

The Gaming Room by Tabitha Pawlowski

# **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 | 03/22/24 | Tabitha Pawlowski | This is the software design document for the Gaming Room application |

**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)

We will create a web-based version of the game Draw It or Lose It. Each game will have four rounds lasting of one minute each. Each team, which will consist of several players, will have 30 seconds to determine their image. If they are unable to guess the image correctly, the other team will have 15 seconds to answer for them. We must ensure that the game is able to have more than one team involved with multiple players assigned to each team. We must also make sure each game and team name are unique so that there are no multiples, and that only one instance of the game is able to exist at a time.

## Requirements

*The game must have the ability to have one or more teams involved at a single time. Each team will also have multiple players. Each game and team name must be unique with no duplicates so that users can check whether a name is in use already or not. Additionally, only one instance of the game can exist in memory at any given time.*

## [Design Constraints](#_2et92p0)

One design constraint is that the game must have the ability to have one or more teams involved. This means we will need to ask the user at the beginning of the game how many teams there will be. If there is more than one, we will also need to keep track of which players are on which team and whose turn it is currently.

Additionally, each team will have multiple players assigned to it. We will need to keep track of each player and which team they are on, as well as how many players are on each team. It will also be beneficial to have the same amount of players on each team.

Another design constraint is that each game and team name must be unique. To do this, we must keep track of the current game and team names. We must also make sure to check any given user input against these values to ensure they are valid. If the names already exist, then we will need to output an error message and ask the user to input another name.

Lastly, a design constraint that must be implemented is ensuring that only one instance of the game can exist in memory at a single time. This will be done by creating unique identifiers for every instance of a game, team, and player. Any user input will be checked against this and if it goes against the rules, an error message will need to be output.

## [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 SingletonTester class is used to ensure that only one instance of the game is accessible my memory at a single time. This means that there is only game instance at a time, demonstrating OOP principles while also fulfilling the design constraints. The Entity class holds the object attributes for the program. Here is where all the attributes for the objects will be held. The player class holds data for each of the players. The team class has attributes for a list of player objects and holds methods including the team names and ids and for adding players to the team. The game class also includes a list for the teams and has methods for storing the game id and name and adding the team’s name. The GameService class is where most of the variables and methods are written. There are attributes for the game list, next game id, player id, and team id, as well as the Game Service constructor. There are also functions for getting the current instance of GameService, as well as adding and getting a game using the name, id, and playerId as parameters. The ProgramDriver class is the class where all of these classes and functions are executed. A parent class is the Entity class and the Game, Team, and Player classes are the child classes that inherit from the Entity class. Dividing the attributes and methods into different classes helps implement OOP principles and also makes it easier to fulfill all of the design and software requirements.**

