

Draw It Or Loos 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 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 5/26/24 | Hiroshi Thomas | Draft |
| 1.1 | 6/9/24 | Hiroshi Thomas | Server Side, Client Side, Development Tools |

**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)

Creative Technology Solutions has been asked by The Gaming Room to create a web-based version of their game app “Draw It or Loose It”. The game should be designed with the following abilities:

* Have one or more teams play.
* Multiple players assigned to reach team.
* Unique Team & Game names
* On instance of game in computer memory at any given time with unique identifiers for each instance of a game, team, or player.

## Requirements

*<* Not being assessed >

## [Design Constraints](#_2et92p0)

The Gaming Room currently employs Draw It or Lose It on an Android platform. To expand its accessibility to the web, CTS has been tasked with adapting it. Java has been chosen as the tech stack for its compatibility with web deployment, leveraging its native support in the Android SDK. Existing APIs catering to the Android platform must undergo review or modification to accommodate mobile usage.

## [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 application comprises a primary driver class responsible for initializing game, team, and player creation. The creation process is is done by the GameService class, which follows a singleton design pattern to ensure that only one instance of the class exists in memory at any given time.

GameService prevents the creation of multiple instances by setting its constructor to private. To instantiate a GameService object, one must use the getInstance() method. This method verifies whether a GameService instance has been initiated and initializes it only if it does not already exist in memory.

Once GameService has started, the driver class can invoke the addGame() method. Utilizing the iterator pattern, addGame() ensures that duplicate Game objects with similar names are not created. The newly generated Game object is subsequently appended to the List of games.

After Game creation, a team can be added using the addTeam() method. This method employs the iterator pattern to avoid the addition of duplicate Team objects in the game. The freshly created Team object is then included in the List of teams.

After the team is created, a player can be add into the team through the addPlayer() method. Again using the iterator pattern, addPlayer() ensures that duplicate Player objects with similar names are not appended to the team. The newly created Player object is subsequently added to the List of players.

The Game, Team, and Player classes all inherit from the Entity class. Entity contains two protected attributes: id and name. The default constructor is also protected, preventing the creation of null objects.

The UML designed uses multiple object-oriented programing techniques including polymorphism and inheritance which are used to extend the Entity class. Encapsulation and Abstraction are also used to add teams.

## [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** | Mac can serve as a host for web-based software applications, although it's not as popular as Linux or Windows. It provides a Unix-based environment, which is good for web application hosting due to its strong security and stability. Mac OS Server incorporates functionalities such as the Apache web server and PHP support. However, server-grade Mac hardware options are limited, and Mac-based solutions may be pricier compared to Linux alternatives. | Linux stands out as a favored option for hosting web-based software applications because it has a wide variety of server deployment options with exceptional stability and scalability. Cost-effectiveness, customization capabilities, and a diverse selection of server software options makes Linux stand out. Linux is also known for its robust security and reliability. Setting up and managing Linux servers might demand more technical expertise compared to Mac or Windows counterparts. | Windows Server stands as a prevalent platform for hosting web applications, particularly those using Microsoft technologies. It seamlessly integrates with most Microsoft tools and technologies, giving a variety of web server options. It also provides robust support for .NET applications. Licensing expenses may be substantial and it might not be as conducive to open-source software development. | Mobile devices typically don't directly host web applications. Servers are used to deliver mobile app the data required. Any of the platforms mentioned previously can serve as backends for mobile apps. Mobile devices have a relatively straightforward deployment for small-scale applications but limited resources may result in performance issues. Potential issues such as battery drain and hardware failures are also a concern. Security risks stemming from potential vulnerabilities and reliance on network connectivity and potential bandwidth constraints should also be considered. |
| **Client Side** | Developing applications for Mac clients usually involves utilizing Apple's development tools like Xcode. Costs are typically moderately high, and proficiency in Swift and/or C might be necessary. | Creating software for Linux clients can vary based on the distribution and desktop environment. Costs are usually minimal, although proficiency might be necessary for specific nuances based on the specific hardware (i.e. x86, x64, arm, etc.). | Creating applications for Windows frequently uses Visual Studio. Costs can fluctuate, but proficiency in .NET languages such as C# might be necessary. | Creating applications for mobile devices entails platform-specific development, with costs contingent on the number of platforms targeted. Proficiency in languages like Swift for iOS or Kotlin for android, or cross-platform tools like Flutter might be necessary. |
| **Development Tools** | Xcode serves as the primary integrated development environment (IDE) for Mac and supports Swift, Objective-C, and C++. | Linux offers support for many programming languages. Popular IDEs include Visual Studio Code and Eclipse. | Visual Studio serves as the primary IDE for Windows development supporting languages such as C#, C++, python, java and others. | For iOS development, Xcode with Swift /Objective-C are popular. For android development, Android Studio paired with Kotlin or Java. Cross-platform tools such as Flutter are popular tools to develop on both platforms. |

## 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**: Developing on Windows will likely be best because of user familiarity and is already most likely integrated into the business.
2. **Operating Systems Architectures**: Java will likely provide the best results for the game with it’s cross platform compatibility.
3. **Storage Management**: Cloud storage such as AWS is a great solution to not only hose the servers but also manage the needed user data.
4. **Memory Management**: Windows Memory Management tracks all memory locations and determines how much memory should be allocated to process.
5. **Distributed Systems and Networks**: Using AWS to host the game provides a robust system with data centers located around the world. All version of the game can communicate with the servers and maintain a centrally located host to run the game.
6. **Security**: Designing the game with security in mind ensures that not only is user data protected but also protecting other data located within the same network. Additionally, regular security checks and updates must be done to ensure that the game is safe and up-to-date with the latest security standards.