

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## Table of Contents

[CS 230 Project Software Design Template 1](#_Toc435761211)

[Table of Contents 1](#_Toc1673226286)

[Requirements 3](#_Toc2062379688)

[Design Constraints 3](#_Toc683460465)

[System Architecture View 3](#_Toc1392426787)

[Domain Model 3](#_Toc1429372149)

[Evaluation 4](#_Toc1737222841)

[Recommendations 7](#_Toc1480435498)

## 

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 02/02/2024 | Robert Taylor | The Executive Summary, Design Constraints, and Domain Model have been completed for the Draw It or Lose It application. |
| 2.0 | 02/17/2024 | Robert Taylor | The Evaluation table has been filled in pertaining to advantages and disadvantages of development utilizing various operating systems. |
| 3.0 | 02/25/2024 | Robert Taylor | The Recommendations section has been filled in. |

**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.

## 

The game *Draw It or Lose It* requires a method of ensuring that only one instance of the game exists in memory at any given time. Additionally, the game must be able to have one or more teams involved, with each team having multiple players. These teams and players must have unique identifiers that are checked for their uniqueness upon creation.

To accomplish this task, the Singleton design pattern will be implemented. This design pattern will ensure that only one instance of the game can be stored in memory at once. The iterator pattern will also be incorporated into the development of the game, and this method will allow any game instance, team, or player to have its identifier checked when it is being created by iterating through the object’s attributes discretely and comparing any object being created to any existing object of the same class.

## 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)

Several design constraints are presented when developing the *Draw It or Lose It* game in a web-based distributed environment. These constraints include:

* + **Connectivity -** A web-based application will require that users have consistent internet access to run the application
  + **Operating Systems -** A web-based application may not be supported by mobile device operating systems such as iOS or Android OS
  + **Concurrency -** A web-based distributed environment runs the risk of concurrency in which too many people may try to run the application a once, risking either application latency or failure
  + **Security -** Running the application in this environment means that user data must be secured from potential threats, as the environment is used my many other users

* + **Scalability -** The application must be able to scale appropriately when handling varying amounts of users and requests

## [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

In the UML class diagram below, the Game, Team, and Player classes all inherit from the parent class of Entity. This is shown with the hollow arrow pointing from the child classes to the parent. The child classes all contain the function toString(), which is an example of polymorphism. These child classes all make calls to the GameService class, as depicted by the black line spanning from each child class to the GameService class. The GameService class contains a default constructor of GameService(), which is where the Singleton design pattern comes into play. This default constructor is private, which prevents other classes from making calls to the constructor, therefore making it so that only one instance of GameService may exist in memory at once. This method displays the principle of encapsulation, as only the appropriate member functions may be addressed, and the rest are encapsulated. The ProgramDriver issues a one-way call to the SingletonTester to assure that the Singleton design is working effectively, as shown by the filled black arrow. All these classes' members are hidden from the players of *Draw It or Lose It*, as they do not need to know these specifics to enjoy the game. This is an example of abstraction.

**"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

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** | Web hosting with MacOS is excellent for safety as the OS comes with built-in encryption and firewalls. Licensing costs are high when using a Mac server as it is the proprietary OS for Apple. The operating platform does offer server-based deployment. Mac does have higher hardware requirements than Windows and Linux. | Linux offers the best support for hosting a web-based application. The OS is low cost free for use, so its costliness is better than the other three options. Linux offers better security than Windows. It is also the most popular hosting OS and supports most scripts. | Windows, like Mac, can be costly for web hosting due to its licensing. This OS is highly scalable and strongly supports cloud computing. Partly due to licensing costs, it is not as popular as Linux. Windows does not offer as high security as MacOS, but it does have lower hardware requirements to run the server. | Deploying a server to host our web-based application on a mobile device is possible but comes with restrictions. Using a mobile device presents challenges that other operating systems may not face, such as processing power, battery life, memory, and storage. Using a mobile device to host a server may present compatibility issues with other operating systems. However, utilizing a mobile device in this scenario grants higher user security than options such as Windows or Linux. |
| **Client Side** | Developing our application with MacOS means that we must consider the licensing costs and expertise that come with utilizing Apple services. Mac is not open source and the SwiftUI framework, which would aid our development, is not free of use.  Developing on a Mac requires the use of the Swift language, which is less popular and may be more costly to find skilled developers. | Using Linux to develop our application would be less costly than Mac, not only in terms of licensing, but in terms of development. Linux utilized commonly known languages like Java and C++, so there would be no need to seek developers for a niche language. With these languages being higher level, development time should be shorter in theory. | Windows development supports many programming languages and frameworks, so there is no additional cost to consider in seeking developers for less common languages. Windows applications typically support multi-user functionality and is the most commonly used operating platform, meaning increased compatability. | Developing our application for mobile devices may vary in cost. IOS devices utilize Swift, which is a lesser known language, meaning that Swift developers would need to be sought out. Android devices, however, utilize Java and Kotlin, with Java being immensely popular. Android devices may not imply additional development costs compared to IOS devices. Multi-user functionality is not typical of mobile devices. |
| **Development Tools** | When developing software on Mac, the usual programming language used is Swift. Xcode is the IDE made by Apple for developing applications on their operating systems. Visual Studio Code may be used, and is typically more popular, and allows for cross-platform development. SwiftUI is a popular user-interface framework for Swift that would allow for efficient development of our web-based application. However, SwiftUI is not free/open source, so this is an obvious drawback. A development team that is familiar with the Apple-specific programming tools listed would be necessary to build the application in this situation. | C is the programming language that is the basis of the Linux Kernal, so it would be a language to consider for this application. Linux-based applications support many popular languages, with the most popular for web apps being Java, Javascript, and Python. Angular would be a great Javascript framework to utilize when building our app and would be supported by Linux. This framework is utilized for building interactive websites and web apps and contains tools for efficiency, including some for client-server communication. Angular is free to use and is open source. | Building our web app on Windows would allow us to use any of the various programming languages that Windows supports. The most sensible for this situation would be Python due to its high level of abstraction from the machine language as well as its plethora of purposes. Python can be written using the IDE PyCharm. Python has a video game development tool called Pygame that we may use to build our application, and it comes with libraries for graphics and sounds that we may include in Draw It or Lose It. Only one development team is necessary for this implementation. Both Pycharm and Pygame are free for use. | When developing our software for mobile devices, there are multiple considerations to keep in mind. One of the most popular mobile operating systems is iOS, which utilizes the Swift language just like Mac. SwiftUI is a framework that can be used in the development process, but it is not free for use or open source. The other popular operating system is AndoidOS, which is utilized by all Android devices. Kotlin is the preferred programming language for Android, though Java is also utilized. Javalin is a web framework that could be utilized in our scenario as it supports both Java and Kotlin development, meaning that we may be able to utilize one single development team. Javalin is free to use and open source. Ktor and Spring Boot are two other frameworks that may work in this scenario. |

