

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 5/17/23 | Michael Kinney | Software Design for Draw It or Lose It |
| 1.1 | 6/3/23 | Michael Kinney | Software Design Evaluation |
| 1.2 | 6/15/23 | Michael Kinney | Software Design 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](#_heading=h.35nkun2)

The Gaming Room is currently looking to develop a web-based game that is compatible with multiple platforms. The goal is to develop this game based on their already existing game, Draw it or Lose it, which is only compatible with android devices. The correct tools must be used to efficiently develop a game that is compatible across multiple platforms, rather than just one.

## Requirements

The Gaming Room has multiple requirements for the development of this game, which include the following:

The game must be compatible across multiple platforms, each game will have the ability to have one or more teams involved, game and team names must be unique and allow users to check whether a name is in use when choosing a name, and lastly only one instance of the game can exist in memory at any time.

## [Design Constraints](#_heading=h.1ksv4uv)

1. Cross-platform compatibility is required for the development of this game. Currently, the game that The Gaming Room offers is only compatible with android. This will require the development team to have to choose specific tools to develop this, such as utilizing frameworks that are compatible across multiple platforms.
2. The game will be required to have one or more teams involved in each game. Developers will need to figure out a way to allow the program to handle users adding more than one team without unexpected responses from the program.
3. The Gaming Room is requesting that team names are unique. Developers will have to be able to design the program to be able to check for team names that are already in use.
4. The game will be required to only allow one instance of the game to exist in memory at any given time. It is possible to achieve this by creating unique identifiers for each instance of each game, team or player, and checking for the unique identifier.
5. The game should consist of four rounds that last one minute each. If the team does not guess the puzzle before time expires, the remaining teams have a 15 second opportunity to solve the puzzle. Due to this requirement, the developers must figure out a way to accurately track the timing of each round. Additionally, developers must be able to render the stock drawing from the library at the correct time each round.
6. Since the game already exists, albeit only for android, the UI for the newly developed application should be similar to the UI that exists for the previous version. This limits creativity for the developers, but also gives a blueprint for the developers to base the new application off.

## [System Architecture View](#_heading=h.44sinio)

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](#_heading=h.2jxsxqh)

In the UML diagram, the Entity class connects the Game, Team and Player classes, which are shown below, and all of which have an arrow pointed at the Entity class. This means that all three of these classes will inherit attributes from the Entity class. This creates efficiency and eliminates redundancy in the code. The GameService class contains all of the game’s complex methods and true functionality. As you can see by the line connecting GameService and Game, there can be 0, or multiple games associated with GameService. This is true for all of the class connections in the bottom row. There can be 0, or multiple teams within each game. Similarly, there can be 0, or multiple players on each team. The ProgramDriver class in the top left contains our main method. This is where we will be able to access and execute all of the classes that we have created. In each class, you can see there are attributes/methods that have a minus (-) next to them, and some have a plus (+) next to them. This is the practice of encapsulation, which is one of the main principles of OOP. The minus symbolizes private attributes/methods and the plus symbolizes public ones. The private attributes/methods cannot be accessed by other objects. This allows you to hide the internal workings of an object, but provide an easy interface for other objects/code to interact with it. This also allows more protected manipulation of each individual object.

