

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 | 01/24/2021 | Ray Peeples | Initial Draft |

## [Executive Summary](#_sbfa50wo7nsh)

Draw It or Lose It, which is currently available in an Android application only, needs to be adapted to be available as a web-based product on multiple platforms. The rules of play will be consistent with the current offering; but, the delivery will utilize a singleton design pattern to ensure a single instance of the service is active at any given time and/or platform while incorporating iterator design patterns to ensure games, teams and players are not duplicated.

## [Design Constraints](#_2et92p0)

* Games will have the ability to have one or more teams involved
* Each team can have multiple players
* Game and team names must be unique
* Only one instance of a game can exist at any given time

## [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 ProgramDriver and SingletonTester classes will be utilized to test the proof of concept. The Entity class is the Parent of the Game, Team and Player classes. These child classes inherit the variables to store id and name as well as the accessors to retrieve these items. Constructors will be utilized to set the id and names for Game, Team and Player classes. A GameService class can have 0 to many Game classes. A Game class can have 0 to many team classes and a Team class can have 0 to many Player classes. All class variables are private and are accessed via a constructor or accessor methods.

****

## [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** | MacOs is gaining market share approaching 10% and with the advent of new VM software are able to test all major browser software making this a decent choice. | Linux is everywhere running about 67% of all servers. It is a large community with much support albeit voluntary for the most part. | Windows provides a broad choice of technologies and with their recent embrace of open source technologies makes this a growing choice. | Other than the convenience of their portability mobile devices tend to vary by device and provide a too varied landscape for hosting web applications. |
| **Client Side** | Though market share is growing Mac is still a more expensive option than others. | Linux clients comprise a very small percentage of the market (less than 2%) and it is fading. Since it is free and support by a volunteer force, it still has some appeal. | By far the largest community of users with large numbers of technologies and tools available. | Testing becomes complicated because of the large variety of device platforms. |
| **Development Tools** | Development tools include Java, NetBeans and XCode. | C and Java languages can be used with and IDE such as Geany. | C# and Visual Basic utilizing the .net framework through Visual Studio or Java using Eclipse. | Java and Xcode. |

## 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**: Draw IT or Lose IT should be delivered from a server operating platform running Linux. Linux is a very economical option with a large community of support. Not only is it economical; but, the options for hardware are vast. Since the platform offers multicore options per machine, the environment can be scaled to offer an almost limitless number of games given enough bandwidth on the network. Employing this platform provides maximum penetration in the gaming market.
2. **Operating Systems Architectures**: The client/server architecture would be the best solution for the deployment of Draw IT or Lose IT. On the server side as discussed above, the platform could be segmented into a 3 tier architecture to accommodate the application on one tier, the database of images and teams on another tier and the clients on the 3rd. The 3rd could be Mac, Windows, Mobile and even possibly some unknown delivery channel that is yet to be discovered.
3. **Storage Management**: Employing a database as one of the tiers on the 3 tier client/server architecture would be prudent. It would provide direct access to the data requested. The requested data can be loaded into memory for use in the game. By adding storage as its own tier, the amount of information that could be supported is almost limitless (within reason).
4. **Memory Management**: Only active teams and images would be loaded into memory at any time. Once a team or image is no longer in use, the instance would be deleted from memory, programmatically, to manage the memory affectively. Most of the memory allocation can be offloaded to the client processes, removing the burden from the server.
5. **Distributed Systems and Networks**: Draw IT or Lose IT can be delivered through a series of distributed servers via the internet to ensure availablity even in the event of a server failure. Additionally, the database tier could likewise be distributed with replication to ensure availability even in the event of a single failure.
6. **Security**: By centralizing Draw IT or Lose IT in a server for distribution oven many channels (clients), the security of the product can be maintained and administered over select channels (messages) with the clients requesting service. Information can be secured and delivered only as authorized by services provided on the server.