

Draw It or Lose It

# **CS 230 Project Software Design**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc94723243)

[Table of Contents 2](#_Toc94723244)

[Document Revision History 2](#_Toc94723245)

[Executive Summary 3](#_Toc94723246)

[Design Constraints 3](#_Toc94723247)

[System Architecture View 3](#_Toc94723248)

[Domain Model 3](#_Toc94723249)

[Evaluation 4](#_Toc94723250)

[Recommendations 7](#_Toc94723251)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 10/06/2024 | Samuel Wilkins | First Draft |
| 1.1 | 10/16/2024 | Samuel Wilkins | Second Draft |
| 1.2 | 10/20/2024 | Samuel Wilkins | Third Draft |
| 1.3 | 10/24/2024 | Samuel Wilkins | Final Draft |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room commissioned CTS to design a web-based version of their Android-platform game Draw It or Lose It. The game should have multiple teams with multiple players per team. Every game only one instance, team, or player should be active at a time. To ensure this convention, the class GamingService which runs the entire game loop will be based on a singleton pattern.

## [Design Constraints](#_2et92p0)

To the extent the original game is an Android-platform app, Java code will be needed and need to be rewritten in other languages where necessary for other platforms used. Also, any current APIs will need to be evaluated for use in the mobile app version of the game.

## [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 UML for the proposed design is shown below.

**"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.**

## The Entity class serves as the parent class to Game, Team and Player which all inherit its attributes. The UML shows this inheritance with arrows. The ProgramDriver and SingletonTester are listed separately since they do not depend on the other classes. GameService is written at the end of Game because it implements the Game class but does not inherit it as seen in the first private constructor which employs the Game class.

## [Evaluation](#_2o15spng8stw)

Below I have included (in table format) the various strengths and weaknesses of each OS we might use as host/server/client in the Draw It or Lose It game limited to Linux, Mac, Windows, and mobile devices.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | OS X Server is available for Mac, but hosting can be challenging and costly unless the client purchases their own hardware.  The pricing for OS X Server is $499 USD for a max of 10 clients or $999 for unlimited clients. | Linux is the most popular operating system for web hosting. Its open-source nature makes maintenance and licensing more affordable compared to closed systems like Windows. Additionally, major cloud providers such as Google and Amazon often prefer Linux over Windows for their services. | Windows servers are user-friendly due to their GUI, compatibility with many office applications, and familiarity. However, license costs are high with prices ranging from approximately $1,100 to $6,200 per year. Additionally, hosting options for Windows servers are often more limited compared to Linux. | Mobile devices can function as personal web or file servers, but they are not designed for multi-user serving. Their hardware, such as RAM, is generally limited and lacks the scalability of blade servers. Additionally, the costs are uncertain since hosting tools would likely need to be designed in-house. |
| **Client Side** | Developing for Mac requires familiarity with macOS-specific development tools like Xcode and Swift.  Also, macOS SDK is written in Objective-C or SWIFT.  Finally, Mac represents a smaller market than other platforms like Windows which can present some difficulty in reception. | Development in Linux should be simple since it requires knowledge of common languages like C/C++, Python, or Java. GNU/Linux supports multiple users. However, developing for GNU/Linux might be limited since it is not so widely used. | Windows is typically developed using C# or .NET which are both common. This convention should make development relatively smooth.  Windows supports multiple users already.  Windows is the most popular development OS and widely accepted. | Mobile devices typically do not support multiple users.  However, designing a client application for Android or iOS is relatively straightforward. Since Android SDK is Java-based, code developed for Windows and Linux can serve as a starting point.  For iOS, development in Swift follows similar requirements to Mac, including specific hardware needs. |
| **Development Tools** | Mac uses Objective-C and SWIFT as development languages.  XCode is the most common IDE used for Mac development.  XCode is listed as $99 USD per year per user. | Linux development may employ C/C++, Java, or Python.  Python IDEs such as NotePad++ and PyCharm are free.  There are many C/C++ IDEs available but not necessarily for Linux. Eclipse would likely be chosen for this project. | Windows is primarily developed using C# and .NET.  Microsoft’s Visual Studio is immensely popular and compatible with many external tools  Visual Studio is available for free as a community version or paid with enhanced features at $45 – $250 USD per user per year. | Android SDK is Java based, and the most widely used Android IDE, Android Studio (Google’s official dev tool), is available as a free download  iOS’s Objective-C and SWIFT languages are almost exclusively developed in XCode.  XCode is listed as $99 USD per year per user. |

## 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**: Google Cloud Platform (GCP) is the system I would recommend based on its low-latency, security, and ability to support a high number of players.
2. **Operating Systems Architectures**: GCP actively incorporates numerous data centers globally, offers a private connection for hosted servers such as one running Draw It or Lose It, has multi-regional support and offers backup and recovery services that could protect games that crash including data like leaderboards, team names and win/loss statistics.
3. **Storage Management**: Since GCP is already a Google service, Google Cloud Storage is a natural choice that is quick and reliable and can be built upon later if the game becomes bigger.
4. **Memory Management**: Google Cloud Storage will cache frequently used data and has built-in processes for reducing memory during runtime and garbage removal.
5. **Distributed Systems and Networks**: Since Google’s cloud services operate out of so many different locations and their virtual servers are highly robust, outages and connectivity problems would be likely minimal. Also, the frequent backups would minimize any damage caused by such an outage.
6. **Security**: Google Cloud Platform uses encryption in stored and communicated data. It also implements great physical security where its datacenters are located and offers multi-factor authentication (MFA) services.