

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 |
| --- | --- | --- | --- |
| 1.0 | 01/21/2023 | Christopher R Anderson | Initial write-up for Draw It or Lose It multiple platform web-based game |

**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 company is planning to transform their Draw It or Lose It gaming app into a web-based multi-player application. This game slowly renders an image from the in-game library for 30 seconds and allows a team 1 minute per round to guess the image with 4 rounds per game, if the team guesses incorrectly then the remaining teams are allowed 15 seconds to present their guesses.

## Requirements

Each of the games, teams and players is required to have a unique name and identifier. There shall also be a minimum of 1 team per game and 2 players per team. There can only be one instance of a game, i.e. no duplicates of a game instance or shared game names/identifiers. The Draw It or Lose It game will be online and multiplayer, therefore this web-based game will require servers active 24/7 to host the various games taking place along with the software infrastructure to allow players to connect with each other to play on the servers.

## [Design Constraints](#_2et92p0)

Possibly the biggest business constraints on the development of the Draw It or Lose It game will be server availability and the deployment style of this game. Server availability is vital to the growth and value of the user base since the game needs servers to host the games as the teams compete and a lack of server space or weak network latency will limit the number of clients purchasing and playing this game. The web-based nature of this game will also create constraints on the size of teams, number of teams per game and the number of games active on a single server, these factors will depend on the predictable network latency. The deployment style will also generate different restrictions should The Gaming Room decide to publish Draw It or Lose It as a flash game introduced through a gaming website, such as Armor Games, or if this game will be published independently through the Google Play store, Steam store or another 3rd party gaming store. The deployment style will also determine the monetization of this game since flash gaming sites will likely pay you a percentage of the advertisement revenue earned from ads on your game along with income from in-game upgrades, whereas if the company publishes Draw It or Lose It independently through a game store then the company will be paid for each copy of Draw It or Lose It sold, or paid a monthly membership fee.

## [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)

In the UML below The ProgramDriver class contains the public main function and has a uses relation with the SingletonTester class that contains the public testSingleton function. The Entity class acts as the parent class to the Game, Team and Player classes and contains the private long id and private String name variables as well as a private default constructor, a public constructor with long and String parameters, a public getId function returning a long, a public getName function returning a String and a public toString function returning a String. The Player class is a child class of Entity, this class also has a 0 to many Association relationship with the Team class and contains a public construct with a long and a String for parameters and a toString function returning a String. The Team class is a child class of Entity, includes 0 to many instances of Player, has a 0 to many Association relationship with the Game class, contains a private list of Player objects, a public constructor with a long and a String for parameters, a public addPlayer function returning as a Player object with a String for a parameter, and a toString function returning a String. The Game class acts as a child class of Entity, this class includes 0 to many instances of Team, this class has a 0 to many Association relationship with GameService class, Game class contains a private List of Team objects, a public constructor with a long and a String for parameters, a public addTeam function returning a Team object with a String parameter and a toString function returning a String. The final class is named GameService, it includes 0 to many instances of the Game class, it contains a private List of Game objects, a private long for the next Game Id, a private long for the next Player Id, a private long for the next Team Id, private GameService object to track single instance of GameService class, a private constructor with no parameters, a public gentInstance function returning GameService, a public addGame function returning Game object with a String parameter, a public getGame function returning Game object with long parameter, a public getGame function returning Game with String parameter, a public getGameCount returning an integer, a public getNextPlayerId function returning a long, and a public getNextTeamId function returning a long.

In this UML diagram, the principle of Inheritance is present between the Entity class and its child classes the Player, Team and Game classes. The Encapsulation principle is also active since there are private members in the Entity class that will be inaccessible to Entity’s child classes and external functions; the private members of the other classes will also be encapsulated from external functions and their sibling classes. There are also three 0 to many Association relations expressed between the GameService, Game, Team and Player classes.

