

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.1

## 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 | 06/02/2024 | Jesel Reyes | Adding Executive Summary, Design Constraints, and Domain Model |
| 1.1 | 06/16/2024 | Jesel Reyes | Adding information under Evaluation |

**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)

Currently The Gaming Room has had an android version of the game named “Draw It or Lose It”. They are currently looking to consult with CTS on releasing a version of the same game in a web application where pictures will be rendered using a stock library and are being rendered within 30 seconds. In addition, The Gaming Room would like to offer teams who could not guess the puzzle within the time allotted, another 15 seconds to complete the puzzle to the time limit.

## 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)

The Gaming Room has an Android based game for the game “Win, Lose or Draw”. One of the main design constraints for deploying a web-based application would be ensuring the tech stack being used to develop the web-based application can be deployed with web-deployment tools.

## [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 proposed UML diagram of the web-based application can be seen below:

**"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.**

As seen above, the web-based application will be having a main driver program that is charge of creating and managing teams, players, and games. The program driver also uses the SingletonTester class in order to test out the GameService class only instantiates one single instance of GameService.

The GameService class has been implemented using a singleton pattern meaning there can only be one single instance of the GameService in memory to minimize duplicates. The singleton pattern implementation includes a private constructor within the class itself to only have one single instance of GameService available and can only be created through the getInstance method if the GameService is null.

Once a GameService instance has been instantiated by the program driver, the program driver creates two Game instances through the addGame method within the GameService class. The addGame method takes advantage of an iterator pattern to ensure that there are only unique name games being added and if they are existing, it will return the already existing instance of the game. The same iterator pattern is also implemented within the addTeam and addPlayer methods. If a new instance of either a game or a team is created, it added to an ArrayList of their respective entity.

The Entity class has been inherited to the Player, Game, and Team classes. Entity class holds the common attributes used by the three classes such as ID and name. Both attributes are protected for it to be accessed by its subclasses.

The use of polymorphism and overloaded constructors is apparent whenever looking at the Entity class and seeing it being inherited by other classes such as Player and Team class. The encapsulation and abstraction of the application can also be seen within the implementation of the addTeam method where a team can be added only through the method itself since the default constructor cannot be accessed.

## [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** | The MacOS server side has the following advantages: the macOS hardware is of good quality and includes various security features such as Gatekeeper and FileVault. One of the main disadvantages of using macOS as a server host is the cost of the license as it is more costly compared to Windows and Linux OS. In addition, macOS’s hardware can be limited especially when it comes to video game graphics as it is difficult to be able to customize hardware due to only being able to get Apple hardware. | One of the main advantages of using Linux is the wide range of distributions and its cost effectiveness as it is an open-sourced product. Linux is also known for its flexibility and security as it can help with customization of configurations making them lightweight or suitable for specific optimized distributions. One of the main disadvantages of Linux is since it is open sourced, there is no dedicated enterprise support team to help if there were an issue to occur in production releases. | One of the main advantages of using Windows as an OS is its integration with Microsoft Technologies such as .NET and having such an integration can be beneficial in the deployment and development of applications. In addition, Windows provides a centralized authentication application called Active Directory, which can be useful in enterprise applications. However, one of the main drawbacks of using Windows is the cost of the Windows Server licenses as they can be more costly compared to Linux licenses and license expenses can be stacked up depending on the tools needed. | One of the main advantages to mobile devices is having the ability for an application to be portable as it makes it easy for the user to access the web application from anywhere. Aside from portability, developing and hosting web applications on mobile devices ensure they are platform independent meaning they are built once and deployed to multiple platforms at once such as Android and iOS. However, one of the significant drawbacks of using mobile devices is the performance limitations specially when it comes to the intensiveness of the web application being used that can greatly affect the battery life of the mobile device. |
| **Client Side** | MacOS’s greatest pros are its intuitive user interface and its seamless integration within the Apple ecosystem to other devices. However, it also comes with its cons, which are macOS has limited hardware options meaning that macOS might not be able to be customized for optimization of video game graphics. Also, when it comes to managing and scaling, macOS does not provide flexibility especially when it comes to the number of users. | Supporting multiple clients on Linux involves several factors. Since Linux is open sourced, it generally has a vast amount support through communities if an issue were to occur. Another advantage of using Linux is having the ability to use different distributions of Linux that may be suitable for web- gaming applications. However, due to the multitude of distributions, it can add more time to the development deadlines as it needs to be tested against different distributions. In addition, the developers should have extensive Linux expertise to effectively develops apps and deploy them as there is a learning curve to developing apps in Linux. | One of the main considerations for using Windows is the cost as it can be easy to add additional charges whenever it comes to deploying applications for licenses and development tools. In addition, ensuring that the different Windows distributions are compatible all around will also take a considerable amount of time. Additionally, becoming familiar with the Windows UI can also take some time for developers to become familiar with as well as to integrate different Windows technologies. | To support multiple clients within mobile devices, there would be a need for using different platform specific programming languages and ensuring that they are consistent across all platforms. In addition, the developers would need to have expertise in the different SDKs needed and development frameworks for the mobile applications to be cross-platforms. Another consideration is also security and data privacy which is another area that developers should have expertise in to ensure that data is being handled properly and to prevent unauthorized access. |
| **Development Tools** | Some of the common development tools available on macOS are the following:  Swift, a programming language used to create macOS and iOS applications, Objective-C, another programming language that was replaced by Swift but still used in legacy libraries. An IDE commonly used is called Xcode, it is Apple’s official IDE for macOS or iOS development. Other utilities used within macOS are Homebrew, a package manager used to simplify the installation of command line tools. | The relevant programming languages usually used within Linux are the following: C, C++, Python, Java, Go, Rust. In addition, common IDE used within Linux are the following: Visual Studio Code, Eclipse, IntelliJ, Atom, Sublime. For deployment tools, Docker is a command line tool used for containerization and deployment of applications as well as Ansible, another tool for the automation of deployment. | Some of the relevant programming languages typically used within Windows are the following: C#, VB.NET, ++, Java, JavaScript, Python. Some of the common IDEs available in Windows are the following: Visual Studio Code, Eclipse, Intellij, PyCharm, and Visual Studio. The Internet and Information Services tool is used for hosting and managing web applications. | To deploy mobile applications to mobile devices, there are different platform specific ways. In Android, usually either Java or Kotlin is used to develop applications and the IDE used is Android Studio. For iOS, Swift or Objective-C is used to develop applications and their respective IDE is Xcode. For cross-platforms, Javascript or Typescript is used. There is also Flutter SDK available for building natively compiled applications for different devices such as mobile, web, and desktop. |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>
2. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>
3. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>
4. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>
5. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>
6. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>