

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

Roderick Fisher

## Table of Contents

[CS 230 Project Software Design Template](#l6ti7uoag22u) 1

[Table of Contents](#j0zll) 2

[Document Revision History](#grjogdjh5fi8) 2

[Executive Summary](#sbfa50wo7nsh) 3

[Design Constraints](#et92p0) 3

[System Architecture View](#ilbxbyevv6b6) 3

[Domain Model](#h2ehzxfam4o) 3

[Evaluation](#o15spng8stw) 3

[Recommendations](#m8aleynsvzvc) 5

## [D](#grjogdjh5fi8)[ocument Revision History](#grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <mm/dd/yy> | <Your-Name> | <Brief description of changes in this revision> |

**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.

## [E](#sbfa50wo7nsh)[xecutive Summary](#sbfa50wo7nsh)

The Gaming Room wants to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is currently available in an Android app only. The main goal is to create a game that has multiple teams consisting of more than one person. A game consists of four rounds of play lasting one minute each. 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](#et92p0)

A design constraint in creating this project would be creating a game that can support multiple platforms. In order to achieve this we will need a team that is knowledgable in multiple languages to rewrite code that will be provided from the Android application. We would use Swift or React to rewrite code and/or make code cross-compatible for Apple products. We would also need to create a game that can allow more than one player and more than one team with unique names and the ability to see if a name is already taken. We will implement Singleton classes into our code to ensure that all requirements are handled correctly.

## [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.

## [D](#h2ehzxfam4o)[omain Model](#h2ehzxfam4o)

The Entity Class creates the relationship between the Game, Team, and Player classes. GameService ties all the information together and determines if all perimeters are set. Game is a reference of GameService. These classes all inherit from the Entity Class. The Diagram illustrates how a Player gets added to a Team, and a Team gets added to a game thus making the Entity a superclass.

**"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.**

## [E](#o15spng8stw)[valuation](#o15spng8stw)

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** | Create software using swift. Good application compatibility across multiple devices. allows for an extremely easy creation of features comparable to Win group policies for Mac clients, much more easy then to do the same for Win clients on a Windows server. | Provides an environment that's powerful, stable and reliable yet easy to use. Linux servers on the Internet have been running for years without failure or even being restarted. Secure and free. | More secure, but can be costly. Windows also has proprietary tools that are used. You can expect more resources and support using windows. | This option needs a wide variety of skills, besides it’s a lengthy process. If you need a wide choice of customization and have the required expertise in your team, you can consider it. |
| **Client Side** | Mac cost more than Windows and the user is forced to buy a Mac system built by Apple. Users are also locked in to the Apple ecosystem when using Mac products. | The good thing about Linux is that it is open sourced. Since it is open sourced, it is much easier to gain and build on. Clients will benefit from more features. | Great platform for developing web sites and native windows apps. Especially if you use Visual Studio or Visual Studio Code. | Mobile devices generally phones and tablets. Developing for these form factors is generally very similar. Phones have very limited screen space, and tablets, while bigger, are still mobile devices with less screen space than even most laptops. Because of this, mobile platform UI controls have been designed specifically to be effective on smaller form factors. |
| **Development Tools** | Objective-C or SWIFT (M1 chip Macs) are the languages most commonly used in Mac OS Programming. | C, C++, CSS, Java, JavaScript, HTML, PHP, Perl, Python, and Ruby are all supported by Linux. | Microsoft, a Windows company uses C++ and it widely used in Windows development. You can use other languages as well like Java. | SWIFT is most widely used increasing mobile apps and can make coding cross platform easier. Java is also used in a lot of mobile development. |

## 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**: Windows would be the best platform for development.
2. **Operating Systems Architectures**: Since Windows can use an interactive GUI, this will allow users to set up the environment to their needs. Windows also has great online support to help solve problems.
3. **Storage Management**: We can use apps such as dropbox, google drive, and OneDrive for storage management. We could also use something like SQL.
4. **Memory Management**: Windows does a great job at having memory limits. This means that we can use Windows to manage the amount of memory our application uses.
5. **Distributed Systems and Networks**: I think that going cloud based would be best. This will allow users to use the application on whatever platform they are using. Using a cloud based network would also allow for easier backend.
6. **Security**: In windows we will be able to create firewalls and encryption. This will give us the ability to make it harder for our application to be hacked and will make security more stable. Every operating system has its own flaws when it comes to security so implementing this features are crucial.