

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 07/15/2023 | Chase Carter | Executive Summary |

**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 software is for an interactive application that is ultimately a guessing game. This game is called “Draw It or Loose It”, the objective of the game is to complete a puzzle. This puzzle will randomize an image from the bank of images. The different teams in the game will have 4 rounds and they will be 1 minute each. The team that does not correctly guess the puzzle the opposing team will have a 15 second chance to steal.

## Requirements/Design Constraints

There will be one or more teams in the game, each team will have more than one player. The team’s name cannot be duplicated, so check and see if that specific name has been taken. Each game will be different once will not be like another, use the specific game, team, or player.

## [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)

GameService, Game,Team, and Player are classes. They can use all the attributes within each other or use none of them. They are all associated with one another, the Entity class is the superclass that contains the main attributes. The program driver class uses the singleton to drive the packages for the other classes.

**"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** | From the terminal make the server flexible, this is so changes can be made later. Cost more on average for the server. Very good security for the servers. A disadvantage of Mac is that there is only Swift as a primary application builder. | From the terminal make the server flexible, this is so changes can be made later. Linux should be cheaper. A disadvantage is that there is not as much support as the others for this OS. | This operating system has the most available resources in software.  Like Linux, there is not as much support as Mac. | The mobile device has certain specifications that must be adhered to in the software.  A disadvantage is their phone security is an issue considering that they are easier to get into than computers. |
| **Client Side** | The operating system must be able to be used by the web browser as well as the application version. Higher costs in software, this is because the OS is more specialized and fewer resources in software are available. | Much more cost-efficient, and securely easier to access. There are fewer programmers for Linux to get assistance from. | Great loading time, easily edited, altered and changed.  There are a lot of programmers that can assist in building. | Most of the traffic will be on phone applications. This software will be the most used. The accessibility to use the application will be wide because the client will have more access because everyone always has their phone. |
| **Development Tools** | Mac can run in all programming languages. For the application swift will be used. | Eclipse will work as an IDE, and PyCharm will also work. | Eclipse and PyCharm will work as an IDE. | The application can be run on Swift for Mac and Android for everything else besides Mac. |

## 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 use Eclipse IDE and Windows OS. I am familiar with these, also there is a lot of support for both.
2. **Operating Systems Architectures**: Windows allows the user to use the accounts and substructure of the OS. Windows applications will use the operating systems resources.
3. **Storage Management**: Windows allows the user to choose where the program and its contents are stored. There are built-in storage management systems in places the is in the design of the OS.
4. **Memory Management**: The whole point of the game is to use a picture from the provided data. So there will be a place inside of the OS where the pictures will be stored to be readily accessible for the game application.
5. **Distributed Systems and Networks**:

The best way to distribute the system is by being able to use them in a cross-platform way. This way they can be used by all the OS but it will be limited to the OS that the files are being converted into.

1. **Security**: I chose to use Windows through the Eclipse IDE. Windows has built-in software that is provided through the OS. I would recommend more security for each individual OS. There is a variety of software out there that can provide an extra layer of protection besides the built-in version for Windows.