

Draw It or Lose It

# **CS 230 Project Software Design**

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 | 01/21/23 | Dominic Drury | New Design Document |
| 1.1 | 02/02/23 | Dominic Drury | Reviewed information present for Evaluation section |
| 1.2 | 02/14/23 | Dominic Drury | Reviewed information for Recommendations section |

**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 is looking to develop a web-based game that serves multiple platforms based on their current game Draw It or Lose It, which is currently only available on Android. The game must have the ability to have one or more teams involved, each team will have multiple players assigned to it, the game and team names must have unique names, and only one instance of the game can exist in memory at any given time. The game consists of four rounds, each lasting a minute. Drawings taken from a large library of stock images are rendered at a constant rate and be complete at the 30 second mark. If the team fails to guess the puzzle before time expires the remaining teams have one guess each to solve the puzzle with a 15 second time limit.

## Requirements

* A game will have the ability to have one or more teams involved
* each team will have multiple payers assigned to it
* Game and team names must be unique to allow users to check if another name is in use
* Only one instance of the game can exist in memory at any given time
* The game must be able to function across multiple platforms
* Game involves teams guessing the guessing the puzzle (Phrase, Title, or Thing)
* Game is four 1-minute rounds
* Drawing renders at a steady rate and completes after 30 seconds
* If the guessing team fails, the other teams have 15 seconds and one guess to guess the puzzle

## [Design Constraints](#_2et92p0)

* The images will be taken from the existing Android version library
* The game needs to accommodate multiple teams, each with multiple team members
* Each game and team need to have unique identifiers
* The game will need to be identical across multiple platforms and devices

## [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 entire program is held within the package com.gamingroom. The ProgramDriver class contains the main function and because of the linking arrow it will use the SingletonTester class to test the program. The parent class Entity is the parent class of Game, Team, and Player classes. The GameService class relates to the Game class, Game relates to Team, and Team relates to Player all in a 0 - many relation. The Entity function contains the private attributes id and name, private operation Entity, and the public operations Entity, getId, getName, and toString. The GameSerive class contains the private attributes games, nextGameId, nextPlayerId, nextTeamId, and Service, the private operation GamerService, and the public operations getInstance, addGame, getGame using id, getGame using name, getGameCount, getNextPlayerId, and getNextTeamId. The Game class contains the private attribute teams and the public operations Game, addTeam, and toString. The Team class contains the private attribute players and the public operations Team, addPlayer, and toString. The Player class contains the public operations Player and toString.

**"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** | The Mac OS server allows for a mixed-platform environment and the ability to configure, deploy, and manage network services. The main weakness is the limited software support for MacOS when using third party programs. MacOS can also become expensive when running more demanding servers. | The biggest advantage and weakness for Linux steam from the same place. The OS allows for extensive customization, though this freedom comes with a steep learning curve to use. The cost is next to nothing for most distributors, though some do charge a fee including extras. | The Windows OS is cheaper than Mac but more expensive than Linux but has more software support than both Linux and Mac. | The weakness for developing with mobile devices is that they are unique from the PC OS. The costs and expertise needed vary from brand to brand. The biggest advantage is that the audience for mobile devices are second to none, with most people owning some sort of mobile device. |
| **Client Side** | The cost is high for the MacOS as most of the cost that pays for the OS comes from buying more expensive hardware. The time required for using the MacOS. The expertise is moderate for MacOS users, requiring the same amount of expertise as its competitors. | The cost is cheap, only needing to pay a fee when gaining added bonuses provided by distributors. The time and expertise are extreme when compared to Mac and Windows. | The cost for developing with the Windows OS is cheaper than Mac but more expensive than Linux. With the wide range of users, the OS is as user friendly as its main competitor Mac, so the time and expertise needed are very low. | For users the cost, time, and expertise needed are often the lowest among OS. For developers the cost can range but the time and expertise needed to port to mobile devices is the highest out of the OS listed here. The controls are limited and the specs for mobile devices range heavily and have massive restrictions compared to PC OS systems. |
| **Development Tools** | The MacOS supports almost every coding language and has a plethora of IDE’s. Development tools are also in great supply, with eclipse, visual studios, and many others fully functional. | Similar to Mac and Widows, Linux supports pretty much every coding language and has a wide arrange of IDE support from most of the major IDE’s and development tools like Eclipse, Visual Studios, and others. | Just like with Mac and Linux the language support includes pretty much every coding language and similarly almost every major IDE and Development Tool has Windows support, even more so than Mac and Linux. | In the modern era mobile devices have gone leaps and bounds in mimicking the PC OS systems. While the hardware may come with higher restrictions, the software is far less so. Most languages, IDE’s, and development tools support mobile OS. |

## 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 recommend the Windows OS. It has an ease of time and expertise that will more than make up for the cost of the operating platform, it is widely used, and third-party support is the highest among the operating platforms.
2. **Operating Systems Architectures**: The architecture of Windows consists of Kernel-mode and User-Mode. The Kernel-mode takes the raw information and processes that data, then sends that information to the User-mode allowing the user the abstract view of the application.
3. **Storage Management**: Since the game will need to have support across platforms, cloud storage would be best, and Windows comes with a variety of cloud storage support with little restrictions on them.
4. **Memory Management**: Windows comes with a give and take between some ease of use and some restrictions. The OS automatically allocated 4 gb of memory for use of applications. 2 gb to the kernel and 2 gb to the application. Furthermore, the OS has a virtual address space for processing.
5. **Distributed Systems and Networks**: Windows distributed systems are cloud based and can perform package exchanges between different OS systems. With the wide range of support for windows, the various available frameworks for cross-platform communication will be relatively easy to implement.
6. **Security**: Windows has windows defender for both users and businesses to provide protection of personal data and information collected by the game and used by users.