

Draw It or Lose It 2

# **CS 230 Project Software Design Template**

Version 3.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 3.0 | 12/7/22 | Aydar Fayzullin | Revised 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)

Create a web-based game that serves multiple platforms based on the Android game. The application will render images from a library of stock drawings as clues. A game consists of 4 rounds of play lasting 1 minute each. Drawings are complete at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams is offered 1 guess each to solve the puzzle with a 15-second time limit.

## [Design Constraints](#_2et92p0)

1. The game should have the ability to have one or more teams playing.
2. Each team should have multiple players assigned to it.
3. Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
4. Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.
5. The game should be able to run on all devices
6. To ensure only one instance of the game can exist singleton pattern should be utilized.

## [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 Entity class creates relationship among Game, Team, and Player classes. All these classes inherit information form the Entity super class. Each class has a common reference such as name and id. Team and Player classes have a has-a (aggregation) type of relationship. Aggregation type of a relationship is a relationship when a class has a reference of another class known to have aggregation. Aggregation is a type of relationship that Game class has a Team class, GameService class has Games class. In our case GameService has a reference of Games, Games a reference of Team, and Team a reference of Player.

**"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 must 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** | MacOS has terminal commands to make changes and you can configure the server access. It’s less popular for doing these jobs than the other OSs. | Linux can make changes. You can configure the server access and it’s more affordable. There are fewer apps to support the web hosting required demands. | Windows is the most widely used OS. It has Windows Server. Expensive to implement. Windows OS seems to be prone to virus attacks. | Mobile Devices are cost-friendly, but they have fewer computing capabilities. |
| **Client Side** | Has a moderate cost. Some experience needed to operate them. Apps built on Macs must be compatible with browser. | Has a low cost. Linux is open source. You need a lot of experience to operate them effectively. | Has a moderate cost. Little experience needed to operate Windows. | Have a low cost. Little experience needed. More time required to satisfy mobile devices requirements. |
| **Development Tools** | Swift and Objective-C are used to create MacOS compatible apps. XCode and Visual Studio Code are the most popular IDEs for Mac apps. | Besides standard HTML, CSS, and JavaScript, you can create Python, Java, C++ apps. Visual Studio, PyCharm, IntelliJ, Eclipse IDEs can be used. | Besides standard HTML, CSS, and JavaScript, one can create Python, Java, PHP, C#, C++, Go, etc. apps. Visual Studio Code, PyCharm, IntelliJ, Eclipse IDEs can be used. | For iPhones Swift, React Native, Ionic languages and frameworks can be used. For Android phones Java, Kotlin, React Native, Ionic languages and frameworks can be used. Visual Studio Code, IntelliJ, XCode IDEs can be used. |

## 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**: To allow The Gaming Room to expand Draw It or Lose It to other computing environments, I would recommend Windows OS since it’s the most universally used OS around the world. It’s relatively easier to get hold of it compared to MacOS and Linux.
2. **Operating Systems Architectures**: Windows OS architecture is a layered design that is comprises two primary elements: user mode and kernel mode. Applications can use the system’s kernel processes without openly affecting those processes in Windows architecture.
3. **Storage Management**: Windows OS comes with built-in Disk Management and Storage Sense. DiskManagement is a Windows system utility that is mainly used for advanced storage tasks, while Storage Sense can automatically free up drive space by getting rid of items that you don't need, like temporary files and items in your Recycle Bin.
4. **Memory Management**: Windows OS has System Memory Management built-in as a system utility. Windows offers such memory management options as physical and virtual address space to allow apps to run smoothly. To store the game’s HD images we’ll have to build a database so that the app can smoothly access and utilize them.
5. **Distributed Systems and Networks**: Distributed Systems and Networks can provide simple communication and coordination between each other. We could build a client-server distributing system to have each client app rely on the individual server so that each client app can be developed according to that client's system's strengths. We would also need a reliable server network to allow multiple clients to connect to one server to play the game at the same time.
6. **Security**: Windows Defender is a built-in Windows security service. To ensure user data is safe we would have to encrypt all the data sent in and out through a reliable encryption technique. Using VPN is another way to help protect user information from being used for nefarious purposes.