

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 2.0

## 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 |
| --- | --- | --- | --- |
| 2.0 | 06/18/21 | Thomas Cogley | Added UML, Evaluation and Recommendation |

**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)

We are developing a game where teams compete to guess what is being drawn. This application will render images from a library of stock drawings as clues. The game will have the ability to have one or more teams. Each team will have multiple players assigned. Game and team names must be unique. Finally, only one instance of the game can exist at a time. We will use singleton classes to ensure there can only be one instance of everything at a time. This will include team names and player names on each team. Our program will loop through the team lists and player lists to ensure there is no duplicates within the game.

## [Design Constraints](#_2et92p0)

We will have multiple design constraints when developing this application.

* We must ensure that the App stays within our client’s budget. This means we must accomplish what the App is supposed to do and not add extra things. We will perform testing and QA at they same time we are developing the App instead of at the end.
* We must also ensure that the Application works on all operating systems. This includes Mac, Linux, Windows, and Mobile Devices. To do this testing will be done on multiple OS.
* The last constraint I will mention is our Client’s feedback. We must be aware that changes to the application may happen at any point. It is for this reason we will stay in constant communication with our client so they can observe the App at certain points. If they require something different, we will adapt.

## [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 following UML Diagram explains how the program will function. Are parent or super class will be Entity. This class will have the private variable id and name. This super class will get id and name and push it to a string that the sub classes will receive. The classes that sub classes of Entity are Game, Team and Player. Both the Game and Team classes will create lists. The Game class will create a Team list. This class will add a team to the list if the team has a name that does not already exist. The Team class will create a player list. This class will add a player to the list if no existing player has the same name. The program driver class will run the main program and the singleton tester class will test the singleton instances. Finally, the game service class creates a list of games and checks whether an existing game has the same name. If it, does it will not create a new game.

****

## [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** | Expensive to develop. Easy server configuration. Very user friendly. | Not very costly. Uses a command shell. Few people use Linux so not as many software options as Windows. | Could be costly. Very user friendly. Many different software and program options. | Cheap to develop. Many different mobile OS could take a while to develop program. Biggest challenge would be making sure game works on all mobile OS. |
| **Client Side** | Can be costly. Time to support Mac may be longer than Windows but less than other OS. Will need a fundamental understanding of Mac to develop. | Will not cost much to develop. Understanding Linux is important. May take much time to get everyone up to speed. | Will not take much time or expertise to develop on Windows. The cost may be costly to develop on Windows. | Will take a lot of time and money to develop. Due to many different Mobile OS. Much testing and QA will be needed. |
| **Development Tools** | There are many different tools that can be used on Mac. Such as JavaScript, CSS, Python and more. Also, Eclipse, IntelliJ, NetBeans and many more IDEs can be used. | Many IDEs can be used on Linux. Such as, Eclipse, NetBeans, Visual Studio and more. Languages that can be used include, Python, C++, Perl, Java and many more. | Like Mac there are many IDEs that can be used. Eclipse, Visual Studio, IntelliJ and more. Also, languages include, C++, C#, JavaScript, Java, Python and many more. | IDEs for mobile Apps include, Visual Studio, IntelliJ, XCode, Android Studio and many more. Languages include, Python, Java, HTML, Ruby, C++ and many more. |

## 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**: Windows would be the best Operating Platform to use. This is because Windows has a very user-friendly interface and many different programming libraries to develop the App.
2. **Operating Systems Architectures**: Windows allows for many different programming libraries. Also, with its many different software libraries, program developers have many options for the App.
3. **Storage Management**: Windows offers many different cloud-based servers. This will allow the program to be backed-up or expanded on through the cloud.
4. **Memory Management**: On 32-bit Microsoft Windows you can address each process up to 4-gigabytes of memory. Memory options on Windows includes physical and virtual address space. This will help the application fun fast with little errors.
5. **Distributed Systems and Networks**: With the App communicating between different OS and having potentially many people on it, issues may arise. Controlling traffic and making sure the different operating systems allocates jobs to processors correctly will be important.
6. **Security**: Windows offers many different security features. It is easy to have anti-spyware protection. Also, there are many different VPN options on Windows. It is very easy with Windows to create account control settings. This will allow only certain Users to access the App on different levels.