

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 | 05/23/21 | Shawn Neal | Filling out document - Executive Summary, Design Constraint, Domain Model |
| 1.1 | 06/7/21 | Shawn Neal | Filling out document - Evaluation |
| 1.2 | 06/21/21 | Shawn Neal | Editing full document & filling out final Evaluation |

**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 is wanting to take their game, Draw It or Lose It, from its current single platform version to a web-based, multiplatform version. The Gaming Room is interested in help to develop the web-based version, as their team does not have the experience or technical expertise to do so. They have technical requirements surrounding the game sessions, game, team and player names, and the use of a unique identifier for each object within the game.

## [Design Constraints](#_2et92p0)

* Game is web-based. Multiple users will interact via an internet connection.
* Game will be available on multiple mobile platforms, Android & iOS, and a web-browser version.
* Users should connect to other users from all platforms.
* The game stores and loads a large number of images.
* Users should be able to securely connect to the game and to other users.
* Game data and user data should be encrypted.

## [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 Gaming Room UML Diagram is composed of seven classes: Program Driver, Singleton Tester, Entity, GameService, Game, Team, and Player.

The Program Driver is the driver of the application, it contains the public main() method. It also associates to the Singleton Tester class, which it uses to test that it is the only version of itself.

The Entity, GameService, Game, Team and Player are all connected. Entity is the super class, and Game, Team, and Player all inherit from Entity, using attributes from it, and overriding one of its methods. GameService is composed of objects that are of the Game class, which is composed of object of the Team class, which in turn are composed of objects of the Player class.

The Entity class is formed of two private attributes, to encapsulate the data, one private method, and four public methods. The GameService class consists of five private attributes, one private method and seven public methods. Game & Team both consist of one private attribute, and three public methods. Player is composed of two public methods.

The UML showcase the OOP principles in several ways. Each attribute is private, showing the principle of encapsulation, using methods to gain access to and show the data protecting it within each class. Polymorphism is show in both the use of multiple constructors for each class that uses them, with one overriding the other if parameters are used. Association and Composition are show by having several classes composed or associated with each other. The connections between Game, Team, and Player are examples of composition and multiplicity, with the connection between those three classes and GameService also showing both composition and multiplicity. Inheritance is also show with Game, Team, and Player all inheriting from Entity.



## [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** | There are options available to host server on MacOS, but less compared to alternatives. They also tend to be expensive, and only controllable via Apple products. Licensing cost are a concern, in addition to increased hosting cost. MacOS does have increased security over Windows, but roughly similar to Linux. | Hosting on Linux is typically cheaper than alternative, due to the open-source nature, and the fact host can often use older equipment, lowering. Scaling can be cheaper on Linux, due to lower hosting cost. Security on Linux can vary, depending on how well the host maintains their local OS kernel, and how the developer integrates security into their design, but is typically considered very secure. | Hosting on Windows can be easier than other options but comes at a higher cost. There are more options available for Linux, but if the developer is using certain languages or frameworks (such as .Net, MS SQL or Sharepoint), Windows is a better choice. Security can be a concern. Windows is a common target, and due to its closed-source nature security risks can take longer to resolve than Linux. | Mobile Devices are fine to manage applications hosted elsewhere but impractical to host directly on the mobile devices. |
| **Client Side** | Higher initial setup when it comes to hardware, but a lot of development tools are available freely as part of that initial cost. Apple provides several tools themselves. Does require experience and knowledge that is specific to the MacOs platform, which can increase cost and time in development.  There are multiple browsers available for MacOS, with Safari being the default one. Chrome, Edge, Opera, and Firefox all have version available. | Low cost compared to other platforms but does require a specific expertise to develop. However, many, many tools are available for free, and/or open source to help offset cost of the expertise. Linux has a wide variety of internet browsers available, with Chrome being the most popular. Firefox and Opera have versions available, with many others. | Moderate cost and expertise required for development. There is a variety of tools, and with Windows being the most popular OS, more people are more familiar with the Windows OS and development on it. Another thing to consider is many multi-platform development options are more readily available on Windows compared to other platforms. Edge is the default browser, though Chrome, Opera and Firefox have versions, with several alternate as well. | Developing on mobile is all but impossible. In most cases, creating an app that runs on the mobile platforms is preferable to having a web-based page. However, depending on the platform, several different internet browsers. IOS defaults to Safari, but Chrome, Opera, and several others are available. Android has numerous available. With Chrome, Opera, and Firefox being the most popular.  One thing to note is cross-browser support does require extensive testing and the use of a CSS framework such as Foundation or Bootstrap. Validation of the CSS framework and HTML is also a consideration to assist with bug fixes. |
| **Development Tools** | Tools: MacOS Server, XCode, AppCode, Visual Studio, NetBeans, SourceTree, Mac Catalyst, Testflight  Languages: Objective-C, C++, Swift, most common languages are available depending on tools used. | Tools : Visual Studio, Eclipse, git, NetBeans, Aptana, Vim, Bluefish  Languages : C, C++, Java, Python, JavaScript, Shell. Rust  These options can vary widely, and there are many, many more options available. | Tools: Visual Studio , NetBeans, Eclipse, GitHub, Bootstrap, Azure, Atom, Linx, Cloud9, Jira, GitLab, Pycharm  Languages: C#, C, C++, Python, Java, JavaScript, Swift, Ruby, Go, Scala, Ruby, Rust | Tools: Xamarin, Qt, Firebase, XCode, AppCode, Appcelerator, React Native, Android Studio, Eclipse, SourceTree, Visual Studio, Flutter  Languages: Java, Swift, Objective C, JavaScript, Kotlin, C#, C++, Python, HTML5. Ruby, Rust.  The tools suggested depends entirely on targeted platform specifics. |

## 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**: The Gaming Room should use a Linux OS for the development and hosting of their game, Draw It or Lose It. Linux provides the best cost to features option. The cross-platform support that Linux has available for development and the lower cost for hosting help balance learning the lesser-known OS. Ubuntu is the most popular Linux version, with the most community and learning resources available for it as well. There are multiple hosting options for cloud storage via Ubuntu as well, from all the major hosting services available.
2. **Operating Systems Architectures**: Linux Operating Systems are an open-source UNIX system. While all versions have a base similarity around the kernel of the OS, versions vary widely on support and features. While ubuntu is the most popular and widely support, other distribution may provide more specified features for development and maintenance of the game. The open-source nature of the Linux family of OS’s
3. **Storage Management**: Using a cloud host with a Linux host would be the best option for storing the images for the game. While local storage may provide faster loading times, these loading times could be hidden and/or minimized with how the game is designed. Using the NFS file sharing system within Linux can help manage how the game looks for and loads in the images for the game.
4. **Memory Management**: Linux relies on the use of virtual memory, with paging helping the OS manage how to best use the available memory. Virtual memory allows the OS to use part of the storage memory as part of the RAM. Paging allows for the OS to load and store data in sections, keeping track of what data is connected to each other.
5. **Distributed Systems and Networks**: Using a cloud host, some of the pressure of outages is taken off the developer and more in the purview of the host service. The host, using many physical servers, will prevent outages by allowing our game to exist across servers, especially as user number increase, and in the case of server hardware failure. The use of the REST API’s HTTPS methods can help manage the application media across platforms that the game may be played on and help manage the security of the application.
6. **Security**: The security of the game is divided between two considerations: the application itself and the users of the game. Implementing some form of user authentication should be key to providing a secure application. One way to do this would be the implementation of a username and password via the use of the REST API. Linux uses a system of user and file permissions, managing how process read, write, save, and deleting data, keeping the application from accessing things it should not and users from harming the application or hardware it may be running on.