**"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 several advantages of using Mac for hosting web-based software applications. Mac has a simplistic and easy to navigate look to it. It works across many Apple devices and has many built-in development tools that are useful when creating and hosting a web app. It is also supported by Adobe, which helps with development. There are also several disadvantages, though, including the lack of compatibility with internet explorer, package managers not compatible across all devices, and it being a closed system. In addition, the development platforms may take a long time to set up in the MacOS. | Some advantages of Linux for hosting a web-based software are that it is open sourced, comes with a variety of compatible tools and languages, and it comes with a variety of package managers that help make it easy to update and manage your apps. It’s also easy to customize the system to fit your needs and is compatible with other devices. Some disadvantages of Linux for a web-based application include the fact that it looks and functions much differently than other operating systems, which requires developers to become familiar with it before any work can begin. In addition, there are fewer users and a lack of general expertise on this OS. | Windows has several advantages, including the ability to create interactive programs, ability to easily update your app, and the ability to scale up to meet the needs of your clientele. Windows provides a familiar, user-friendly experience and is compatible with other Windows devices. In addition, it is easy to make changes to several web applications from a single control panel. Disadvantages include the fact that it is a more costly operating system than the other options and can be seen as not as secure and reliable. There can also be issues when trying to run multiple processes at a time, which can lead to crashes if many users are on at once. | Some advantages of hosting a web-based application on mobile devices is that they work faster than web pages, are able to be monetized, are not as dependent on a connection to the internet or WiFi, are more convenient to use for users, and allow for notifications, and thus, greater engagement. Some disadvantages include the fact that mobile apps are expensive to develop and maintain, they can take up a large amount of memory and storage, and they must be found and downloaded among the millions of apps available in the app store. In addition, there is an additional cost of an annual fee just to be able to stay on the app store. |
| **Client Side** | A MacOS server can be bought for about $20 and can be used to house a domain or web application. A web application may take just a few minutes to deploy onto the internet for people to use. Expertise involves knowing the specific requirements and expectations of the client for their program, as well as making sure the developers are familiar with the workings of the Mac operating system and the functions and features provided by it. | The cost of supporting multiple types of clients on the Linux server may range from $11-$15 a year for one domain. Depending on the size and complexity of the program, it may take a while to develop, and several minutes to hours to deploy. Because Linux is much less-widely known, in addition to knowledge regarding the requirements of the client, special expertise on how this operating system works will be needed in order to utilize Linux as a server. | Depending on the size of the web application, costs can range from free, if it is a personal or small project, to around $10 for a larger application. It will take a couple months of development to design and develop the program, but should take up to a half hour to deploy, depending on its size and features. Specific expertise is needed when developing software that supports multiple types of clients on a Windows app. Expertise on the programming languages, specific hardware, and specific client and program requirements are important to ensuring that the program is compatible for all its users. | Creating and deploying mobile apps on mobile devices come at a cost. It is more expensive than deploying a web page and can cost anywhere from $10,000 to $30,000 to design and develop. In addition, there is a large cost in maintaining and updating the app. It may also take a few days for the app to be reviewed and approved before being accepted to enter the app store for download. It is important for developers to know the difference between developing a web page and a mobile app for mobile devices, in addition to recognizing the requirements and expectations of the user for their program. Also, consideration on what type of mobile device, such as Apple or Android, is important. |
| **Development Tools** | Languages used by the Mac operating system include C, C++, Objective-C, Swift, and assembly language. Some IDES used to create code for a Mac OS are XCode, Visual Studio, CLion, and Eclipse. | Linux uses the GCC, Python, Ruby, and Java programming languages. Some IDES that are used for Linux are VIM, SpaceMacs, Kate, and Geany. | Programming languages used when developing code on a Windows OS include C, C#, Visual Basic, and JavaScript. The Visual Studio IDE is a common platform for creating this code. | Languages used for mobile apps on mobile devices include Swift, Kotlin, Flutter, Java, and Python. Several IDES used for mobile applications are Android Studio, IntelliJ IDEA, Eclipse, and Visual Studio. |

[**https://medium.com/@jennythomas2365/what-is-windows-hosting-advantages-and-disadvantages-ceecd2fc6c17**](https://medium.com/@jennythomas2365/what-is-windows-hosting-advantages-and-disadvantages-ceecd2fc6c17)

[**https://azure.microsoft.com/en-us/pricing/details/app-service/static/?cdn=disable#pricing**](https://azure.microsoft.com/en-us/pricing/details/app-service/static/?cdn=disable#pricing)

**]**

[**https://learn.microsoft.com/en-us/dotnet/fundamentals/languages**](https://learn.microsoft.com/en-us/dotnet/fundamentals/languages)

[**https://www.linkedin.com/pulse/7-compelling-reasons-why-linux-better-choice-software-muzahid**](https://www.linkedin.com/pulse/7-compelling-reasons-why-linux-better-choice-software-muzahid)

[**https://www.ko2.co.uk/advantages-and-disadvantages-of-linux/#:~:text=A%20general%20disadvantage%20of%20using,setback%20if%20you're%20learning**](https://www.ko2.co.uk/advantages-and-disadvantages-of-linux/#:~:text=A%20general%20disadvantage%20of%20using,setback%20if%20you're%20learning)**.**

[**https://azure.microsoft.com/en-us/pricing/details/app-service/linux/?cdn=disable**](https://azure.microsoft.com/en-us/pricing/details/app-service/linux/?cdn=disable)

[**https://www.redswitches.com/blog/best-linux-ide/#:~:text=Some%20good%20IDE%20options%20for,extensive%20configuration%20and%20extension%20capabilities**](https://www.redswitches.com/blog/best-linux-ide/#:~:text=Some%20good%20IDE%20options%20for,extensive%20configuration%20and%20extension%20capabilities)**.**

[**https://stackoverflow.com/questions/1147685/what-are-the-pros-and-cons-of-using-a-mac-for-web-development**](https://stackoverflow.com/questions/1147685/what-are-the-pros-and-cons-of-using-a-mac-for-web-development)

<https://discussions.apple.com/thread/253748850?sortBy=best>

<https://www.udacity.com/blog/2023/03/c-ides-for-mac.html#:~:text=Xcode,graphical%20debugger%2C%20and%20interface%20builder>.

