

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 | 03/19/2023 | Benjamin Leanna | Project 1  Project 2 |

**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 create a application called “Draw It or Lose It” that is loosely similar to the 1980s television game *Win, Lose or Draw*, where teams compete to guess what is being drawn. Instead of players drawing images on an easel to help team members guess the puzzle, the application will render images from a large library of stock drawings as clues. A game consists of four rounds of play lasting one minute each and the drawings are rendered at a steady rate and are fully complete at the 30-second mark. A game will have the ability to have one or more teams and 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 the team’s name and only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

* The application must be web based and be able to run on multiple different environments.
* 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 may exist in the system memory at any given time.
* Each team will have multiple players assigned to it.

## [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 GameSerice class will be the driver and manager for implementing player’s game instances that can refer to 0…\* (zero to many) amounts of instances of the Game objects. In effect of this, the Game object can serve from 0 to many instances to the Team object and thus that to the Player objects. The Game, Team, and Player objects are all classes that extend from the Entity class that serves as the base class for all objects in the game itself. The base class then contains the id and name properties which are all unique to identify the person playing. This UML diagram shows encapsulation by protecting the instances running and making sure that each one is unique. Because of this, abstraction is also a part of the design by allowing a singleton tester to be used to be able to address and solve the program’s functionality without affecting the player. By using different unique identifiers for each instance of a game, team, or player, polymorphism is used.

****

## [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** | For good points, Macs have ease of use through administration. The licensing, however, is quite expensive and wouldn’t be ideal for a large scale distribution in smaller companies based on cost. It is also not open source so you are restrained to their configurations. | Linux has no licensing fees, is open sourced, and free to run. Due to this, Cloud or in-house server hosting is great because of no licensing costs. The major setback would be the usability from users that are not familiar with Linux so training employees could be a cost, albeit hopefully smaller than paying for licensing fees. | Windows is in between both Mac and Linux regarding costs. It has lower fees than Mac and even though not as expensive, obviously more than the free Linux license. Windows is also known wider but that means it also has more security vulnerabilities as more people use it. | Mobile devices are able to server host and can be developed for both websites and phone interchangeably but would be extremely hard to handle large amounts of traffic through and would not be a good choice. |
| **Client Side** | Since this is a web based application, as long as it is made for a browser that Mac is compatible with (basically all common browsers), client side shouldn’t have a problem running the application on a Mac. | Even though this is the least used OS out of the four mentioned here, as long, again, as the client is able to run a compatible browser with correct plug-ins needed for the application, it should run just fine on Linux. | The game as Mac and Linux. Windows users should have no problems running instances of the game since it is a browser-based Web application. | Unlike the other two, Mobile interfaces can run into problems as they don’t have the configuration abilities that modern OS’s have. Depending on the brand and type of phone, it could present a problem with its web browser not being compatible. |
| **Development Tools** | Development on a Mac for this project would be a great place for Java and Python. Unlike C++ where compiling constraints would come into play across different platforms, using Java and using a web-hosting application like XAMPP to host the stie would work great. | Much like MacOS, a Java runtime/development environment will be required. XAMPP or LAMP or other web-hosting programs will be required for server deployment. A Java IDE will be needed. Linux shares the same client requirements as MacOS for web-based client site. | While Windows also is great for app development and is capable of Java and XAMPP like the previous two, it also has its own web hosting application named IIS. | For Mobile development, both Apple and Android have specific libraries for application that are easy to use and that are secure. |

## 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**: While Linux licensing is free, I would choose Windows as the best development environment for the Draw It or Lose It application. Based on the quantity of people that use Windows and the amount of and availability of tools created for the OS, it would make the easiest choice when starting the project.
2. **Operating Systems Architectures**: Again, because of the easy-to-use GUI and amount of people familiar with Windows, this would my choice. Not only is it easy to use, Windows also has great technical support in case of problems that may occur and vast online resources to go to since it has the largest user base.
3. **Storage Management**: Since this game is quite small and Web-based, using an online hosting company would be the route I would take like OneDrive or MS-SQL.
4. **Memory Management**: Since Windows allows you to specify the amount of memory that you want to allocate for each application in a virtual instance during runtime, I would use that OS.
5. **Distributed Systems and Networks**: Cloud based tech servers with company databases to let players call the app on whatever environment they are using is a great feature and something I would suggest for implantation in this application.
6. **Security**: Although security will always be an issue even though perhaps lower for Linux, using a Windows based security is easy and efficient. You can create databases, firewalls, and Windows offers security support.