

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 | 7/22/2021 | Renaldo Musto | Added the entity class and completed the player and team classes. |

**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 goal of this project is to develop the basic framework for the game so that it can be used in a web based environment. To be specific we must implement code that allows for the creation of multiple teams per game and multiple players per team. We also need to ensure that the game and team names are unique and that there is only one instance of the game loaded into memory at any given time. We can assure that there are no duplicate games or teams by using an iterator to search through the list we have. We can make the class containing the game a singleton to ensure that only one instance can exist at a time.

## [Design Constraints](#_2et92p0)

The code must be adapted from the Android specific platform to be available in a web based environment which means that compatibility for several browsers must be implemented. The game will also need a website to exist on so that users can access it via their web browsers. The game should also be compatible with multiple operating systems such as Windows, Linux, MacOS or mobile devices. This will make application development somewhat difficult especially in the realm of testing because each web browser will have to work on all three operating systems.

## [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 GameService, Game, Player and Team classes all inherit attributes from the Entity class. The ProgramDriver class uses the SingletonTester class to run the testSingleton method. According to the multiplicity stated in the UML diagram there does not seem to be a defined range for the amount of Game, GameService, Player or Team classes that can potentially exist, it can go from 0 to infinity. The diagram provides an excellent abstraction of the program allowing us to see the whole code as just a few classes. The Game, GameService, Player and Team classes can interact with each other for example, the Player class is ultimately under the Team class which is under the Game class but none of these classes inherit properties from each other even though they are associated in the way where a game needs teams and a team needs players to function properly.



## [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 has the benefit of being able to run Unix and it has a much larger market share than Linux so there will be more trained professionals to work on it. Hosting a server on this operating system does have the drawback of reduced customization. Since all Macs are pre-built it will be difficult to find the proper amount of computing power for our needs wasting money because we will never get an exact fit and to repair or replace the machines is rather expensive. | Linux has the benefit of being able to immediately use Unix which will be useful because it can work on all 3 major operating systems. Ubuntu also has its own server software and this is also one of the most common distributions of Linux so we will be able to cheaply and immediately host from this operating system. The main drawback to hosting on Linux is the lack of trained experts on that operating system. | Windows can support use of Unix but it isn’t the most effective method for Windows so it may cost a bit extra to develop something on another platform such as Docker. If the game plans on hosting thousands of users then it will need a considerable amount of server space which is costly to both setup and maintain. It will cost thousands to buy the hardware required for such an activity if we don’t already have it available. Windows has the benefit of being able to run on basically any hardware so we can make a custom server that is exactly what we need it to be increasing efficiency and reducing costs. | Hosting a web based service meant for thousands of players on a mobile device would not be a good idea they simply do not have the processing power required to do such a task. Unless a specially designed machine were created I don’t think hosting an application of this scale on a mobile device would be possible. |
| **Client Side** | Mac will need support for all the major web browsers as well as its own browser safari. This will increase costs slightly. Considering that the application will be web hosted there should not be much difference at all between Firefox run on Mac or Windows so we can save time and money in that regard. We would still need to test on both OS but we should not need to develop a separate set of code. Someone with expertise in MacOS would be useful for the project but overall the regular software development team should work fine. | Linux has many distributions but thankfully only a few are widely used by the average consumer as an OS so we will not need to test and develop for every possible platform. Most major browsers such as Google Chrome and Firefox has ported their software to Linux as well so a lot of the work in compatibility is done for us. Linux doesn’t have an official browser like safari or explorer so we do not need to develop for something like that making Linux most likely the cheapest to develop for. | The cost should be relatively on par with that of developing for MacOS. Windows runs the major browsers but it also has the drawback of having its own unique pre-installed browser like mac expect it is called Edge. This will increase costs slightly as we will need to port our application for this browser as well. This will take a little more time to accomplish as well. | Mobile will most likely be the most expensive and time consuming to develop given the wide range of possible browser and OS options available. We will need experts in mobile design as we will need to have a separate mobile version of our website that is coded very differently with things like touchscreen support and a different UI. |
| **Development Tools** | MacOS is capable of running Java code without an issue so I don’t think we would need unique tools for this Java code and Eclipse as an IDE should work fine as it is available on MacOS. | Linux is both able to run Java code and has a version of Eclipse for it so I don’t believe we would need unique tools for Linux development. | Java is the language we would want to be coding in and Eclipse is one of the most common IDE’s for developing Java code. We can use that to create a Maven project which will allow us to make the web based application. | Android and Apple devices will have no problem running the Java code. Now it is very difficult to develop code on a phone so we will need an app development software on separate computers and a way to relay that information to the phone for testing. |

## 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 believe that Windows would be the ideal operating system to host the game on. This will allow in my opinion the easiest and cheapest way to expand the player base.
2. **Operating Systems Architectures**: Windows uses the RESTful API architecture to build distributed applications. This means we will need to make our application accessible via an http request which is the standard in this kind of system architecture. We will also need to send information between the client and server as JSON files which is standard protocol for the RESTful API as it allows almost any programming language or operating system to understand the data being transmitted as JSON is a very simplistic and straightforward way of organizing information that even an untrained person could read it with ease.
3. **Storage Management**: Ideally we would host most of our information on a remote server preferably an AWS server as that seems to be the cheapest option available for our needs while still offering appropriate speeds. We should host the library of images as well as player and game data on the servers only to be loaded into the clients RAM as needed to play the game.
4. **Memory Management**: Once the player logs into the game their player data and that of their teams should load into their memory specifically the RAM as soon as they join. Then from there once a game starts the server will download the correct image to each players RAM so that once everyone is loaded in the image can be loaded quickly. As successive rounds occur the image in the clients memory will be replaced each time as to not overload their RAM. Before the game is ended the client will send the updated player and game information back to the server so it can be archived there then the users RAM would be cleared of all data pertaining to the game once they closed the application on their computer.
5. **Distributed Systems and Networks**: If we host the game over the internet we will be much more concerned with the application running on various web browsers rather than an individual operating system. The three major operating systems use a few primary browsers that cover most users which are Chrome, Firefox and Edge. We will also need support for Safari on MacOS but as long as our application runs on these four web browsers they should be able to be accessed on any of the three operating systems we are working with because those browsers are already ported to work with that OS. None of these operating systems are particularly faster than each other so we should have roughly even game play speeds across games and teams even if they are using different operating systems.
6. **Security**: Information sent to and from the server or between players should always be encrypted to ensure that even if the information is somehow intercepted by a malicious party it will be unusable. We also have some basic authentication upon loading the application that requires a username and password to access sensitive information like player data. All three major operating systems have an authentication feature of some kind usually a simple username and password to access the computer which further helps with security if someone were to forget to log out of the game.