

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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  2.0  3.0 | 11/12/20  11/25/20  12/9/20 | Michael Linsenbigler | Produced Executive Summary, Design Constraints, Domain Model Explanation  Operating System Evaluations  Recommendations |

**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 wants to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is currently available in an Android app only. The new game will be developed with a client-server environment to allow users from any platform to access it via their web browser. It will also provide a unique game name to allow one or more unique teams made up of multiple players per team. The Gaming Room will need to host local or cloud-based server services to run the proposed game. Further details of the required hosting environment will be provided below.

## [Design Constraints](#_2et92p0)

* Existing program utilizes Android App programming
* Game is web-based
* Must serve multiple platforms
* A game will have the ability to have one or more teams involved.
* 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 a team name.
* Only one instance of the game can exist in memory at any given time

Because the game exists exclusively on the Android platform, we must develop the new game from that starting point and not from scratch to maintain consistency.

A web-based game will require a client-server environment so basic user specifications will need to be considered.

## [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 Game, Team and Player classes all inherit their attributes from the Entity class. They have a child/parent relationship. That way, any changes to the data in the Entity class will be reflected in the Game, Team and Player classes. Those 3 classes are also associated to each other and can have zero-to-many instances. The Team class “has a” Player, so there is a zero-to-many association between the two. Each Game “has a” Team, so there is a zero-to-many association between the two. Each GameService instance “has a” Game, so there is a zero-to-many association between the two.

The ProgramDriver class “uses” the Singleton Tester class to ensure only one instance of each entity exists within the program. The diagram does a good job displaying the visibility of each attribute and method using either a “-“ for private or “+” for public. This allows for a more secure program because only classes that require the information are allowed to access these from other classes.

****

## [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 server OS is expensive. It does provide an intuitive interface for ease of use and configuration. | Linux server is the cheapest option. Significant experience is needed to manage a Linux server. | Windows server OS carries high licensing costs. Very robust features and configuration tools. | Android and iOS are not meant to host and manage web-based game server platforms at the scale of this project. |
| **Client Side** | End user Mac OS/hardware is quite expensive. Same interface benefits as server OS. | Supporting Linux clients would pose a large time cost as the end user environments can vary greatly. | Easiest client platform to support due familiarity the public has with Windows OS. Cheaper than Mac OS. | Many different versions and hardware configs exist for Android and iOS making supporting mobile clients challenging. |
| **Development Tools** | Common programming languages used on Mac OS are Swift, JavaScript, Java, Python, C++, HTML. Multiple teams may be needed to accommodate other platforming languages. IDE tools include Visual Studio, Eclipse, XCode, NetBeans. | Common programming languages used on Linux OS are JavaScript, Java, Python, HTML. Most Linux applications are open source and carry no cost. IDE tools include Visual Studio, Eclipse, NetBeans. | Common programming languages used on Windows OS are JavaScript, Java, C++, C#, Python, HTML. Windows development software normally carries license fees. IDE tools include Visual Studio, Eclipse, NetBeans. | Common programming languages used on mobile OS are JavaScript, Java, Python, C++. IDE tools include Visual Studio, Eclipse, AIDE, NetBeans. |

## 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**: It is recommended that The Gaming Room utilize the Windows platform for development of Draw It or Lose It in order to effectively allow the development team to expand the application to other computing environments. The ability of Windows to integrate with various development tools and design software provides the most flexible environment for the development and support of the application across multiple client platforms.
2. **Operating Systems Architectures**: The Windows system can be configured in a few different ways to act as the application’s server. I recommend utilizing an asymmetric cluster architecture composed of blade servers. The blade servers provide multiprocessing that will allow developers the ability to balance the workload of the application across multiple processors. The asymmetric cluster configuration has one server acting as a hot spare that can be swapped to in the event of another server failure. This architecture will provide excellent performance and redundancy and ensure the game service is available to its customers as much as possible.
3. **Storage Management**: The Gaming Room should utilize a combination of traditional hard disk drives (HDD)s and faster solid state drives (SSD) hardware for their storage management needs. The HDD can store less-dynamic data such as user account info and gaming session records and act as long term archive storage. The HDDs should be configured using a redundant policy such as RAID-1 to prevent data loss. The SSD would be used for storage and transfer of application critical files such as HD images and in-game scoring records. This data can be delivered at a fixed rapid rate to the client applications because of the increased I/O rate of the SSD.
4. **Memory Management**: Windows can utilize memory management techniques such as direct access and indexed allocation to allow the developers great control over memory reading/writing and access. These are also a good way to keep wasted memory space to a minimum on the server. Indexing the images in memory would be highly recommended because it would allow the application to find the image file very quickly using an index reference as opposed to searching the image files directly. The client application will have its own methods for handling memory usage and requests based on the individual operating platform being used.
5. **Distributed Systems and Networks**: The fact that Draw It or Lose It will be accessed and played using a network connection provides a good number of development options using distributed systems and network services to allow the application to be used across multiple platforms. Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. To accomplish this, each platform will need to communicate with the server using a universal system of syntax and I/O requests. A recommended solution for this hurdle is to develop the application using Representational State Transfer (REST) guidelines. REST implements a set of rules each component must follow in order to have the entire distributed system capable of communication with each other regardless of operating platform. Also, using the Java programming language will allow each platform to download and operate within an applet that provides system options that may not be native to the specific platform.
6. **Security**: The Windows operating platform is capable of providing many customizable levels of security for servers and clients. The Gaming Room should focus on finding the right balance of restriction and ease of use for its application. If there are too many layers of security involved, the client may find accessing the application a hassle. There should be a basic level of authentication for clients to gain entry into the server’s game resources using a login/password structure. This will also allow the client to track past gaming records. Windows can also protect user and application data based on login credentials; only allowing users access to the resources they need and not the entire program. The development team can use Windows’ security features to create different levels of system access and only allow resources to interact with necessary components. In this way, if there is a security breach, it stays isolated. Windows has available security software that can allow the server to monitor and log all resource and memory request made by each client to help identify any malicious patterns.