

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## **Table of Contents**

[**CS 230 Project Software Design Template** **1**](bookmark://_l6ti7uoag22u)

[**Table of Contents** **2**](bookmark://_30j0zll)

[**Document Revision History** **2**](bookmark://_grjogdjh5fi8)

[**Executive Summary** **3**](bookmark://_sbfa50wo7nsh)

[**Design Constraints** **3**](bookmark://_2et92p0)

[**System Architecture View** **3**](bookmark://_ilbxbyevv6b6)

[**Domain Model** **3**](bookmark://_8h2ehzxfam4o)

[**Evaluation** **3**](bookmark://_2o15spng8stw)

[**Recommendations** **5**](bookmark://_m8aleynsvzvc)

## **Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 7/13/2022 | Christopher Reyes | Begin development on the gaming 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**

The Gaming Room requests assistance in the development of their web-based game, Draw It or Lose It. A set of software requirements and functionality has been provided. The requirements for hardware are subject to change as the software development process progresses. My solution is to systematically design, implement, and develop a product that meets the clients' needs.

## **Design Constraints**

1. The design of the game must resemble that of the 1980’s television game Win, Lose or Draw.
   1. The application is restricted to a theme and functionality that is like the old TV game. This means that freedom to customize the game is not available.
2. The game must have one or more teams, with each team having multiple players assigned to it.
   1. With the game allowing one OR more teams, a way to create multiple instances of an object must be implemented. The same goes for having multiple instances of players for each team.
3. Only one instance of the game can exist at a time.
   1. This means that the game instance must be unique as well as the teams and players. The use of the singleton pattern should be used for this functionality.

## **System Architecture View**

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**

The UML diagram below shows multiple classes and relationships. The base class is the Entity class, which is inherited by the game, team, and player classes. The GameService class is associated with the game class which is associated with the team class which is then associated with the player class. Each association shows a 0 to many relationships, meaning there can be multiple instances of each class made. On the side we have the ProgramDriver class, which uses the SingletonTester class. The ProgramDriver class contains the main() method and is where the program begins and makes calls to the other classes. With the Entity class parenting the game, team, and player classes, it makes it possible for all three classes to have unique functionality for name, id, and entity. This makes the software requirements possible to achieve, where the client requests that there be multiple unique instances of each class.

"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 

## **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** | Mac is a closed platform that requires updates from Apple and must be updated to the newest software version (paid updates) when required. Mac is usually the more expensive option out of the three. A benefit of using Mac is the reliability from a hardware and software standpoint, as well from a security standpoint. | Linux is an open source platform, making it the cheaper of the three operating systems. Linux also requires the least amount of computer resources to run properly and has the least amount of updates required. Although it is a cheaper option, it is also not as secure as the Windows or Mac. | Windows is a closed platform that requires updates from Microsoft and is the second most expensive hardware/software system. Windows needs to be frequently updated. There are multiple tools that are Windows specific, such as ASP and .NET frameworks. | Mobile devices are restricted in terms of which programming tools/language that is available for each platform. (android vs IOS) The hardware and network requirements also make development more difficult on mobile platforms. |
| **Client Side** | Mac development can have a steep learning curve, and applications are directed towards the Apple App Store for deployment. This means developers must pay the App Store fee. Mac development help from online sources can be hard to come by for beginners. | The cost of development with Linux is cheaper when compared with Windows and Mac options. The expertise required for Linux applications is more than Windows and Mac. However, there is a lot of support and resources for Linux as it is an open-source platform. | The software development cost for Windows is expensive and can have a moderate learning curve to build applications that support multiple platforms and client’s needs. Also, the time required to build such applications is moderate. | Mobile app development requires learning technology that can be used for both IOS and Android. The cost of development can vary based on the complexity of the app, but to host an app on the store also costs money. The time required to build such apps are usually quicker than building apps on desktops. |
| **Development Tools** | Xcode is a Mac specific IDE that is great for building software on Apple devices. Mac supports most major software and tools; which can be downloaded off the internet. | Linux supports most programming languages and IDE’s that are also available on Mac and Windows. Linux also comes with built-in tools and compilers. Some downloadable tools are difficult to configure with Linux. | Windows supports most programming languages and uses IDE’s such as Visual Studio for development. Development environments and compilers need to be installed on Windows, which is usually easy. | Mobile devices for Androids use Java/Kotlin and for IOS uses Objective-C/Swift. Some examples for IDE’s for mobile are Android Studio and Xcode. |

## **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 Windows for the development of Draw It or Lose It.
2. **Operating Systems Architectures**: The Windows Operating System architecture uses a simple structure such as the MS-DOS. I recommend this structure because of its simplicity and support of many applications. The structure of the operating system was not divided into modules like the other modern operating systems. The MS-DOS system uses layers that hide certain functions and operations from other layers.
3. **Storage Management**: The storage management system recommended for this game is the tree-structure directory. I chose this system because the game deals with multiple instances of games, teams, and players, all of which need to be organized and stored individually. When dealing with separate object instances and images, it is best to have an organized structure that can store multiple different files. This storage management system also pairs well with the Windows operating system.
4. **Memory Management**: Windows uses a segmented memory management technique called file mapping, which stores data to addresses in memory. The address is then used to open, read, or write from, making memory storage and access efficient. I recommend this use of file mapping because the game is fast paced and requires opening and closing files often. The game requires different management of different object instances and image files. With many different types of data needed at any point in the game, it is best to have an efficient memory management system.
5. **Distributed Systems and Networks**: I recommend the client server system to communicate between the various platforms and connect devices to the network. The client server system is simple and requires a central server set up where clients can connect to via networked devices such as a cell phone or tablet. This would be ideal for our game because it supports various platforms and good storage management systems. It also allows central control over the devices and games. This makes managing the game app more efficient and simpler.
6. **Security:** Security is crucial for applications that are hosted on a network. By encrypting the network, we can make sure that only the specified receiver should be able to read messages. This includes if the game holds any kind of personal or sensitive information. The information would be safe while submitting it and playing the game. This protects the users and the gaming room from any cyber threats. User authentication is also a good way to protect the users by requiring a unique username and password combination to play the game. This ensures that only the authorized player can access his game, team, and players' information.