## 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 choice of operating platform for the development of Draw It Or Lose It is Linux. Linux has been optimized to run on any hardware, meaning that its implementation would help allow our application to run on more computing systems. On top of this, Linux software is free to use and is open source, resulting in a cheaper solution to our problem. Use of open-source software also eliminates the need for licensing. Another key benefit of using Linux as our operating system is that Linux is an excellent choice for cloud computing, which is what our application will utilize.
2. **Operating Systems Architectures**: The architecture of the Linux operating system follows a layered structure. This structure includes the kernal, system calls, libraries, applications, shell, device drivers, and GUI. One of the biggest advantages of the Linux operating system is its customizability and scalability. The software’s scalable nature would allow us to keep up with the changing system requirements brought on by a varying number of players and user data in Draw It Or Lose It. Additionally, the Linux kernal contains all of its core operating system in one single executable file. This allows the kernal to support a wide variety of hardware and system configurations, as do the device drivers. This further shows how Linux would be an excellent choice of operating system for Draw It Or Lose It.
3. **Storage Management**: Cloud storage will undoubtedly be the wisest choice for storage management in our application. The best service to choose for this task would be Amazon S3. S3 offers one of the lowest prices of popular cloud storage services while containing some of the best performance and security. Storing our application’s data on the cloud will allow for fast access to the game’s files while not requiring the user to store any of the data on their own device. A close competitor with our choice of cloud storage service is Azure Files by Microsoft. This service is low cost, easy to use, and fully managed. However, Amazon S3 will suit our needs better due to its increased performance and inclusion of user permissions. These user permissions incorporate a “bucket policy” that allows an administrator to manage access to stored resources.
4. **Memory Management**: Linux manages application memory through the use of a virtual memory system. This system manages memory by utilizing the kernal to transfer data between RAM and the storage on the hard drive. This is known as space-swapping, and it is done by breaking programs into pages and the kernal then swaps pages that are not necessary out of RAM and to the hard drive. This dependency on a device’s RAM allows for less strain to be put on the CPU and hard drive. Linux’s method of managing memory helps to boost performance and makes it a better choice for Draw It Or Lose It over other operating systems, such as Windows.
5. **Distributed Systems and Networks**: The application’s need to communicate among various platforms can be met by implementing APIs. APIs allow for applications to communicate with each other, so its implementation would allow our software to communicate with other operating systems. Our application’s API should follow REST guidelines since it is a popular and widely used practice in developing web applications. Cloud based architecture can provide helpful features within our application in the way of server connectivity, possible outages, and other threats to application integrity. Running Draw It Or Lose It on a cloud server would minimize dependencies within the distributed systems and networks by storing program files in a backed-up and off-site location, meaning that data may be recovered if it is compromised and that any outage to a particular device would not affect any information stored in our cloud server.
6. **Security**: Some best practices for protecting user data between various platforms include multi-factor authentication within the app and controlling user access privileges. Multi factor authentication is an excellent means of protecting data and helps to ward unauthorized logins. For example, the user’s password as one factor of authentication and an emailed code as a second factor could prove to be a great security measure within our app without being too inconvenient to the user. Controlling user access privileges by allowing only certain users to access to certain information or actions will further help to secure our application, for instance limiting access to user account information to only the specific user and any administrator. Linux is an excellent choice of operating system for Draw It Or Lose It in terms of security because it supports both of the features listed above as well as file encryption.