

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 |
| --- | --- | --- | --- |
| 1.0 | 9/19/2021 | Matthew Trembley | Added information to all headers. |

**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 Gaming Room wants to develop a web-based game that is based off their Android app, “Draw It or Lose It”. This game will have multiple teams of a few people each that will play 4 rounds. When an image pulled from a stock photo database appears, a team will have 30 seconds to guess what it is as it is drawn. If the team fails to guess correctly or at all, each other team will have the chance to guess with 15 seconds on the clock. The Gaming Room would ultimately like to expand availability of this game to other platforms, thus requiring to be written in more than just Android-applicable languages.

## [Design Constraints](#_2et92p0)

There are a few constraints which are listed below:

* A game will have the ability to have one or more teams involved
* Each team has 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. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

These are only the constraints of the game. On the development side, there is currently only support for Android. This will require the team to rewrite “Draw It or Lose It” in other languages to support more platforms.

## [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)

Below in the UML Diagram, the following classes Game, Team, and Player inherit traits from the class Entity. This will allow for these traits to be easily created within their respective scope. Class ProgramDriver uses the SingletonTester class for easy object creation, but also runs the whole program. The largest class, GameService, is where the program will have any search function for teams, games, or players and display an output, but also retrieves different variables throughout the program.

**"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** | Mac has flexible terminal commands that will enable the server to be configured. Mac is tending to be more expensive. | Linux is very cost effective. It is somewhat difficult to navigate. Like Mac, Linux has a command shell for simple server configuration and accessibility | Windows is somewhat as expensive as Mac but is very user-friendly. There are many software options available for Windows. Windows also has command prompt. | Mobile Devices are very inexpensive. Mobile devices have challenges in creating the game that is compatible with all mobile OS. |
| **Client Side** | A lot of time is necessary to support Mac users. Some moderate skill is needed, and Mac tends to be more expensive. | A vast amount of time is needed to support users Linux. The experience required is near mastery, but other than that it is very cost effective. | With Windows, minimal time and experience is needed to support the OS users. Cost is about the same of Mac OS. | Maximum time is needed to support many different mobile device users. Mobile device OS platforms are difficult to perform on other devices, making it more expensive to develop. |
| **Development Tools** | Mac is similar as most OS, where the most common of programming languages and IDE’s that are used JavaScript, CSS, HTML, Python, Swift. Some IDE’s are common, like Eclipse and Visual Studio. | Linux uses a few languages like Ruby, Java, Python, CSS, JavaScript and a few others. Common IDE’s are Visual Studio, command prompt, nodejs. | Many common languages are used on Windows, like Python, Java, JavaScript, C++, C#, HTML, and many more. The most widely used IDE is Visual Studio. | Commonly uses languages seem to be Python, Java, C++, and JavaScript for mobile devices.  Many development tools are used like Visual Studio, node.js, and GitHub. |

## 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**: The Gaming Room’s application “Draw It or Lose It” would best be used on a Windows operating system. A Window’s server operating system has both a protected and supervisor mode, which helps enhance the reliability if the various computers it will be ran on. On top of that, it has the capability to fulfil any request a client may have on the specified network. This feature alone creates an atmosphere for performance for the system and it’s user. At a low cost and a minimum level of training needed, it will be very easy to use.
2. **Operating Systems Architectures**: Windows server operating system architectures have a vast menu of techniques for memory and file management, which will essentially allow the user to allocate the memory optimally.
3. **Storage Management**: Storage management with Windows is very easy, as Windows will continually keep track of items and some broken registry files that can be cleaned out per system on a user-decided schedule. This will allow the system to have ample space needed to modify the program without the worry of running out of storage.
4. **Memory Management**: Window’s utilizes a built-in memory management utility. Supported files and tasks will be moved from random access memory to space on a hard drive to free up RAM for any other possible tasks.
5. **Distributed Systems and Networks**: A distributed system that uses a network is a fantastic way to implement new software and programs for the specific systems on the network. The main feature of this is a simple line of fast communication between different machines on the same network. A strong network is highly suggested as this game will be using multiple different clients to connect to one single server to play one game as a group.
6. **Security**: Multiple layers of security will be the most helpful in preventing and data loss or breaches. This may include some form of encryption, secure multi-factor sign in, and anti-robotic puzzle solving. The simple installation of multiple layers will deter most attackers, and even those who try will have a tougher time. Windows server OS have a built-in protection Windows Defender, which helps detect and take care of malware.