

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 2.0 | 06/02/23 | Nathan Schmidt | Added my commentary and 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 make their game, Draw It or Lose It, into a web-based multiplatform game. Hosting a server run on macOS with the game developed in Unity could be a potential solution.

## Requirements

Software Requirements requested by the client to meet include:

* Must be a web-based game that serves 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. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

Gameplay Requirements:

* Application will render images from a large library of stock drawings as clues.
* Game consists of four rounds lasting one minute each.
* Drawings are rendered at a steady pace and are fully complete at the 30-second mark.
* If the team at play does not guess the puzzle before time expires, remaining teams have the opportunity to offer one guess per team to solve the puzzle with a 15-second time limit.

## [Design Constraints](#_2et92p0)

A web-based game can only be played with an internet connection and the quality of the gameplay experience can degrade if the connection is poor. The game will have a minimum of one team participating in a game at any given time, though the inclusion of more teams is encouraged. No game can run without a team formed. Each team will have more than one player participating in a given team. No team will have only one player on their own. No names for games nor teams nor players can be the same. All components of the game will be uniquely labeled to avoid confusion amongst the system running application when referring to individual components. Using unique identifiers given to each instance of a game, team, or player, one and only one instance of a game can be saved into memory at any given time. No duplicates of game can exist.

## [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** class contains the main() method from which the game program will be executed. The **ProgramDriver** class uses the **SingletonTester** class which tests to see if a class contains one and only one instance of itself. The **GameService** class can store an instance of itself that remains static so that there is only one instance of itself. The **GameService** class stores a list of instances of the **Game** class. For an instance of **GameService**, zero to many instances of **Game** can exist. The **Game** class stores a list of instances of the **Team** class. For an instance of **Game**, zero to many instances of **Team** can exist. The **Team** class stores a list of instances of the **Player** class. For an instance of **Team**, zero to many instances of **Player** can exist. Classes **Game**, **Team**, and **Player** are each extensions of class **Entity**. Class **Entity** is a parent class to the child classes **Game**, **Team**, and **Player**. Class **Entity** and its child classes all have a unique ID and name assigned to them.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [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 OS does offer a method for server-based deployment. They also offer a tool called Xcode which helps deploy a program across multiple platforms while running computation heavy operations in the cloud server. Licensing costs start at $29.99.  Mac servers tend to be stable and well prepared to deal with the challenges of hosting. | Linux does offer a method for server-based deployment. If Linux is harder to deploy, IBM Tivoli Provisioning Manager for OS development provides a console which is easier to use for remote deployment and management of operating systems. Costs of Linux servers vary depending on a number of factors including, server provider. Among housing sites, Linux is known for stability and can be kept running for several days without having to reboot. Linux has a lot of processing power. | Windows does include Windows Deployment Services (WDS) which is a depreciated component of operating system “Windows Server” that enables centralized network-based deployment of operating systems to bare-metal computers. Microsoft Configuration Manager and Microsoft Deployment Toolkit (MDT) are alternatives that provide a more flexible and feature-rich experience for deploying Windows images. MDT has no licensing costs.  While the OS has improved a lot, Windows has a history of bad blue screens, system crashes, and bugs making it less stable than Linux or Mac. | Mobile OS are very limited in their processing power. It would also require rearranging the UI completely in most cases for games. A simple web game like this is possible with the mobile games however. |
| **Client Side** | Mac is more expensive than Linux because Mac OS X is a proprietary operating system from Apple which results in high costs for licensing. Mac OS X offers one GUI, desktop backgrounds, icons, and links that simplify the administration process. | Linux is cheaper than both Mac and Windows because it is an open-source operating system that can be obtained for free. Linux is more difficult to use than Mac or Windows. To make several resources available to users requires the use of a terminal (similar to Windows command line) which imposes a challenge for those with no technical experience. | Windows is more expensive than Linux because Windows is a proprietary operating system from Microsoft which results in high costs for licensing. Windows offers one GUI, desktop backgrounds, icons, and links that simplify the administration process. The Windows server provides and environment that is similar to the home computer making it very easy to use comparatively. | The time, cost, and effort to developing mobile games can be much lower in comparison to desktop users. Mobile games can be simple, fast, and responsive. They also don’t have much room for screen space which makes designing the UI more of a struggle to make it immersive. |
| **Development Tools** | Game Development Engines and Platforms can be used like, Microsoft Visual Studio, Unity, and Unreal Engine, which can use programming languages C# and C++. | Game Development Engines and Platforms can be used like, Unity, echo3D, and Incredibuild, which can use programming languages, C# and C++ and C. | Game Development Engines and Platforms can be used like, Microsoft Visual Studio, Unity, and Unreal Engine, which can use programming languages C# and C++. | Google provides Android Studio as a development tool, which can help write your code in Java. Apple provides XCode and has Swift as its programming language. |

## 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**: Linux could be a reliable OS to use for development of a web game for multiple platforms.
2. **Operating Systems Architectures**: Linux is an open-source operating system. Linux’s code is available to anyone and can be freely modified and distributed. Linux has plenty of community support for those who need help working on the comparatively difficult code. Other operating systems are able to deliver high performance, however; Linux excels because it allows large amounts of people to work simultaneously. The popularity of Linux makes it network friendly. Linux is capable of handling a wide range of file formats.
3. **Storage Management**: Generally, Linux uses the concept of logical volumes to produce storage for users. Logical volumes abstract the storage that is available to the user from physical disks. These logical volumes are managed by a software system called *Logical Volume Manager* (*LVM*). The LVM operates by grouping physical volumes into a volume group. LVM then manages a volume group as one pool of storage that is split into one or more logical volumes by the LVM.
4. **Memory Management**: The operating system stores temporary memory for functionality being used during a game. Data that needs to be stored long-term such as high score can be stored in 24 and 32-bit addressing. With Linux, the CPU and memory of a computer generally stay stable and do not slow down or freeze. A computer can be rebooted after a short time if it stops or freezes.
5. **Distributed Systems and Networks**: A host server will hold the processed information to run the game. Individuals with various platforms will join online provided a strong internet connection. The server will send information to the individual players and the players will respond by sending back information via the internet. Developers will need to accommodate for the different operating systems used in the programming for the game.
6. **Security**: Linux is far more secure security wise than other operating systems. The purpose of the application is to preserve the privacy of the user.