

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 | 6.7.2020 | Jordan Sims | Addressed the following sections:   * Executive Summary * Design Constraints * Domain Model |

**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 *Draw It or Lose It* is to create a game where teams compete to guess a stock image being generated by AI. Many teams may be involved in the game. The game has already been created for Android, and we are going to be further porting it to a web-based format that can service multiple platforms. This could will involve rewriting the game from the ground up for an entirely new platform. We will use the existing Android app as a model and base our new app around it.

## [Design Constraints](#_2et92p0)

The Design Constraints are as follows:

* It must be web-based
* It must be able to run on multiple platforms through the web
* It must have the ability to have one or more teams involves
* Each team can have multiple players assigned to it
* Game and team names must be unique
* Only one instance of the game can exist in memory at any given time

All of these design constraints will be taken into account when developing the program.

## [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 below domain model program is for the proposed program, *Draw It or Lose It.* First, you have the program driver and the singleton tester. These are common in such programs. You need a module to drive the program and a program to test that Singleton structures are working so that not more than one instance of the game is running. The rest of the boxes are modules that setup the classes. First, there is the Entity class. This is the base class introduced to hold common attributes and behaviors. Then, there is the Player class. The class gets player information such as name and what team they are on. This takes us to the Team class, which is referenced in the Player class. This class gets the team name, and then also adds players to the team and puts them in a list. Next, there is the Game class. This class dictates the attributes of the game, such as getting the team names participating in the game and getting the id for the game itself. Finally, we are led to the GameService class. This is a Singleton class pattern that makes sure that there is only one instance of each game, team, and player.

****

## [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 comes equipped with the Apache web server software, so out of the box it is able to run a server. They also have macOS Server, which costs just $20. Although, this probably does not meet the needs of the client. | Linux is a very popular OS for running servers. It can cost around $1,000 a year for an average server but depending on traffic for the website it could be more. Because Linux is open source, this also makes it cheaper and easier to run a server compared to other operating systems. | Windows has similar costs to Linux servers, but can be more costly. The pro of windows is its large amount of official support and integration with Microsoft. | It is unreasonable to attempt to run a server from a mobile device. |
| **Client Side** | See the development tools section for how we will reach the goals the client has set for us. As far as time goes, if we choose to use GameMaker Studio that will cut down time considerably as we will only need to build the application once and then export to all the different systems. We will, as stated below, need to either train a team on GML or hire a team of GML programmers. | See the development tools section for how we will reach the goals the client has set for us. As far as time goes, if we choose to use GameMaker Studio that will cut down time considerably as we will only need to build the application once and then export to all the different systems. We will, as stated below, need to either train a team on GML or hire a team of GML programmers. | See the development tools section for how we will reach the goals the client has set for us. As far as time goes, if we choose to use GameMaker Studio that will cut down time considerably as we will only need to build the application once and then export to all the different systems. We will, as stated below, need to either train a team on GML or hire a team of GML programmers. | See the development tools section for how we will reach the goals the client has set for us. As far as time goes, if we choose to use GameMaker Studio that will cut down time considerably as we will only need to build the application once and then export to all the different systems. We will, as stated below, need to either train a team on GML or hire a team of GML programmers. |
| **Development Tools** | I have discovered a tool that will allow the release of the game being designed on MacOS, iOS, Windows, Linux, and Android. It will also be able to run on a web based page in HTML5 (as well as all three major consoles, if we so choose). The tool is called Gamemaker2 and it is what I suggest the team uses. There are two drawbacks to this: First, licensing for release will need to be bought for each operating system, but this is relatively cheap at around $450 for lifetime use. The other main drawback is that this software uses its own language called GameMaker Language (GML) which our programmers will need to learn. The good thing is that the language is not that difficult to learn! It is very similar to Python! | I have discovered a tool that will allow the release of the game being designed on MacOS, iOS, Windows, Linux, and Android. It will also be able to run on a web based page in HTML5 (as well as all three major consoles, if we so choose). The tool is called Gamemaker2 and it is what I suggest the team uses. There are two drawbacks to this: First, licensing for release will need to be bought for each operating system, but this is relatively cheap at around $450 for lifetime use. The other main drawback is that this software uses its own language called GameMaker Language (GML) which our programmers will need to learn. The good thing is that the language is not that difficult to learn! It is very similar to Python! | I have discovered a tool that will allow the release of the game being designed on MacOS, iOS, Windows, Linux, and Android. It will also be able to run on a web based page in HTML5 (as well as all three major consoles, if we so choose). The tool is called Gamemaker2 and it is what I suggest the team uses. There are two drawbacks to this: First, licensing for release will need to be bought for each operating system, but this is relatively cheap at around $450 for lifetime use. The other main drawback is that this software uses its own language called GameMaker Language (GML) which our programmers will need to learn. The good thing is that the language is not that difficult to learn! It is very similar to Python! | I have discovered a tool that will allow the release of the game being designed on MacOS, iOS, Windows, Linux, and Android. It will also be able to run on a web based page in HTML5 (as well as all three major consoles, if we so choose). The tool is called Gamemaker2 and it is what I suggest the team uses. There are two drawbacks to this: First, licensing for release will need to be bought for each operating system, but this is relatively cheap at around $450 for lifetime use. The other main drawback is that this software uses its own language called GameMaker Language (GML) which our programmers will need to learn. The good thing is that the language is not that difficult to learn! It is very similar to Python! |

