

Draw It or Lose It

# **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 4**

[**Evaluation 5**](#_Toc115077325)

[**Recommendations 7**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/11/23 | Lane Berrevoets | Additions of Executive Summary, Requirements, Design Constraints, and Domain Model, for use during creation of Draw It or Lose It web application. |
| 1.1 | 10/11/2023 | Lane Berrevoets | Additions to Recommendations section |

**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)

Our new client, The Gaming Room, wants to develop a web-based game that serves multiple platforms based on their current game, “Draw It or Lose It”, which is currently available in an Android app only.

A game consists of 4 one-minute rounds. Each team, consisting of multiple players, will have 30 seconds to guess the puzzle (a phrase, title, or thing). If the team does not provide the correct answer in the allotted time, the remaining teams have 15 seconds and one guess to solve the puzzle.

Creative Technology Solutions (CTS) will either re-write code to allow support of Linux, Apple, and Windows platforms, or a method to utilize the existing code will be employed. Ease of update, expense between either method, and time to completed product will need to be assessed in order to determine which of the methods CTS should utilize.

## Requirements

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names will be unique so that users can check whether a name is in use when choosing a team name.
* Only one instance of the game can exist at any given time.

## [Design Constraints](#_2et92p0)

The game currently runs on Android devices. In order to have a playable game on Apple, Linux, or Windows, we will have to come up with a method to use the existing code or re-write the code. The amount of work involved will be dependent on which of the aforementioned methods used.

Each team will have the ability to have one or more players assigned to that team. This will be accomplished by use of the iterator pattern within the addPlayer() method.

Game and team names will be unique so that users can check whether a name is in use. To do this, we will utilize the iterator pattern to complete both the addGame() and getGame() methods.

As described under Requirements, we will only have one instance of the game in existence at any given time. We will utilize the Singleton Pattern to adapt the GameService class, so that only one instance of the game may run at any given time.

## [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 above UML class diagram is a Structural Diagram; specifically, it is a Class diagram depicting each individual class and its association with all of the other classes.

The Entity class is the Superclass or parent class in which the Game, Team, and Player classes inherit attributes and methods for use within their own class.

The ProgramDriver and SingletonTester classes are an association, meaning that a relationship between the two exists. In this case, the SingletonTester uses the ProgramDriver.

Looking between GameService, Game, Team, and Player, we see that each associates with each other. In descending order, they all have multiplicity. For instance, Game will consist of zero to many Team(s), which will consist of zero to many Player(s). Likewise, a Player will only have one Team, and a Team will only have one Game.

Within each “box” there are three parts. First, the top block will consist of the Class name. The middle block will contain attributes associated with that class. Finally, the bottom block will consist of operations. The “-” indicates that the attribute or operation is private, while the “+” indicates that the attribute or operation is public.

Above, we described that the Entity is the “parent” class of the GameService, Game, Team, and Player classes, which shows Inheritance. Inheritance is one of the Object-Oriented Programming (OOP) principles used within this program. It allows us to override the parent values or methods but also add new data and functionality to the parent.

Next, by setting attributes and operations to private, we are using Encapsulation. Each field will utilize a getter and setter in order to pull the information and write to it.

Polymorphism is used as well. Polymorphism allows us to perform actions, such as Overloading in the instance of the getGame() method in the GameService class. This allows us to call the method with the specified parameters, thus changing the outcome of the method. We also see the toString() methods called out in Game, Team, and Player classes, as well as the parent class. This most likely means that method Overriding will be in use. Overriding allows us to define specific outcomes of the child class methods within the child class itself.

By utilizing the OOP principles, we create a program that is easy to read, edit, and use.