<https://www.linkedin.com/pulse/top-programming-languages-mobile-app-development-2024-zrhyc>

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

**Operating Platform**: The Linux operating platform will allow The Gaming Room to expand Draw it or Lose It. Linux is a very commonly used operating server, making it well known and widely accepted. It includes a wide range of tools and accepts several different programming languages that allow multitudes of programs to work on the Linux server, as well as be compatible with other computing environments in the case of the game’s expansion. Linux is open sourced, meaning that it a collaborative environment that allows developers to aid and benefit one another with the sharing of resources and ideas. By allowing the reuse of certain materials, developers can focus their efforts on improving and expanding upon existing work. Linux is also known to be very reliable and stable, as well as a relatively inexpensive option. Lastly, the Linux operating platform allows users to customize the settings and controls of their work environment, allowing for easy workability and compatibility between different systems and devices. All these factors make Linux an appropriate operation platform that would allow The Gaming Room to expand the Draw It or Lose It software to other computing environments.

1. **Operating Systems Architectures**:

Linux is an open-sourced operating platform. The main parts of the Linux OS are the hardware, shell, applications, utilities, and most importantly, the kernel. The kernel is the main component of the Linux operating system and controls the hardware components by ensuring only one process happens at a time to avoid any overload or collision. The Linux operating platform also consists of a system library, which houses some universal functions that can be used by any program or software. The shell tells the kernel what the user wants it to do. The hardware is like the equipment of the server, and includes parts like device drivers, memory management, and CPU control. Lastly, the utilities component of the Linux operating system allows the user to complete specific jobs and tasks.

<https://www.geeksforgeeks.org/architecture-of-linux-operating-system/>

1. **Storage Management**:

Storage is the long-term or permanent holding of software information and data. The operating platform Linux would have to store information regarding repeated information, such as the corresponding password to a user’s username, what type of account or membership a specific user has, the image library used by the software, and more. The storage management in the Linux operating system works to store this information in a secure and efficient way so that it makes sure that it does not slow down the program or run out of enough storage space to store all the information. Linux’s storage management system consists of two parts, a physical and virtual server. A good storage management system for Linux will consider the cost, efficiency, and the amount of data that can be stored.

<https://medium.com/@kishorbabum9/storage-management-195dca3df9c3>

1. **Memory Management**:

Memory management consists of the shorter term and temporary holding of information. This can include information regarding who the current winner of the game is, who’s turn it is, what the current image being displayed is, the guesses made by the players, and who is on each team. Memory management in Linux focuses on how to store information in a way that can be quickly and repeatedly accessed. In order for the system to load images and data, Linux uses a technique called dynamic paging to switch out old data to be able to access new data. The memory management system’s main focus is on memory allocation among the kernel and user spaces. The goal of memory management is to store information in an easy-to-access fashion that does not slow down the program, cost too much money, and does not limit the amount of information that can be accessed.

<https://docs.kernel.org/admin-guide/mm/index.html>

<https://www.baeldung.com/cs/demand-paging#:~:text=Demand%20paging%20allows%20the%20system,bring%20it%20back%20into%20memory>.

1. **Distributed Systems and Networks**:

Linux is a very universal operating platform, meaning that it works with and is compatible with many other servers and devices. This will come in handy as the client attempts to have the game communicate between different platforms. If the Draw It or Lose It game is able to allow players to play and communicate with other users on different devices, this is a distributed system. Through a stable network and internet connection, Linux allows players on independent devices to be a part of an overarching, inter-connected shared network. In Linux, distributed software is often placed into packages that are given to different devices. Each package contains information about files, functions, and data that can be shared to that device. With the help of these software packages, different devices can interact with one another in the distributed system.

<https://en.wikipedia.org/wiki/Distributed_operating_system#:~:text=A%20distributed%20operating%20system%20is,the%20global%20aggregate%20operating%20system>.

<https://www.techtarget.com/searchdatacenter/definition/Linux-distros-Linux-distribution#:~:text=Generally%2C%20Linux%20distributions%20consist%20of,contain%20thousands%20of%20software%20packages>.

1. **Security**: Security is a very important part of protecting user information on and in between various platforms. User information can be protected through two-and multi-factor authentication. Using authentication factors can increase the security of the user information by restricting unwanted access to sensitive information, such as user statistics, addresses, and passwords. In addition, a firewall can help serve as an additional layer of protection against attacks. Creating and deploying consistent software updates is also a way to increase the security for the client. Updated software is more likely to maintain the most current safety features and gives security hackers less time to learn and become familiar with the program. The more time hackers are given to work with and learn about the program, the likelier the chances are of them performing some sort of attack. Therefore, consistent updates help protect the client’s sensitive information. Lastly, features like Role Based Access Control help maintain a certain level of control and restrictions throughout the program. By giving people the amount of control appropriate to their level of knowledge and relationship to the software, such as giving developers more permissions than gamers, you ensure that no one person has the ability to alter or use the software in a way that is inappropriate based on their role.