

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 03/18/2023 | Connie Knupp | Filled in the following sections: Executive Summary, Restraints, Design Constraints, and Domain Model. |
| 1.1 | 04/02/2023 | Connie Knupp | Completed the Evaluation section. |
| 1.2 | 04/16/2023 | Connie Knupp | Completed the Recommendation section. |

**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 requests assistance establishing an environment that will expand the game “Draw It or Lose It” from the current Android application into a web-based game capable of operating on multiple platforms. This is best accomplished using a server-style infrastructure, this configuration utilizes multithreading well and will allow for many users on different platforms. Memory allocation will be needed to house the centralized data. Clients can then interact with the central network; this allows for shared information and many users. Run-time can be affected in this set-up by network internet speed and the number of users, as such we recommend investing in a connection with a larger bandwidth. Specialized staff will be needed to maintain the network and server.

## Requirements

The web-based game must be capable of operating on multiple platforms. The names of the players, teams, and games must be unique. Only one instance of each game can exist in memory at any given time but there should be able to be many games being played at once. Runtime must be manageable and not prone to lagging. Memory must be sizable enough to accommodate games. Must be scalable in case the game rapidly gains users.

## [Design Constraints](#_2et92p0)

* Framework compatible with existing servers and websites.
* Framework compatible with multiple versions of iOS.
* Framework compatible with multiple versions of Android.
* Must stay within the budget set by Gaming Room.
* Memory must be managed and kept to a reasonable amount of data.
* Ability to run many games simultaneously.
* There can be multiple players assigned to each team.
* A single instance of the game can exist in memory at a given time.
* The player, team, and game names must be unique; therefore, users must check whether a name is already in use when choosing a name.

## [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. Memory allocation will be needed to house the centralized data.

## [Domain Model](#_8h2ehzxfam4o)

The classes: Program Driver, SingletonTester, Entity, Game, Team, Player, and GameService are all contained within the gamingroom.com package. The ProgramDriver class has an association with the SingletonTester class, it is one-directional, and the driver uses the SingletonTester. The Team, Game, and Player classes all inherit from the Entity class. The GameService is associated with the Game class which in turn is associated with the Team class. Relationships continue as the Team class is associated with the Player class. Object-oriented programming is the application of four principles: portability, inheritance, encapsulation, and polymorphism. Each of those principles are demonstrated in the UML diagram below.  There are separate classes for each object, demonstrating portability or modularity. Inheritance is illustrated as the classes Player, Team, and Game all inherit from the Entity class. The attributes and methods of Entity are replicated in those subclasses. Encapsulation is best illustrated by the private attributes of each class; other classes can only interact with those variables by using public methods. For example, the ProgramDriver must use the addGame function to create a game. Polymorphism is apparent in how an instance of a player or game, would also be an instance of an Entity, one object existing in multiple forms.

**"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** | An advantage of hosting a web-based server application on Mac is that Mac uses Unix-based languages and can work similarly to Linux. Disadvantages are that it requires Apple hardware and requires propriety software owned by Apple increasing cost. Utilizes a graphical user interface which will use more resources. The biggest disadvantage is Apple discontinued Mac OS Server in April 2022. As a result of discontinued support cost for using a Mac-based server would be higher as the Gaming Room would have to employ staff to provide support internally. | Linux is open-sourced and does not require a license because of this it has a lower start-up cost. It can incur more cost for support though as support has to come from staff trained and experienced with hosting via Linux either in-house or maintained with a subscription. Additionally, Linux-savvy software developers are in higher demand and are paid slightly more than their peers. This may increase the cost of support over years. The scripting used in Linux lends great flexibility in the server although the lack of a GUI means administrators must use the command line. Consider using a cloud-based server setup as it can come with support and is cost-effective at start-up until a certain tipping point for size and workload. | Hosting a gaming server on Windows will have a higher upfront cost for licensing, approximately $6155. However, Microsoft has better support and the long-term cost can even out. Also, since the Gaming Room started this game on Android it does not already use Linux or be predisposed to the Apple framework. Windows servers and software generally have more bells and whistles than open-sourced software allowing for a more polished look and enhancing the users’ experience. Windows hosting utilizes a GUI which may make it easier for a less experienced developer to use. Consider using a cloud-based server setup that offers the same benefits for Windows as Linux. | Hosting the server side on a mobile device is not recommended as mobile devices have less reliable internet access, smaller bandwidths, and the IP address can change periodically. Hosting a web-based server requires a fixed IP address so you would need to install a Dynamic DSN app to update the IP addresses if/when it changes.  This still leaves a less reliable internet connection. Also, mobile devices have a smaller bandwidth and will not have the same scalability as hosting on a more powerful device. |
| **Client Side** | Apple discontinued Mac OS Server support in April 2022 this means as devices and software on the client’s side are updated it will require more staff and resources from the Gaming Room to keep it functioning. As Mac is less commonly used it requires staff that are experts in its particulars. These things mean it will have a higher cost to maintain than using Linux or Windows. | Creating a web-based application for Draw It or Lose It means it can easily be used on any OS using Linux. Most computers, regardless of the operating system, have some sort of internet browser and will have no trouble running the HTML code of the front-end client. This would require a front-end web developer to be on staff to help maintain functionality and update the program as needed. | Creating a web-based application for Draw It or Lose It means it can easily be used on any OS using Windows. Most computers, regardless of the operating system, have some sort of internet browser and will have no trouble running the HTML code of the front-end client. This would require a front-end web developer to be on staff to help maintain functionality and update the program as needed. | Mobile devices typically include web browsers so they should be able to access a web-based application without additional software requirements. If the Gaming Room wants to improve the mobile device interface for the client side on a separate app it can use one of several languages. The most popular languages for Android apps are Java and Kotlin. This would incur additional costs as it requires its client software to be developed. Creating gaming apps for multiple platforms can cost anywhere from$20,000-$250,000 depending on the complexity of the game and application. |
| **Development Tools** | Mac encourages the use of the Swift language although C#, C++, Java, and Python are all popular choices that can be used with the appropriate compiler. The best IDE to use is based on the programming language chosen but Visual Studios is one of the more popular choices and can work on Mac with many languages. | C#, C++, and Java are common choices for servers using Linux OS. My research depicts Visual Studios for Unity as a good choice for IDE you can use many languages in Visual Studios. Eclipse can also do these languages and more so it is a valid choice as well. Additionally, packages like Jersey client builder can be utilized to decrease development time. Typically, the development team would consist of a team working on the back-end Linux server setup as well as a team working on the front-end client/web application setup. | C#, C++, and Java are also good choices for servers using Windows OS. My research depicts Visual Studios for Unity as a good choice for IDE you can use many languages in Visual Studios. Eclipse can also do these languages and more so it is a valid choice as well. Additionally, packages like Jersey client builder can be utilized to decrease development time. These languages work well as they work with both Linux and Windows operating systems. Typically, the development team would consist of a team working on the back-end Windows server setup as well as a team working on the front-end client/web application setup | Hosting on a mobile device can be done with additional tools but would not be effective for large-scale use or an app that expects a high volume of users. If attempting this method additional tools are required each tool requires additional expertise to maintain and comes with additional cost. It would be more cost-effective to use a shared hosting provider (smaller application/user base), VPS (medium application/user base), or Cloud hosting provider (large application/user base). The cost of these services ranges from $4 to $1000 a month depending on the size of the database and workload requirements. |