## [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** | **Characteristics:**  Popular within coding community.  Terminal is fairly easy to use.  Not typically used to host large scale applications.  **Advantages:**  Upgradable option.  Web hosting options.  Offers server-based method of deployment.  Less-susceptible to viruses.  **Disadvantages:**  Stigma creates limited end-users.  May have to pay licensing costs due to the use of MacOS software. | **Characteristics:**  Free to use.  Open source, means that enthusiasts can modify it as necessary.  Same basic terminal as Mac.  Popular choice due to security features.  **Advantages:**  Free to use.  Security flaws are quickly corrected by users.  Open-source, meaning that there are no licensing costs.  **Disadvantages:**  Difficult to use by newer individuals in the field. | **Characteristics:**  The most software out of all other OS.  Used by most of the population.  **Advantages:**  Regular updates.  Used by a large population.  Ease of use, most of the population has used Windows OS in the past.  **Disadvantages:**  Susceptible to viruses.  May have to pay licensing costs due to use of Windows OS software. | **Characteristics:**  Portable.  Very popular, most individuals have-use smartphones now.  **Advantages:**  Wide popularity in end-users.  Cost-effective option.  Native application development frameworks.  **Disadvantages:**  Comparatively lower security.  Difficult to capture entire end-user crowd.  Knowledge requirement when creating the application such as Swift and Xcode for Mac products or Android Studio and Java for Android. |
| **Client Side** | Cost comparable to Windows products, moderate time and expertise required to develop products. | Free, open-source platform, most amount of time and expertise required to develop products. | Cost comparable to Mac products,  Least amount of time and expertise required to develop products. | Potentially cost effective. Difficult to implement over other products. |
| **Development Tools** | Uses Swift programming language and Xcode development tool.  Swift is specific to MacOS products, may need to put a team together specifically for this project. | Typically uses C programming language and Eclipse as the IDE or Development tool. | Uses Visual Basic programming language and Visual Studio as development tool.  Visual “suite” may require a specific team, dependent on experience. | Apple – Uses Swift programming and Xcode development tool  Android – Uses Java programming language and Android Studio as the development environment.  Advantageous to utilize multiple development teams working in tandem to create the application. If the Android application is “live”, may not need this option. |

## 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**: Creative Technology Solutions recommend The Gaming Room utilizes Linux. Linux is widely used for server applications and provides fantastic support for application creation and customization. Due to Linux being an open-source platform, it is constantly updated and maintained by fellow users. Additionally, due to being open-source, there will be no additional cost to utilize the features within Linux or Linux specific licensing costs.
2. **Operating Systems Architectures**: Draw It or Lose It will be a cross-platform game. Due to the game being created as a web-based application, CTS recommends using a Three-Tier Database Architecture. This will result in a faster gaming system that is more secure. A three-tiered database will not allow the user to communicated directly with the database, and utilizes a middle application tier to handle the logic of the program. Either side of the application tier will consist of the data end and user end tier. Three-tier architectures are more readily scalable than their two-tiered counterpart. Linux
3. **Storage Management**: Due to the game being a web-based application with cross-platform capabilities, Creative Technology Solutions recommends the use of cloud-based storage environment to store game data. As the game popularity rises and future updates are added, it is important to have a storage system that is both scalable and secure, both of which are true with cloud-based storage. Local files should be retained on user Solid-State Drives (SSDs) at a minimum to prevent any adverse effects to game performance.
4. **Memory Management**: Draw It or Lose It should not have any inherent issues with memory management as long as the architecture supports efficient memory management. We do not want any inadvertent memory leaks within the gaming application. With each high-resolution picture being contained within 8 MB and most platforms containing 8+GB of memory, it has been determined that most platforms will handle the game well. Linux utilizes a paging system to manage memory, this will allow for efficient, optimized use the user’s device memory while operating Draw It or Lose It.
5. **Distributed Systems and Networks**: The Gaming Room would like Draw It or Lose It to be a cross-platform game. A RESTful API is recommended as they are widely used for web applications and utilize HTTP to communicate over the internet. There is excellent support within Linux for RESTful APIs.
6. **Security**: Security must be maintained to ensure that the game is protected against attacks and data breaches. CTS will employ secure coding practices while developing the code for Draw It or Lose It. Encrypting data and verifying user input will assist in ensuring that the game remains secure and fair for all who play it. Linux provides access control and firewalls, and due to its open-source nature, is frequently receiving security updates. Utilizing Transport Layer Security or TLS encryption during user communications between platforms will assist in creating a secure environment for communication to occur and prevent any inadvertent interception of information by attackers.