**"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** | There are many aspects to consider when developing a MacOS server. First, it is easy to convert a standard MacOS machine into a server. Second, Apple software is constantly updated, and the OS does include many system-native features. But most Mac machines are not as energy efficient and will add clutter to the workplace with their own keyboard, mouse and screen. When developing a MacOS server there are several aspects to consider. | There are several features to appreciate when developing a Linux server. The Linux servers are extremely stable with long-standing protocols and software kernels, this will present the most reliable server environment currently available. Furthermore, the Linux OS is not as exploitable as Windows OS that is known to have constant security concerns. But the Linux OS can be slower than other OSs to develop compatibility into software package that can cause restrictions when entering the marketplace. When developing a Linux server there are several features to consider. | There are a variety of characteristics to understand about Windows OS before implementing a Windows server. The Windows servers are less expensive to purchase and operate than MacOS servers. Also, Windows servers have a more refined client-server infrastructure than Linux. But there are more security concerns in Windows servers than in the MacOS and Linux OS servers. Before implementing a Windows OS server it’s paramount to understand the characteristics of the Windows OS. | There are several aspects to consider when developing mobile device servers. Mobile Device servers offer the best network performance, improved security, and lower costs for network management and maintenance than the standard OS servers. But developing mobile applications will require technical expertise specific to mobile development since the mobile platforms are designed differently than standard OSs. When developing mobile device servers there are a few characteristics of mobile devices to understands. |
| **Client Side** | When implementing the client side of a MacOS server it’s important to consider the costs and software requirements. There are a variety of MacOS development tools available that will allow for ease of development. But costs will be higher than competitors to purchase and run the MacOS machines. The time and expertise necessary will be lower due to the familiarity of the MacOS with developers. It should also be expected that greater hardware requirements will be necessary for a MacOS server. The costs and software requirements are important considerations when developing a MacOS server. | When implementing the client side of a Linux server it’s vital to consider the costs and software requirements. There are also a variety of opensource and licensed development tools for Linux OS so costs will vary upon the development package. Since the Linux system does not offer many designer tools, the development of applications will be slower and more costly with more time directly programming the software and GUIs. The Linux OS servers will have lower hardware requirements. The costs and software/hardware requirements of the Linux OS are vital considerations when developing a Linux server. | When implementing the client side of a Windows server it’s paramount to consider the costs and software requirements. The Windows platform offers excellent scalability and versatility for multi-platform applications that plan to increase their user-base. The Windows OS is the easiest server platform to configure for multiple client types making it more cost effective for implementation than its competitors. But the Windows OS does include greater costs for security vulnerability management. This means that hardware requires will usually be average to lower than average compared to that of competitors. Software requirements and costs are paramount considerations when implementing the client side of Windows OS server. | When implementing the client side of a mobile device server developers must consider costs and device requirements. There are numerous opensource and licensed development tools available for iOS and Android that will allow for hasty development of applications. The cost of development will vary depending on the scale of the application due to the level of device specific expertise required. Due to the nature of mobile device OSs, there will be lower hardware requirements, since the majority of the software will be run on non-local servers and the mobile version of the application will likely be simplified to improve operation on mobile devices that are also simplified and weaker than standard OS machines. Developers must consider costs and device requirements when implementing the client side of a mobile device server. |
| **Development Tools** | When selecting development tools, it is important to choose an IDE and tools for the MacOS framework. The IDEs and tools suggested for MacOS are FlexiHub, Xcode, NetBeans.IDE, Visual Studio, RapidAPI and Hype. The IDE and tools will be determined based on budget and project requirements at the time of implementation. The software languages best for implementation in MaOS are JavaScript, Python, Java, TypeScript, and Go. It is vital to select the proper IDE and tools for developing in the MacOS framework. | When selecting development tools, it is vital to choose an IDE and tools for the Linux framework. The IDEs and tools advised for Linux are Sublime, Atom, and Visual Studio. The software languages recommended for development in Linux OS are C/C++, Java, Python, and Ruby. It is important to select the proper IDE and tools for developing in the Linux OS framework. | When selecting development tools, it is paramount to choose an IDE and tools for the Windows OS framework. The IDEs and tools suggested for Windows OS are AppBuilder, Atom, Eclipse, Visual Studio, Qt Creator, and PyCharm. When deciding between these tools developers will compare open budgets and deployment requirements. The recommended software languages for developing a Windows OS server are JavaScript, Python, Java, Ruby, and C++. The selection of the proper IDE and development tools is paramount in the Windows OS framework. | When selecting development tools, it is key to choose an IDE for Mobile Device frameworks. The IDEs and tools recommended for mobile device frameworks are Android Studio, Eclipse, Visual Studio, DroidScript, Xcode, AppCode, Android Virtual Device, and Buddy.  The best software languages for developing multi-platform mobile applications are C# with Xamarin and JavaScript with React Native since the Xamarin and React Native platforms allow applications to access both iOS and Android software development kits. It is key to determine the best a IDE and tools when developing for multiple Mobile Device frameworks. |

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**:

I recommend the Windows OS given the versatility of the system and the abundance of

developer tools. These characteristics will be advantageous because developers will be able to download code libraries and GUI templates specific to implementation in Linux, iOS, MacOS, Windows and Android environments. The versatility of the Windows OS also shows when determining the project’s budget since the opensource or basic development tools can total less than $100.00 per software license or for more advanced IBM development tools the budget can be over $1000.00 per software license. The versatility of the Windows OS and the abundance of developer tools available makes this OS ideal for Draw It or Lose It due to the scope of this project.

