

<Draw It or Lose It>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/23/22 | Gary Landstrom | Executive summary update. Design constraints update. Domain model update. |

**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 Draw It or Lose It team is unsure how-to set up the environment for the application and have also expressed concerns with streamlining the development of Draw It or Lose It. Draw It or Lose It is currently limited to the Android market, the port to a web-based game must have the ability to host multiple teams with multiple people on each team. Draw It or Lose It also must have unique player names and the option to select a team, most importantly only one instance of the game can exist in memory at each time.

To solve the Draw It or Lose It team’s problems we will need a few things:

* A dedicated team of JAVA/C++ developers.
* Unique identifiers for each game instance, player, and team.
* The ability to view the existing code for Draw It or Lose It, this could speed up development.

## [Design Constraints](#_2et92p0)

* Draw It or Lose It 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 the name is used when selecting a team name.
* Only one instance of Draw It or Lose It can exist in memory at any given time.

Most of the Draw It or Lose It team’s concerns involve having more than one instance of something, whether that be a team, username, or active game in memory. Luckily for us, these issues are easily addressed by implementing unique identifiers for every game, player, and team in Draw It or Lose It. Beyond this, we can implement a system that ensures that every team Id is unique and allow x number of teams per game. We can implement a similar system for players.

## [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)

In the UML diagram below, we can see that Player, Team, and Game are all related to Entity. All three are inherited from the Entity class, and if we look closely, we can see that Game, Team, and Player all have remarkably similar attributes. We can also see the relation between Player to Team, Team to Game, and Game to GameService. This is an excellent example of Aggregation as in the below figure GameService references Game, Game to Team, and Team to Player.

**"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** | Mac features very flexible terminal commands for server configuration, changes, and access.    The Mac OS is currently a popular choice for hosting software over the web.    The Mac OS features many options for different web hosting needs; this would be great for Draw It or Lose It.    Mac allows for easy integration from computer to mobile devices, saving some money.    The major downside of Mac is the cost of development and the lack of flexibility. | It also features a flexible terminal like Mac at a fraction of the cost.  Linux is a secure option to develop in and one of the most preferred.    One of the greatest advantages is security; security flaws are often caught early. Linux is among the most preferred operating systems for web hosting today.    The greatest weakness is finding applications that support the web hosting needs is how challenging it can be. This could be an expensive issue for the Draw It or Lose It team. | Windows is the most popular operating system today; as such it offers the most software to work with. Offering developers an abundant pool of choices for web hosting options.    Windows is the dominant choice opening Draw It or Lose It up to the largest amount of users possible.    Windows features a shorter loading time, as well as consumer-friendly systems. However, it is by far the most susceptible to viral attacks and lacks adequate tech support for such a large platform.    Windows is an expensive choice but will expose Draw It or Lose It to the most significant amount of potential players. | Best implemented with an immobile server that can be tracked, more flexibility is found within the other operating systems.    Mobile applications are a popular choice because of their accessibility (Everyone has a mobile device these days.).    Mobile applications feature the greatest reach, with a cost-effective development cycle.    The weaknesses are varying performance from one device to another, affecting gameplay in a time-sensitive environment. IOS devices are very secure, but Android devices are not secure due to their open nature. This issue could prevent cross-play between IOS and Android. |
| **Client Side** | The development of Draw It or Lose It on Mac will require moderate expertise. Finding Swift developers will be easy, and the Draw It or Lose It team will find they do not require the most expensive developers to complete the application.    A moderate amount of time will be needed (roughly six months); this is because of the inflexible nature of the Swift os. Because Swift runs on both Mac and iPhone, it will be cheap and fast to port Draw It or Lose It.    It will be expensive to develop on Mac; this is primarily because of the inflexibility of the Mac os. | Linux will be the cheapest option to develop on because of its open nature. However, this will be the most challenging option to monetize though.    Linux will also be the hardest to develop, requiring the most expertise and time to complete the project. Because Linux is the cheapest option, spending more money on the development team behind the application is advisable. | The cost associated with developing Draw It or Lose It on Windows will be similar to Mac. The cost similarity is caused because Windows is closed source and will require a strong development team to complete a functional application.    Because Windows is the most popular os for gaming, there are abundant resources on the topic. The development cycle for Windows will be among the shortest options for the Draw It or Lose It, team.    We believe moderate expertise is needed to complete the application on the Windows platform; this will help minimize cost and the time to develop. | Two development teams will be needed to release Draw It or Lose It on IOS and Android. Even though two separate applications will need to be developed, minimal time will be needed to release on the mobile platform.    The cost will also be minimal as most of the Mac application will translate directly to IOS; the same development team can be used for both Mac and IOS. It may be challenging finding an adequate Android development team though.    The development of the IOS version will require moderate expertise using Swift. The development for Android will require minimal expertise due to the open nature of Android. |
| **Development Tools** | For development on Mac, the most popular IDE is XCode. This is an accessible and easy-to-use tool that the development team will be familiar with. Swift will be the language used to develop the application on Mac; the same is true for IOS making porting cheap and fast. | The best IDEs for Linux development are IntelliJ IDEA, Eclipse, and Apache NetBeans; this is a prime example of the flexibility and skill required to develop on Linux. The strongest language for the application will be C++. This is because of how fast C++ is when implemented correctly; it remains the most popular choice for game developers. | The most common IDEs used in Windows development are Microsoft Visual Studio, Atom, and Eclipse, all of which are strong choices for this project. Much like Linux, Windows is not confined to a small set of languages; it is recommended that C++ be used to develop Draw It or Lose It because of its speed. Staying consistent with languages utilized will save money on development. | IOS will utilize Swift and XCode just like Mac; this will keep the cost of porting one to the other cheap and very fast.    The popular IDE options for Android development are Android Studio and Eclipse, much like Linux and Windows. It is recommended to keep the development of this application across platforms as consistent as possible. Utilizing C++ for Windows, Linux, and Android will allow the development teams to cannibalize code and speed up development. |

## 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**: The best operating system for the Draw It or Lose It application is the most popular and accessible. My recommendation is the Windows operating system; this OS is more popular for gaming when compared to IOS and provides a solid foundation for Draw It or Lose It to grow.
2. **Operating Systems Architectures**: Windows features 32bit and 64bit systems, both of which provide a compelling environment for system operations, computing, gaming, and development. It makes development with tools like Unity and the Unreal engine easy and provides the greatest range for compatibility and support.
3. **Storage Management**: If the Draw It or Lose It team decides they need physical storage, SSDs should be used. They offer incredible speeds and more longevity than their HDD counterparts. The best option would be cloud-based storage; this is the most flexible option as more space can be purchased as needed. The data can also be backed up in case the worst happens, and the company needs to use the backup copy
4. **Memory Management**: Windows utilizes virtual address space on both 32bit and 64bit, the primary difference being the amount of data addressed at once. 32bit addresses up to 4 gigabytes of memory at once, and 64bit addresses up to 8 terabytes. A thread cannot access memory that is already or in the process of being addressed. This is great because it helps prevent the corruption of data.
5. **Distributed Systems and Networks**: Web-based games typically host a database that is shared amongst all the application users because all users contribute to this shared server-side database. Drawing information from it per user request should not be an issue.
6. **Security**: Windows users have access to strong cyber security through their operating system. They can customize their firewall settings, set up admin privileges, and encrypt their personal folders, requiring a password to access the information. The Draw It or Lose It application could rely on data structures like hashing/chaining, which will implement a recursive algorithm. This structure will help encrypt and register player data kept server-side for the user. A login system could also be implemented as additional security user-side.