

# 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 | 07/18/2025 | Ivan Valdez | Final software design |

**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 currently offers an Android based app called Draw it or Lose it. They want to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It. Draw it or lose it is comprised of multiple games, multiple teams with multiple players. The game requires each game and team name to be unique to function properly. The game contains a large library of stock images where the game draws from. The staff at the Gaming Room hired CTS because they don’t know how to set up the environment.

## Requirements

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

Android, IOS, and the internet have different software development kits. That is why it must have web-based software to support the different users in real time. This leads to further restrictions as an efficient way to use servers to support the new number of users must be found. Another design constraint will be the uniqueness of the team names and players, which may lead to issues as the game is played by a broader audience.

## [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 ProgramDriver class contains the main method, and it uses directed association to test there is an instance of GameService in SingletonTester. Entity class is the parent class from which Game, Team, and Player classes inherit from. A Team can have a Player, but not vice versa. A Game can have a Team, but not Vice versa. GameService can have a Game, but a Game cannot have a GameService. GameSerive can only have one instance of each Game at a given time. Each Game can only have one unique team at a given time. A Team can only have one of each unique player at a given time.

**"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 must 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** | Mac is known for its stability and performance as well as security operating on a Unix environment. The downside would be that Mac is not the best choice for large scale developments due to cost and hardware limitations. | Linux is a good choice for server-based software. It has excellent stability, is cost-effective, and highly customizable due to it being open source. The downside is that it requires more technical expertise to set up compared to its competitors. | Windows Server is a common platform for web applications as well as server deployments. It offers excellent integration with Microsoft tools and offers multiple web server options. The downside is that licensing fees tend to be high. | Mobile Devices don’t host web applications. They use any of the other platforms in the background of apps. They tend to have easy deployment of applications. Limited hardware and capabilities can result in scalability issues and security issues. |
| **Client Side** | Xcode is offered on the App Store for free, but the cost of Mac tools and hardware tends to be more expensive than competitors. Mac consumes more time to develop and open for multiple platforms. Given that the game will launch on multiple platforms, teams with expertise in cross-platform developments will need to be considered. | Development for Linux usually keeps costs low in regards of licensing fees. but requires a higher expertise level for specific applications. Given that the game will launch on multiple platforms, teams with expertise in cross-platform developments will need to be considered. | Development of Windows involves Visual Studio. Costs of licensing vary on the edition and team size, but expertise in different languages like C# are required. Given that the game will launch on multiple platforms, teams with expertise in cross-platform developments will need to be considered. | Mobile devices will require platform-specific development. Cost will vary as expertise in Swift and Kotlin will be required. Certain tools and cloud services will require licensing fees. Given that the game will launch on multiple platforms, teams with expertise in cross-platform developments will need to be considered. |
| **Development Tools** | The programming languages are Swift and Objective-C. The IDE’s are Xcode and AppCode. | Linux supports various programming languages (C, C++, Python, Java, etc.) and popular IDEs such as Visual Studio, IntelliJ IDEA, Eclipse, and PyCharm. | The primary IDE for windows is Visual Studio. It supports languages such as C#,C++, NET, and more. | IOS runs on Swift and Objective-C using Xcode as the IDE. Android runs on Kotlin and Java. The IDE for android is Flutter and Android Studio. |

## 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**: An appropriate operating platform that I would recommend is Linux. Linux is used for server applications because of their support for web applications. Not only that, but it is highly customizable to allow developers to adapt it to their requirements. Given that it is open-source, there will not be an issue finding competent developers to work on the project.
2. **Operating Systems Architectures**: The operating systems architecture that I would recommend would be a multi-tier architecture. By utilizing multi-tier architecture, we can separate application, presentation, and data from the application. This will allow for efficient scaling.
3. **Storage Management**: Seeing as how the game must render images from a large library of stock images, I would recommend a high-performance storage, such as an SSD. A Solid-State Drive will have higher read and write speeds when compared to a traditional hard drive. A second option would be to implement some form of cloud storage.
4. **Memory Management**: Linux has excellent memory management in which it uses a paging system. The paging system allows physical memory to be used efficiently by swapping unused data to disk by using virtual memory. This would allow memory to be optimized and will allow for smooth operation of Draw it or Lose it.
5. **Distributed Systems and Networks**: To allow communication between different platforms, a RESTful API is recommended. RESTful APIs allow for communication over the internet for applications using HTTP. Linux offers excellent support for RESTful APIs and have easy implementation.
6. **Security**: Linux offers security features such as firewalls, and they are known to be less vulnerable than other operating platforms. To protect the company’s security and its users, SSL/TLS encryption would help to ensure the data between the server and clients is safe.