

# Draw it or Lose it

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 01/25/23 | Ryan Cooper | Final Completion of Executive Summary/Requirements |
| 2.0 | 02/05/23 | Ryan Cooper | Final Completion of Evaluation Chart |
| 3.0 | 02/19/23 | Ryan Cooper | Final Completion of Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

Draw it or Lose it, is a current Android only mobile game made by The Gaming Room. The game has teams competing to make a guess about an image that is being rendered. Each team will get a turn to guess first in 30 seconds. If the first team does not guess the image then the other teams will have 15 seconds to make a guess and steal the point. To grow the playability, The Gaming Room wants a web-based version that can run on multiple platforms. They are unfamiliar with web-based programs and have asked us to help with the development of the new web-based version. To meet the demand, a server operating system like windows will need used. This will give the best user experience for the player because of the quick speed and fast turnaround time for updates.

## Requirements

The Gaming Room has requested the following requirements. The game will allow one or more teams to play in a single game. Teams will have multiple players assigned to them. For each game, the team names need to be unique and a way for the user to check that their team name is not in use. Lastly only one instance of each player, team or game may exist at any given time.

## [Design Constraints](#_2et92p0)

Since the game will be web-based, it will need to run on multiple operating platforms like mobile phones, desktops, and tablets. The server will have to be secure so that user information is not leaked. Operating systems will have to be researched on what is the best to maintain and run for the server. Since the game will have multiple games running, the server and network will need to be fast enough for all the users. The design and function should be the same at the current mobile build of the game that is on Android. The current Android version can be used as a reference.

## [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 Entity class is inherited by the Game, Team, and Player class. This is so there will be a unique id and name for each child class. These inheritances are shown by the arrows that point upward. That is why there is an upward arrow for the Game, Team and Player class pointing to the Entity class. Another way to think about the relationship is each Game, Team and Player is-a Entity. Then each of the classes have line connecting them horizontality. These are the has-a relationships or know as composition. Meaning that each Game service has-a game and each game has-a team and each team has-a player. Lasty, encapsulation is used in the UML diagram with the private variable in each class. The way they are accessed is through getter and setter methods. This way other classes cannot change them without permission.

**"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** | MacOS has an unlimited user license with ways to increase processing power with having multiple machines. macOS can only run on apple hardware so price can be expensive. | There are many distributions/versions since it is open source. It is also free to use but less popular so extensive knowledge will be needed | It is a licensed product so will need to be paid for. Windows is closed source so not too many changes can be made. Most people use it so less knowledge is needed. | Server can scale with demand by buying more hardware if and when it is needed. Security and compatibility are a concern. |
| **Client Side** | Mac is a high cost for users since users must buy apple products. An average amount of time and expertise is needed to develop for macOS. | Since Linux is open source, the cost is free for users. Most people are not familiar with Linux, so a lot of time and expertise is needed for the application development of this operating system. | Windows does need payment to use but is a one-time fee, thus cost is low. Time and expertise are low since it is popular for most developers. | Development time is high because there are multiple different systems to consider. Cost and expertise are also high since development is needed for both Android and Apple. |
| **Development Tools** | Swift is the main language for macOS. Visual Studio code or Xcode are good choices of IDEs for Mac. Popular web development languages are HTML, CSS, and JavaScript. | Almost any language can be used to make programs for Linux. Visual Studio code might be the best IDE for Linux. Popular web development languages are HTML, CSS, and JavaScript. | Almost any language can be used to make programs for Windows. Visual Studio code might be the best IDE for Windows. Popular web development languages are HTML, CSS, and JavaScript. | Swift is the main language for iPhone development. Java or Kotlin is used to develop on Android. Xcode and Android Studio should be used for IDEs. Popular web development languages are HTML, CSS, and JavaScript. |

## Recommendations

1. **Operating Platform**:

The Linux operating platform is widely recognized as the most maintainable option for server environments. While the expertise level required to manage Linux may be higher than some other options, the benefits make it an excellent choice for hosting Draw It or Lose It. Not only is it cost-effective, as each server is free to use, but it is also highly customizable and offers a broad range of support and troubleshooting resources. With a large and active user community, troubleshooting issues is made easier and can be resolved quickly, leading to increased uptime for the game. Moreover, Linux is an incredibly popular operating platform, and its prevalence means that it is highly compatible with a vast range of software and hardware configurations. This compatibility makes it an excellent starting point for hosting Draw It or Lose It, and its flexibility means it can be expanded upon later with different distributions to meet the growing demands of the game.

1. **Operating Systems Architectures**:

For the client's system, I would recommend a multi-tier architecture. This is because Draw It or Lose It is a web-based application that will be accessed by multiple users on personal computers or mobile devices. A multi-tier architecture would provide greater scalability and performance for the system, as well as improved security and reliability. It would also allow for easier maintenance and updates, as changes can be made to specific tiers without affecting the entire system.

1. **Storage Management**:

For Storage management on the software side, Linux has Logical Volume Management (LVM) that allows users to create logical volumes, manage storage allocation, and add/remove storage devices without disruption. Another storage feature of Linux is RAID (Redundant Array of Inexpensive Disks) configurations. It allows users to increase storage capacity, improve data reliability, and provide redundancy to stored data. For hardware, a solid-state drive should be used over a hard disk drive because of the faster read and write speeds. Also, solid-state drives are more reliable since hard disk drive have more moving parts that can fail over time.

1. **Memory Management**:

Linux will determine how much memory is needed for each process for the Draw it or Lose it game. Some examples of this is swapping or virtual memory, which means Linux will swap unused data to the hard drives so more RAM is available to use for processing. Linux as dynamically allocates memory as different programs need it. The allocation and deallocation is all taken care of by Linux so it will be done right and worry free.

1. **Distributed Systems and Networks**:

Linux can work as a disturbed system and network so multiple computers can run just like one machine. They can be in different places so if connectivity will be better for most users, given the getter area coverage. Outages are less of a problem because itis unlikely that all systems will go down over a spread-out area. Thus, the program will almost always be online. REST is a good system to build Draw it or Lose it. This is because it could be used to allow users to save and retrieve their game progress, to provide social features like leaderboards or player profiles. Since functions are standardized, most operating systems and browsers can be used to play the game through REST.

1. **Security:**

Linux is widely known for its robust security features, making it a preferred choice for many server environments. The security features of Linux include user permissions, access control, encrypted file systems, firewalls, and logging. The system's package management provides a secure and centralized way to install and update software packages, ensuring that the system is running up-to-date and secure software. Additionally, there are many other security tools and best practices that can be used to harden a Linux system, such as implementing strong passwords, regularly applying security patches, and using secure communication protocols like SSH for remote access. These features make Linux a highly secure operating system and an excellent choice for running servers for the Draw It or Lose It game.