

Draw it or Lose it

# **CS 230 Project Software Design Template**

Version 3.0

## 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 |
| --- | --- | --- | --- |
| 1.0 | 01/27/2023 | Daulton Truett | Wrote the executive summary which consisted of identifying the software design problems, I proposed a solution to this issue, I identified the design constraints with this application, and I described the provided UML class diagram domain model. |
| 2.0 | 2/05/2023 | Daulton Truett | Added descriptions to the development requirements under the Evaluations section. |
| 3.0 | 02/23/2023 | Daulton Truett | Added final recommendations for “Draw it or Lose it” to determine an operating platform. Discussed the purpose for choosing the operating platform and highlighted the various aspects of the platform. |

**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)

Our client, The Gaming Room, currently has a game called “Draw it or Lose it” that runs on Android OS and is wanting to develop it to run as a web-based game. The game needs to allow for a minimum of one or more teams playing with each team consisting of multiple players. There can only be one instance of the game in memory at once. The game will need to be able to allow the players to create unique team names and check whether a team name they want is already in use. In order to help aid this process, the game should create unique IDs for each instance of a game, team name, or player.

The solution I propose is to develop the game on either Windows, MacOS, or Linux, and host the game on a server online that users will be able to connect to from the host systems. The game can then be run within a virtual environment, so regardless of the users host operating system, the game will run natively on our host OS of choice and will in turn provide cross platform cooperation.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

Currently, this application is only being run on Android OS and therefor creates an issue for the development of the application to run on a web-based system. Application development designed for a particular operating system, specifically for a smartphone OS, are designed to run specifically for that environment. The Android OS and other various smartphone OS applications tend to run simpler program code as those operating systems are less complex than many other operating systems found on PCs and what the game will encounter when ran in a web-based system. A design constraint lies with enabling the game to run on various operating systems across various platforms. For example, the game will need to run properly on Mac OS, Windows, and Linux, at least, as well as allow these users to play together across these different platforms. Designing the game to operate among various operating systems and on the internet also requires the use of the game communicating across a network and adapting to these varying environments found within these different operating systems.

## [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 UML class diagram below provides a visual overview of how the program is tied together and provides its functionality. The ProgramDriver class is the “main” class that “drives” the program. This class is where we will find the code that runs the entire program and invokes all other classes with the program to operate the game as it is intended. The arrow pointing to the right from the ProgramDriver is indicating that the ProgramDriver uses the SingletonTester class, but there is no other relationship or need for this class beyond this purpose as this class does not interact with any other class in the program. The Entity class is the Parent class to the GameService class, Game class, Team class, and Player class. These four children classes inherit the attributes from their Parent Entity class and are all associated with each other as indicated by the line with the “0...\*” interconnecting all of them. This means that there is an association between these children classes that can be defined as 0 or many, with many being an undefined and uncapped number. For example, the Game class can have 0 teams to a game, or 50 teams to a game, and a team can be composed of 0 players, or 100 players, to make a game.

The object-oriented principles defined in this UML class diagram demonstrate inheritance amongst the Parent Entity class and its children classes. There is use of Portability with the Entity class, which enables the class to be used as more than a single object and reduces redundant code. The GameService class allows for only one game to be in memory at one time with the use of the service object. GameService allows us to check the player’s name, game name, and team name to ensure no name is used more than once for any of these variables. The GameService class also allows us to use nextTeamId, nextPlayerId, and nextGameId, which follows the software requirements of only allowing one game, player and team name to have a unique ID since it increments the ID after every new game, team or player is created.

