

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 | 11/27/22 | Ezra Maynard | Added evaluation summary. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wishes to create a web-based game that will work on multiple platforms that is based on their current game only available on Android currently. The game will be called “Draw It or Lose It”. The game consists of four rounds that are one minute long in length. Images will be drawn from a large library collection and the teams guess the puzzle(a phrase, title, or thing). If the original team fails to guess correctly, then the opposing team will have 15 seconds to guess.

## [Design Constraints](#_2et92p0)

1. Each team has multiple members assigned to them
2. The game needs to allow multiple teams to be involved
3. Game and Team names must be unique and allow users to check if the name is already taken or available
4. Only one instance of the game can exist at a time
5. Needs to run on multiple platforms

Since The Gaming Room wants this game to run on multiple platforms, we will need to convert some of the code from the Android application to run on Mac, Linux, and Windows and other mobile devices. For Apple devices we should look into using Swift and for other operating systems it would be best to inherit other languages and attempt to re-use our existing Android code.

## [Domain Model](#_8h2ehzxfam4o)

Entity has a relationship between Game, Team, and the Player class. Team, Game, and Player have a “is a” relationship with Entity. All of these classes inherit and obtain information from the Entity class. Entity is a superclass and shares attributes “id” and “name” with these other classes. Team and Player is a “has a” type while Game has a Team and GameService has a Game these are represented by aggregation. GameService obtains a reference of the Game class, Game has a reference of the Team class, and Team also has a reference of the Player class.

This makes sense since GameService needs a Game, the Game needs a Team, and the teams need Players. ProgramDriver uses SingletonTester.

"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Very flexible terminal commands that allow configuration of a server and its access protocols.  Most expensive option.  Popular for web hosting. | Very similar to Mac, flexible terminal commands that allow configuration of a server and its access protocols. Less expensive than Mac.  Very secure and therefore very popular for web hosting services.  Less software options compared to Windows. | One main advantage of Windows is the amount of software that is available compared to other OS. Less flexible terminal commands that other OS.  More susceptible to viruses. Close platform. | Worse hardware than other options meaning a smaller server with less performance. Ideal if the server is immobile. Cost effective approach with poor security. |
| **Client Side** | Cost is similar to Windows. Expertise and time required is moderate. | Very low cost. Expertise and time required is maximum. | Cost is similar to Mac. Expertise and time required is minimum. | This can be tricky to get full functional compared to other options. Can be time consuming. Does provide flexibility to view updates at any place or time since it is mobile. |
| **Development Tools** | Swift is a major language for Mac with the others included. HTML, CSS, Java, Python, JavaScript. Tools include PyCharm, Eclipse, VS Code, Git and GitHub. Databases include SQL, MongoDB, Cassandra. No Azure. Macs can run all languages. | HTML, CSS, Java, Python, JavaScript and many more languages.  Tools include PyCharm, Eclipse, VS Code, Git and GitHub. Databases include SQL, MongoDB, Cassandra. Azure. | HTML, CSS, Java, Python, JavaScript and many more languages. Tools include PyCharm, Eclipse, VS Code, Visual Studio, Git and GitHub. Databases include SQL, MongoDB, Cassandra. Azure. | HTML, CSS, Java, Python, JavaScript and many more languages. Tools include PyCharm, Eclipse, VS Code, Git and GitHub. Databases include SQL, MongoDB, Cassandra. Emulators are helpful here like XCode and Android SDK. |

## Recommendations

1. **Operating Platform**: I personally would recommend we start developing on the Windows operating system. This is because there is a familiarity to it and most on our team have used it already for daily use. There is almost unlimited software and IDEs we could utilize and its pretty cost effective as well.
2. **Operating Systems Architectures**: Windows enables the use of Graphical User Interfaces (GUIs) and intuitively includes all required services for accessing system resources, utilities, and more. There are many Windows-based application such that include messaging, web services, graphics, and multimedia. It also has plenty of hardware options that will be compatible and has a built-in process manager.
3. **Storage Management**: Windows has many storage applications built into the operating system as well as lots of additional software and drivers you can add on top of the included applications. One major storage management system is called storage sense, which lets you change and manage files on your hard drive and tells you how much storage is taken up. It has a disk utility that allows the deletion of unused or old files that may be lurking taking up storage on your drives. It also includes a disk cleaner that defragments drives which improves the performance of the drive. You also have the option of using cloud-based storage like Google Cloud, or OneDrive. Windows also allows you to easily store everything using File Explorer which is an advanced GUI for the creation, deletion, and viewing of files and folders on your drives.
4. **Memory Management**: We will have to store and create a database or library for our pictures when we create this game. Ideally, we will be adding more pictures as the game expands in the future. Windows has excellent memory allocation which will allow us to store the pictures outside of the default picture folder. Its virtual memory functions allow reservation of memory for processes ensuring that we have enough memory for the game to function without another process stealing our needed memory. It also allows caching of our most used files to ensure a fast response rate for our game and the most efficient way to access those files.
5. **Distributed Systems and Networks**: I have found Visual Studio and Visual Studio Code to be excellent IDEs for cross platform development. Visual Studio Code runs on mac, Windows, and Linux and has unlimited plugins to assist cross platform development and would be my first choice in IDE. For Network, we need to ensure our developers are hard-wired and that our servers are maintained well by a reputable company that includes battery backups in the case of a power outage. We also need to ensure that the servers can handle as many clients as we expect the game may have, or better yet, have servers that will be able to be expanded in the future once the game gains popularity.
6. **Security**: Windows has plenty of operating system level software features including a virus scanner and automatic removal of viruses once they are found. In addition, it has a built-in firewall although we would want to avoid having to rely on that and just stick with VPN’s on all company devices. Windows also has been known to patch security updates very quickly when they do find and vulnerabilities and this can happen at a set time, so we are not interrupted while we are working. To store user data and information we should use a reputable company and not keep anything on our personal network due to the liability and security hazards. We need to ensure we are using a role-based security system and only giving admin or access to files when they are absolutely needed for what the employee is doing. In addition, we will use two factor authentication for all company devices and accounts.