## 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 best two choices for operating platforms to run the game Draw It or Lose It are the Windows operating system or the Linux operating system. As you could see from the evaluation both systems would work well however I am recommending the Windows operating system. If we were creating the game from scratch without a prior Android game and lacked data about how popular the game may become, I would recommend Linux. This is because Linux has lower startup costs and would be less risky if the game were not popular. Since the Gaming Room has already created this game on Android and found it successful the best choice is Windows. Windows may have a higher startup cost, as a result of license fees but it will come with better support and lower cost over time. As the game will be a web-based application it can easily be used on other computing environments.
2. **Operating Systems Architectures**: Windows operating systems use a layered design with user and kernel modes. The user mode is limited in which resources it has access to while the kernel mode has open access to the system resources and files. This system is packet-driven and the layers interact via process requests made by packets. This abstraction helps to protect critical files from corruption. This layered approach will also help with security in the Draw It or Lose It server. The game will utilize a server-client architecture that will be web-based. It will allow a wide variety of client platforms to interact with the server using their built-in browsers.
3. **Storage Management**: Storage refers to how permanent files and data are kept this includes the user profiles, the stock image files the game will need to render, and of course, the code for the game itself. Due to the variable size of storage needed to be based on the number of users, and whether the image stock is expanded, I recommend using a cloud-based storage program. One benefit of cloud-based storage is it is more cost-effective as you only pay for the storage being used. It also eliminates the need to maintain a physical space for the storage unit, further decreasing cost. In addition, the responsibility of backing up the files stored and maintaining the space falls on the provider of the cloud storage and not on the clients.
4. **Memory Management**: In a Windows operating system the user and kernel modes each have access to equal parts of the memory. In a 32-bit system, Windows allocates virtual address space for each 32-bit process. The virtual address space does not represent the actual location and the address of each process is private and can only be accessed by other processes if shared. This improves security and decreases the likelihood of one process interfering with another process. This also supports multiple processes occurring at the same time increasing the efficiency of the system. For Draw It or Lose It the system should only copy the data for the images being used in the current game to memory allowing for quick access without using all the memory space.
5. **Distributed Systems and Networks**: The Gaming Room anticipates Draw It or Lose It to work on multiple platforms. This is accomplished using the client-server architecture. The server running on a cloud-based Windows system would allow for users to access the game from many devices. The system would be available if the user had access to the internet on a device with a web browser, as most devices come with built-in browsers this means that it will be accessible to most devices. The benefit of a cloud-based Windows server is that all outages would be the responsibility of the cloud provider to fix allowing The Gaming Room to focus solely on running and improving the game.
6. **Security**: Security must be built into the Draw It or Lose It game from the onset of coding. This is cleanly done using DropWizard and Jersey to configure roles and regulate the permissions and access for each role. These tools work well with web-based applications and are formatted to work with a Windows-based operating system. This allows for creating a REST application that authenticates users before executing any command. Any command that is not accessible for that role is then blocked from execution. Using two-factor authentication increases security by making it harder to hack into a user’s account. Tips can be added to login screens that remind users not to give out secure information that is commonly targeted by phishing agents.