**"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** | MacOS can be used to run a server through the built in Apache server and using the command line terminal. MacOS is a secure platform and rarely experiences security issues. MacOS requires that its software be run on Apple hardware and isn’t available to use on other systems outside of VMs. Since MacOS is run on Apple hardware, upgrading memory and storage is not an option as they are closed systems, so scaling can become an issue as user count increases. Mac is also closed source which requires purchasing their software from Apple. MacOS previously offered MacOS server but is now discontinued as of 2021. | Linux can be used as a server to host the web-app with an Apache server and the command line terminal. Linux is open-source and has many different versions of the platform. Since it is open source it is free to use and can be used on almost all hardware. Linux has lots of support since it is open-source and is also able to be scaled dramatically as user count increases due to its versatility. | Windows is available to use as a sever with its “Windows server” but requires a high licensing cost. Windows is closed source so the software must be purchased from Microsoft. Windows is relatively secure and offers robust security features. It is an open system so the hardware can be upgraded to provide scaling as user count increases. | Mobile devices are able to be used as a server to host apps but are not a popular option currently. The problem with mobile devices is their limited CPU, memory, and storage, so scaling as user count increases will be difficult. Mobile devices are usually Android or iOS which means that licensing will be required. |
| **Client Side** | Safari is the most commonly used web-browser for MacOS, so the application must support this browser. The cost to develop on Mac and support multiple clients should not be too expensive and should not take too much time. Mac requires expertise in the Swift programming language and allows for Windows and Linux applications to run within its environment. | Linux supports many modern web-browsers. The cost to develop on Linux should be low but the time may take longer than other platforms as there are not as many developers with expertise in the Linux environment. However, Linux can support multiple clients from various platforms and multiple languages can be used in development. | Microsoft edge is the browser that comes installed with Windows, but many more common browsers are supported. Windows has many developers that are familiar with its platform so time and cost will be low. But the issue that it will face is supporting Mac users. Windows will need to run Mac users in a VM since these users will be designed to run on MacOS which runs on Apple hardware. Many languages can be used for development of the app. | Cost and time will depend on Mobile devices should both be low as Mobile device development is very popular today. Expertise will vary as developing for Android and iOS are very different and use very different technologies. iOS will use Safari web browsers and require expertise in Swift. Android will use many common web browsers and supports building the app with Java, C++, and Kotlin. |
| **Development Tools** | Programming languages:   * Swift   IDE:   * Xcode * Visual Studio Code | Programming languages:   * Python * Java * C++   IDE:   * Eclipse * Visual Studio Code * Atom | Programming languages:   * Python * Java * C++   IDE:   * Eclipse * Visual Studio Code * Atom * Visual Studio | iOS:   * Swift   Android:   * C++ * Java * Kotlin |

## 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 operating platform I am recommending as a server for “Draw It or Lose It” is Linux. Since Linux is open source the cost is free, and many options are available for initial configuration such as the distribution of the Linux kernel, with a popular distribution choice being Ubuntu. Linux is scalable because it can be upgraded with minimal effort and cost, it can operate on any available hardware, and Linux applications can be easily deployed on both Windows and Mac. Linux has a huge community surrounding it with many active developers and huge pool of support.
2. **Operating Systems Architectures**: Linux utilizes a system architecture that is composed of layers. The layers consist of the physical hardware, the system libraries, the Linux Kernel, the Shell, and user applications. The physical hardware includes the CPU, storage (such as HDD or SSD), memory (RAM), GPU, and I/O devices. Applications form the highest layer and are typically what the users engage with on the operating system. The shell is how the user interacts with the operating system as a whole and Linux now commonly includes both a GUI and command line interface. As the user makes requests, the system libraries connect the user operating in the shell to the Linux Kernel. The Linux Kernel then communicates with the hardware and provides instructions on what operations to perform. These operations include both user generated requests and system call requests that Linux makes in order to run the system.
3. **Storage Management**: The storage management system I recommend for “Draw it or Lose it” on this operating platform is a MySQL server. MySQL is an open-source relational database management system that works well with the Linux operating platform. A MySQL server can be scaled as needed to an unlimited storage potential as needed, it offers a high level of security with use of SSL and is well suited for web applications.
4. **Memory Management**: Linux utilizes paging and swapping for its memory management techniques. Paging is used first to load processes of the game into memory for execution. This can occur as users begin a new game and various input and output request are made. Once the memory is exhausted from available memory, Linux uses swapping to swap out the exhausted pages back onto the disc to create a continuous stream of available memory.
5. **Distributed Systems and Networks**: “Draw it or Lose it” is game that requires users to communicate across various operating platforms. This is accomplished using HTTP(S) over TCP/IP. All operating platforms utilize TCP/IP to access the internet, and HTTP has been a longstanding protocol to access various webpages over TCP/IP. It is commonplace, and usually best practice, to now use HTTPS as It provides the same service to access web pages over TCP/IP but uses SSL/TLS to ensure a secure connection. The host operating platform, which we chose as Linux, will host the server that holds the web application data within a MySQL server that users may access with HTTPS and TCP/IP. In this model, If the server is down on our host Linux server, then users will be unable to access the data required to run the web application, but server outages can be minimized with redundant measures to ensure the sever does not have extended downtime.
6. **Security**: Linux is a very secure operating platform as it utilizes many modern security features in demand amongst operating platforms today. Linux features user security and data encryption in the form of password hashing, file encryption, automatic security updates, SSL/TLS for TCP/IP, and stack/heap protection. Linux uses partitions to organize data within the operating platform and these partitions can be encrypted to ensure data security. With regard to hosting the web application server, Linux features SSL/TLS to ensure a secure connection between both client and server. We are utilizing a MySQL server to host the web application data and user data which emphasizes secure data storage and user retrieval with SSL.