**"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](#_heading=h.z337ya)

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 offers a very robust and reliable platform for hosting web applications. The platform they offer for hosting these applications can handle demanding workload and is often very secure. MacOS offers a lot of development tools that create a very efficient ecosystem to develop applications. MacOS development, however, sometimes has compatibility issues with other software, applications and/or operating systems. | One of the major advantages of Linux is that it is open-source and using it would be very cheap. This also leads to another major advantage of flexibility and compatibility with programming languages and frameworks. A downside to Linux, is that there is potentially a larger learning curve than other operating systems, and might be difficult to use for people with no experience. | Windows also offers a very robust and reliable platform for hosting web applications. Unlike Linux, it is not open source and will be more expensive to use to host web applications. Additionally, security is generally a concern with Windows. However, one major upside to Windows is the compatibility it has with other software and applications. | Hosting web applications on mobile devices is not nearly as robust as doing so on the other operating systems mentioned, especially since they have major limitations in terms of processing capabilities, however it has its advantages. Mobile devices are extremely accessible and portable. They are also typically touch screen, which can allow developers to make unique design choices. Despite these, mobile devices are not typically designed for this purpose. |
| **Client Side** | Mac is not open source/free, so there would be costs associated with using MacOS for development. However, Mac generally has a very user-friendly environment. | Since Linux is open source/free, associated costs would be very low, which could make this a preferable OS. Despite this, support/resources for learning how to use it may be more limited than other operating systems and may take more time to learn for people with no experience. | Similar to Mac, costs associated with Windows would be higher due to licensing and other fees, since it is not free/open source. However, Windows is very supported, has many resources available and generally easier to use than an OS like Linux. | Costs would vary and might be a little more unpredictable with mobile devices since there are various mobile operating systems to choose from. A major concern is functional limitations since mobile devices are not optimized for this sort of development. |
| **Development Tools** | Swift would be the most likely used language for Mac development, since it is Apple’s own programming language designed to be used for MacOS. Objective C is another language that is used for MacOS development. HTML, CSS and JavaScript are also used for styling and creating interactive web applications for MacOS. XCode is Apple’s IDE that specifically has tools to aid in MacOS development, so this would be the most ideal IDE to use. | Linux is very flexible and it is common to use all types of programming languages such as Python, C/C++, Java, etc. SQL is a popular query language that is commonly used on Linux to manage databases. HTML, CSS and JavaScript are great for styling and creating interactive interfaces for the applications. A very popular IDE that is used for this is Visual Studio or Visual Studio Code. | C# is a very commonly used programming language with Windows since it was developed by Microsoft. .NET is a framework that is very popular for Windows development and would likely be used as well. HTML, CSS and JavaScript can be used for styling and interactivity similar to the other platforms. Visual Studio and Visual Studio Code are very popular IDE’s to use for this sort of development. | For mobile development, it would depend which mobile operating system you are designing your application for. For iOS, typically Swift or Objective C would be used. For Android, Java is commonly used. XCode is the IDE that is typically used for iOS development, and Android Studio is the IDE used most commonly for Android development. There are ways to only write code once and be able to deploy it on both platforms. React Native is a framework that allows developers to build applications compatible with multiple platforms using JavaScript. It allows the UI components to render natively on each individual OS. Despite this streamlined process, there still may be some platforms specific code that is needed to support cross platform compatibility. |

## 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 seems like a great choice to use for this application due to the low cost and flexibility. It will allow developers to have more tools to use during the development of this application, and will also allow compatibility among various software.
2. **Operating Systems Architectures**: Linux, like any other operating system, acts as an interface between the user of a computer and the resources that the computer uses to carry out functions. It is an open-source operating system with a monolithic kernel architecture, where the kernel contains all the essential functionality. It follows a modular design which allows for addition and removal of kernel functionality through kernel modules. Linux is extremely flexible since it offers a wide range of file systems and a comprehensive network stack. It uses user permission and encryption to maintain a very secure platform to host a web application.
3. **Storage Management**: Cloud based storage seems to be the most ideal for a web application, since hardware limitations will not cause any issues for memory management by using cloud storage.
4. **Memory Management**: Linux uses virtual memory, which allows each process to have its own address space ensuring processes are isolated from one another. This creates a much more efficient memory management system. Additionally, during the development of this application, unique identifiers can be used as a technique to allow only one instance of a game to exist in memory at any time.
5. **Distributed Systems and Networks**: Using client-server architecture to design this web based game could be an ideal solution to seamlessly allow Draw It or Lose It to communicate between various platforms. The game can actually run on a central server, and multiple devices connect to the server using their device. All devices will need to be connected to a network such as WiFi to allow communication between the server and devices. Of course, disconnecting from the network will cause issues during the game, and the game needs to be designed to detect and handle these issues, even if just in the form of an error message. Ideally, you would like to deploy a method for allowing users to reconnect to their existing game, if possible.
6. **Security**: Requiring all users to use multiple factor authentication to login to their accounts is one technique that can be used to mitigate security issues. Additionally, implementation of encryption algorithms is another advanced security measure that can be used to protect user information. Fortunately, Linux provides many encryption libraries that can be utilized to achieve this. Lastly, constantly monitoring the security system in addition to deploying upgrades and patches to the system will be yet another technique to ensure security for the client.