1. **Operating Systems Architectures**:

In the Windows OS, formally known as Windows NT, there is a hybrid kernel

implementation. This architecture is organized into two primary sections, the simple kernel where applications and user input are processed and the hardware abstraction layer where drivers, processes, etc. directly manipulate the device’s hardware. Further, the simple kernel is composed of two layers, the user mode layer where OS environment subsystems operate as well as the kernel mode layer where executive services, i.e. I/O manager, Inter-Process Communication manager, and Process manager, but also kernel drivers, the microkernel and the hardware abstract layer are operating. A hybrid kernel architecture has been utilized to construct the Windows OS, formally known as Windows NT.

1. **Storage Management**:

When determining the best storage management system, I avoid relying heavily on

company run servers to maintain user data. I compensate for the lack of remote server storage space by storing application data, user account data and team data directly on the user’s device with my goal of following a peer-to-peer pattern with some Backend as a Service characteristics. These characteristics will show as peer devices may store the same team data to avoid unnecessary usage of server space while the servers will only act as temporary hosts processing the game(s) and accompanying data, returning data to user devices to be stored and erasing game data on the server upon completion of the game. In my storage management system users’ devices will be responsible for maintaining a user’s individual data and team data that will alleviate stress from the servers that will only need to host active game sessions.

1. **Memory Management**:

In Windows NT memory is managed by the memory manager, a subservice of kernel

mode. The microkernel receives requests from the memory manager, an executive service, to allocate or deallocate virtual memory and dynamic memory. The memory manager also coordinates shared memory, drivers and copy-on-write. These functions of the kernel mode’s memory manager will be invaluable upon execution of the Draw It or Lose It application, since RAM will need to be allocated at the start up of this online game then it will require an additional memory allocation upon entering a game session and finally will need to deallocate some memory upon exiting a game session. Memory will need to be shared between the application and network processes; the various drivers active in the Windows NT system will also be necessary to allow for the user mode operating environment. The memory manager performs a vital role in the kernel mode of Windows NT.

1. **Distributed Systems and Networks**:

The Draw It or Lose It app will be able to communicate between the various OS

platforms due to the flexibility of Windows NT’s distributed systems and network. When

distributing software between client devices and remote servers Windows NT provides a variety

of options on how to delegate the responsibilities of the software package. For Draw It or Lose It, this means that the client devices, with various OS’s, will take responsibility for UI and network tasks related to connecting with the Windows NT server, where the game sessions will be processed. When considering the dependencies built into distributed systems, I would advise that the server have several network checks to ensure that all active team members are continuously connected so that there are no active game sessions that are unoccupied by connected clients, network checks to find sources to network connectively loss e.g., if a user is a source of excessive network lag in an active game session then the server should disconnect this user from the game session, and there should also be network checks to ensure users are not entering active game sessions on remote servers that inadequate network ping. The versatility of Windows NT’s distributed systems and network will give Draw It or Lose It’s servers superior communication abilities between the various client OS platforms.

1. **Security**:

There are a variety of security features available to protect data transferred between

a variety of OS platform clients to a Windows NT server. When dealing with network communication between a client and server security measures such as authentication and encryption are commonly implemented as a Public Key Infrastructure. In the PKI, the server will act as a certification authority that will issue a new encrypted binary certificate to any new application client this is done to affirm the client’s identity and this certification will bind to the client’s device. The PKI security system can be used with multiple platforms, so long as the platform includes a version of Windows NT’s Configuration Manager with Microsoft Authenticated Session template and MacOS clients will additionally need a Microsoft Workstation Authentication template then all OS’s will easily synchronize processing on the Windows NT server. In the instance that The Gaming Room determines that the Draw It or Lose It application does not require the two key encryption of PKI then there are also other security measures for multi-platform network-dependent applications such as single key cross-platform cryptography in the .NET framework such as the Advanced Encryption Standard (AES) cipher algorithm in the Cipher Block Chaining (CBC) mode that recognized by Windows NT, Linux, macOS, Android and iOS. The Windows NT system provides a wide variety of cross-platform security measures that offer various levels of protection to data that is transferred between a Linux, macOS, iOS or Android client and the Windows NT server.