## 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**:

My recommendation for the operating system to use is Linux, as it is an incredibly versatile system that can handle running the servers on all the perquisite platforms. This is due to the nature of it being incredibly customizable because the platform is open source and so easy to modify.

1. **Operating Systems Architectures**:

I believe the best architecture choice would be a client-server setup. We will set up the servers on our side which will then receive and respond to requests from clients. This could either be through their browser or through the game. We will need a simple client-server setup, as opposed to something more complicated such as a three-tier network setup. Clients will directly interact with our servers.

1. **Storage Management**: There are multiple things we need to focus on when talking storage management on our Linux system. First and foremost, we need to be space and size conscious. This means not having redundant files and compressing whatever we can to the smallest we can get it without losing functionality to our clients. We also need to make sure we have the right hardware backing up our software. Advanced server dedicated PCs running Linux with large SSDs for speed. We can also try to put as much as our storage on a cloud server so that we have less of a need for physical storage. Also, backing everything up to the could helps us cover an important facet of storage management; disaster recovery. We will need to be prepared for the potential event that shuts down our local servers and need to have a hot or cold storage site ready so that our program does not go down for long.
2. **Memory Management**: As with storage management, a large part of memory management is hardware. We need to make sure our physical systems are setup in such a way that memory is not an issue. This means having advanced RAM sticks on the hardware. Linux also implements the use of shared virtual memory. Virtual memory is the sharing of memory to be used in multiple computer processes. This will certainly make the machine run faster. Linux also utilizes memory swapping.
3. **Distributed Systems and Networks**:

Having a distributed system, with multiple hardware pieces as well as some cloud computing all working toward the goal of having the game and its network be constantly available and operational, is a very important part of keeping this game up and running. Having all these machines working together creates some dependencies, but in the end, it also helps each machine work individually. If one location goes down or has an outage, the rest will be ready to pick up in its place so the servers do not go offline, or if they do go offline, it is only for a short amount of time.

1. **Security**:

Security is incredibly important to us and our clients. First and foremost, internal controls will be a very important part of our plan. We will have our company fully compliant with Security and Organization Controls (SOC) and be audited on a bi-annual basis, especially with the Security, Confidentiality, Availability, and Privacy criteria. These controls help ensure our clients data is safe, as well as our networks. They include things such as firewall security measures, other software security measures, security awareness training for all employees, physical and logical access controls for all software and hardware sites, data encryption, and many others. We will be strictly SOC compliant. We will also employ fully efficient Linux security measures on all of our systems.