

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | <07/21/24> | Kevin Bagayas | Initial Project Components/Specifications |
| 2.0 | <08/4/24> | Kevin Bagayas | Evaluations |
| 3.0 | <08/11/24> | Kevin Bagayas | Recommendations |

**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)

The client, The Gaming Room, would like to develop a web-based game named “Draw It or Lose It” which can only be accessed in Android app to be accessible in multiple platforms. This expansion will aim to reach more audience because it would be playable in different platforms rather than the Android app. It will be modified by rendering images from a large library rather than a player drawing images in an easel.

In order to fulfill the demand, Creative Technology Solutions will develop the client’s requirements by properly planning the project with the complete technical details and overall framework of the game structure. The experience should have the same feel when used in the Android app and to the proposed multiple platforms and devices.

## Requirements

Software Requirements:

* 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 must be unique to allow users to check whether a name is in use when choosing a team name
* Only one instance of the game can exist in memory at any given time.

Technical Requirements:

* Multiple Operating Systems (Mac, Linux, Windows and Mobile Devices)

Gameplay Requirements:

* Consists of four rounds of play lasting one minute each
* The application will render images from a library instead of a player drawing images on an easel
* Drawings are rendered at a steady rate and are fully complete at the 30-second mark.
* If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit.

## [Design Constraints](#_2et92p0)

* Multiple Platform Operability – it should have the same functions and features when the application is used in different platforms. Compatible in different platforms.
* User Interface – user experience should be uniform across different platforms and easy to use.
* Copyright - in terms of the images used from the library and the gameplay similar to the Win, Lose or Draw.
* One instance per game – only one occasion can be saved in the memory thus limiting the game to the last saved event.
* Multiple Players – gameplay should accommodate number of players grouped in different teams.

## [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)

On the UML diagram, the Entity class has a relationship with the Game, Team and Player classes. With the arrow pointing to the Entity class means these classes inherit from a super class. It has the common attributes like id and name. For the Game class, it contains the teams that are playing. For the Team class, it contains the players where it would have the team’s name. For the player class, contains all the players in the game and their name. For the singletonTester, the programDriver is used to test the code. This is to ensure that only instance exists one at a time. There is also inheritance within the diagram by having three classes share attributes. Encapsulation also is within it by only using the necessary parameters needed for 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 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](#_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** | I think this would be the less preferable option since it is not commonly used but comes with strong security system and can easily be integrated to its Apple products. It would be an expensive option and comes with limited options. | This would be an open-source and highly customizable system but may need more technical expertise than when compared to other platforms. | This server is one of the commonly used because of its easy integration and comes with a lot of options in terms of software compatibility. Large user community support. Security vulnerable system. | It does not usually host web applications but due to its portable size, app-based and simple interface makes it more convenient to develop compared to the other three. |
| **Client Side** | Specialized expertise with Swift and also using Apple’s Xcode would be required. Moderate time and also expensive. | More expertise needed and cost will depend on the distribution and desktop environment used since it is also open-source. | Costs can vary and since it is widely used, minimum expertise is needed. Simple and easy to learn. It uses mostly the known development tools. | It is platform specific development and costs will be based depending on the intended platform. Cross platform tools can be used. |
| **Development Tools** | Xcode would be the primary IDE for Mac. It also supports Swift. Applications needs review by Apple. | Supports wide range of programming languages. Open source. Popular IDEs are Visual Studio Code, Eclipse for Java. | Visual Studio is mostly used IDE and can support multiple languages like C++, Java. | For IOS devices, XCode and Swift. For Android, Android Studio and Kotlin/Java. |

## 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 would recommend the Microsoft operating system. It is best in terms of the range of IDEs to work with. It is also flexible in terms of adaptability of the Windows operating system. Many systems can work or aligns with the Windows system. Plenty of emulators available on Windows.
2. **Operating Systems Architectures**: Windows 11 is the latest release that is developed by Microsoft. It has increased security features and its collaboration features as standard in terms of software integration. It also a good directory structure for data storage.
3. **Storage Management**: Windows 11 comes with the Windows Storage Management. It features allows to manage the computer’s storage resources. It offers storage spaces, optimization, sense, disk cleanup and storage management console. Another option is the Cloud Storage.
4. **Memory Management**: Windows 11 has an improved memory management for faster and more efficient memory loading. Each process in Windows 11 improves the utilization of virtual memory address spaces which is more sufficient for this specific application.
5. **Distributed Systems and Networks**: Windows have the Microsoft Azure which offers cloud computing services. It offers development and management of applications that are designed to operate in multiple machines or nodes in a network.
6. **Security**: Windows have a built-in security that includes an antivirus program named Microsoft Defender Antivirus. It provides a user-friendly interface and tools to manage common security features. It also can integrate easily other security protection application to supplement it.