

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 | 05/21/2023 | Bryan Pirrone | Executive summary, requirements, design constraints, and domain model |
| 1.1 | 06/03/2023 | Bryan Pirrone | The operating systems evaluation table updated |

## [Executive Summary](#_sbfa50wo7nsh)

Our company, Creative Technology Solutions (CTS), was asked by our new client The Gaming Room to develop a multi-platform web-based game called Draw It or Lose It. Draw It or Lose It is currently only compatible with Android devices. Inspired by the hit 1980’s television game show *Win, Lose or Draw*, this web-based game will allow for multiple teams of multiple players to compete for fame, glory, and of course, bragging rights.

A singleton creation pattern has been chosen to prevent multiple game instances existing at the same time, as well as an iterator pattern will prevent conflicting team and player names.

## Requirements

The Gaming room has given us several requirements for the game application; a game will have the ability to have one or more teams involved, each team will have multiple players assigned to it, and game and team names must be unique to allow users to check whether a name is in use when choosing a team name, and only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

The Gaming Room has previously deployed its game, Draw It or Lose It, on Android platforms exclusively. CTS has been brought on to help The Gaming Room implement their game, Draw It or Lose It, as a web-based game on multiple platforms.

Java has been selected for the purpose of deploying Draw It or Lose It as a web-based game. Any existing code or integrations for the Android game need to be re-evaluated to ensure compatibility across multiple platforms and operating systems.

Complications may arise when configuring compatibility across all platforms and operating systems. We will have to ensure rigid and thorough testing during implementation.

## [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 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.**

The main class, ProgramDriver, initiates the creation of new games, teams, and players. Game creation is generated through the GameService class by following a singleton design pattern. This allows for only a single GameService class to exist.

The GameService class has several private and public variables. The private variables are games, nextGameId, nextPlayerId, nextTeamId, and service. The public variables are getInstance, addGame(name: String), getGame(id: long), getGame(name: String), getGameCount(), getNextPlayerId(), and getNextTeamId(). In order to initiate GameService is with the getInstance() method. getInstance() determines whether GameService has been started and only initializes it if it isn’t already started.

Once GameService has started, the ProgramDriver class can call on the method addGame(). addGame() uses an iterator pattern to prevent Games from being created with similar names. Once a new game is created, it is added to the games List.

Once a game is created a team can be added to the game with the method addTeam(). addTeam() uses a very similar iterator pattern to addGame() which prevents teams from being created that have the same name. The newly created team will be added to the teams List.

Once a team is created a player can be added to the team with the method addPlayer(). addPlayer() uses a very similar iterator pattern to addGame() and addTeam() to prevent identical players from being created. Newly created players will be added to the players List for their associated teams.

The classes Game, Team, and Player are all subclasses of the Entity class. The Entity class has two private attributes: id and name. The default constructor is also private.

The UML design shows the Object-Oriented programming techniques of polymorphism, inheritance, abstraction, and encapsulation. Inheritance and polymorphism are used in the extension of the Entity class and the overloading of constructors. The concepts of encapsulation and abstraction are used when adding teams. A team and player may not be created directly since the construct is blocked, but the end-user is capable of accomplishing this function with the addTeam() method, even though the user does not know how that team is added to the list.

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac has many advantages, disadvantages, and characteristics. Some advantages include easily accessible terminal commands for server configuration, ease of maintenance, and the cost can be relatively cheap. Some disadvantages are that macOS server has been discontinued on April 21, 2022, so anyone who previously owns that can use it but there is no future support for it, has limited hardware options available, limited compatibility with Windows, and can be quite expensive. | Linux has many advantages, disadvantages, and characteristics. Some key advantages of using Linux for our server would be it is free, it is open-source and therefore has many resources available and can be extremely reliable regarding cyber protection. The main disadvantage of using Linux would be based on familiarity with computers and Linux. We would specifically need someone who is versed in Linux to run the server since it may not be the most intuitive to people who are not familiar with it. | Windows has many advantages, disadvantages, and characteristics. Since Windows is the most used operating system, implementation and use would be the easiest. Windows server costs can be expensive, so that is another factor to consider. Windows has the capability to run on many different hardware options. Licensing costs can also become expensive in a short amount of time. | Although a mobile device can be used as a web server or file server, these devices are not intended for this purpose. Hardware limitations would make it extremely ineffective at hosting a multi-user server. Regarding costs, this could be cheaper but would require a lot of development. |
| **Client Side** | Expensive for client-side implementation and requires the purchase of an Apple product which is also expensive in comparison to Windows. Apple software is limited regarding the creation of applications when compared to Windows. | While the best cost-effective option, significant time may be required since not all people are familiar with using Linux. This is primarily because it is not as widely used as an OS compared to Windows. | Windows is the most used operating system. Due to this fact, most users have experience with using windows, therefore time and ease of use aren’t much of a concern. Unfortunately, since Windows is not open source, the cost will be expensive when compared to other options. | Multi-user functionality is not inherent in the design of a mobile device. Applications design is very straightforward for both iOS and Android devices. While iOS and Android devices are the most common, there are other kinds of operating systems for mobile devices that could prove to be an obstacle. Other concerns such as hardware and battery life may also prove to be factors that we need to consider. |
| **Development Tools** | The most common and highly recommended open language for building powerful apps is SWIFT. This would probably be our best option because it is up to 2.6x faster than Objective-C and up to 8.4x faster than Python. XCODE would be the IDE used with SWIFT since it is recommended on Apple’s website. SWIFT also includes Objective-C interoperability to allow for SWIFT code to co-exist alongside existing Objective-C files in the same project, per the Apple developer website. Xcode would be the IDE used for iOS programming, running at $99/year. | Linux supports most programming languages. The most used are C, C++, Java, JavaScript, HTML, and Python. While this list is not all-inclusive, Linux can support many more programming languages. Python IDEs are often free and easy to use, such as PyCharm or NotePad++. For other programming languages, IDEs may not be supported but can be used through virtual IDEs. | Similarly, to Linux, Windows supports most programming languages. PyCharm and NotePad++ are both useable options for IDEs if using Python. Microsoft’s Visual Studio is a very popular IDE with a large selection of plug-ins available, although it isn’t free. This is very common for .NET and C++ developers. Microsoft Visual Studio costs can range from $45/month for the professional subscription to $250/month for the enterprise subscription. | iOS’s use of Objective-C and SWIFT are commonly developed in Xcode, with a current price of $99/year. Androids’ software development kit (Android SDK) is primarily Java based and is used commonly with the Android Studio IDE. Android Studio is currently free. |

## 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.>