supports programming in multiple languages which will make planning for future ports of the game to other devices easier and more cost effective.
3. **Storage Management**: Windows has many useful features built into it that allow for much easier storage management and maintenance. One of those features is their storage sense capability which allows the user to effortlessly manage files on the local hard drive as well as visualize space being consumed by particular parts of the game application. Implementing a local storage solution with a backup to the cloud can then be integrated to insure there are no issues if or when a mistake may be made, or an unforeseen accident occurs.
4. **Memory Management**: Through implementing a stack to store and hold files locally on a user’s device, say the library of pictures and files required to run the game, the core of the game can be stored securely and cohesively in one place on a user’s computer. This also allows for more effective RAM management as they can only hold temporary values related to the current instance of the game that is running and allow the CPU to focus on other tasks allowing for a smoother and more memory efficient load on the computer. Ascribing to this technique during the development process is beneficial as well, as it allows for the IDE being used to consume less resources and operate faster thereby increasing development speed in the long term.
5. **Distributed Systems and Networks**: Cross-platform integration can be achieved through third party development environments that allow for the seamless integration of one language with another, like MonoDevelop or KDevelop, along the with use of a backend solution implemented through routing traffic between servers that physically host instances of the game via the cloud. While it is costly to deploy, maintain, and monitor servers for each platform the game will operate on, it is the most dependable way to ensure there is zero downtime when an unforeseen event occurs and implementing the suggested memory management system simultaneously ensures these solutions can be most effective.
6. **Security**: Unfortunately, one of the downsides of Windows ubiquity is that it is the most scrutinized operating platform when it comes to people searching for security exploits and bugs that can be used for malicious intentions. However, Microsoft does an excellent job at updating their accompanying security suite to ensure the local user’s systems remain secure by pushing updates automatically to their devices. When securing a user’s connection to the server hosting the game however, end to end encryption via creating and verifying SSL certificates and utilizing most modern browser’s integrated SHA-256 encryption algorithms can ensure that the user’s data is not being sniffed while transferred over the internet. Implementing local security solutions relevant to the physical servers is crucial to protecting user information as well and security configuration during server deployment is vital to maintaining a secure link between end user and the server host.