

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 | 03/22/2022 | Justin Foster | Initial iteration software design for Draw It or Lose It |
| 1.1 | 04/06/2022 | Justin Foster | Evaluation of Operating System options set |
| 1.2 | 04/18/2022 | Justin Foster | Recommendations completed |

**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 has developed an app, Draw It or Lose It, within the Android environment. The Gaming Room is looking to create a web-based version of this application, supported on multiple platforms. Only one instance of the game may exist in memory at any given time. Therefor, each game, team, and player name must be unique. A library of stock drawings has been provided by The Gaming Room to be used for the game.

## [Design Constraints](#_2et92p0)

The following design constraints must be addressed:

* Game and team names will need to have unique ID’s assigned to allow players to check if a name is already in use
* Only one instance of the game may exist in memory at any given time.
* The game currently exists only within the Android OS environment
* Teams must have players assigned to it
* Games will allow one or more teams to play

## [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 holds the main() method. SingletonTester is used to test if an instance of the game already exists using Direct Association. Entity will be the parent class for the Game, Team, and Player classes. Each of these classes will inherit the required attributes of the Entity class. GameService will have a game, and may only consist of one instance of the game at any given time. A game will have a team, and a team will have players. Team names must be unique, and player names must be unique. As shown in the UML diagram, a Player cannot have a team, a Team cannot have a Game, and a Game cannot have a GameService.

**"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 offers a simpler, more user-friendly operating system compared to other operating systems. Additionally, developing software on MacOS allows for an easier integration with other MacOS devices, such as iPhones.  Unfortunately, developing software on a Mac is expensive due to hardware and licensing costs. Furthermore, Mac is not a widely used operating system as, for example, Windows would be. | Linux was intended to be a free and open source operating system, and is popular for server-hosting. Linux is no doubt one of the lesser-used operating systems, but excels in cost efficiency. While your every day programmer/coder does not often use Linux, it is by far the most popular operating system for supercomputers and server hosting. | Windows is an expensive operating system in terms of product licensing costs and the operating system itself. While Windows does offer an intuitive GUI, it presents a great issue in terms of accessibility for more in-depth uses. Windows pales in comparison to Linux in server hosting. Windows is user-friendly and widely used by your every day user, but is not often a popular choice for coding and programming. | Mobile devices lack the superior functionality of other operating systems listed for server hosting. Mobile devices are meant to be compact and portable, leading to a sacrifice in hardware and server hosting capabilities. It is not impossible, but it certainly would not be the best option. Ultimately, as the development tools available on mobile devices are also available on other operating systems, there is little reason to choose a mobile device over the other options. |
| **Client Side** | The time investment of finding an individual both experienced and efficient with MacOS would be the largest challenge. Providing a suitable work environment to develop in MacOS would pose an issue as the hardware can be pricey compared to alternatives. | Similarly to MacOS, the difficulty in using Linux for developing is finding someone who is experienced and efficient with the operating system. However, costs for licensing and hardware would be lower and more manageable. | As noted above, Windows will be an expensive option for developing and coding. Finding an expert in developing on Windows is of utmost importance as the accessibility limitations require configurations to work around. | Mobile devices are a cost efficient and flexible option. General mobile device usage has become significantly more popular over the last decade, and being familiar with developing mobile apps is in higher demand. However, mobile devices is vague in the sense of the operating system being run. Android, iOS, and Windows being the most popular, you would need to be proficient in each to accurately and efficiently develop on a mobile device. |
| **Development Tools** | MacOS has a slew on useful developer tools that it supports, making it one of the more appealing choices for developing software. Software is developed in the Swift programming language. | Linux was largely coded in C and remains a popular choice for coding in Linux. However, Python and Java are also commonly used. | Visual Studio is the most intuitive and popular IDE for Windows devices. There are a wide variety of languages supported by Windows and virtually any could be used. Microsoft’s Windows kernel was primarily developed in C, and C++ remains a popular choice for most Windows applications. | IDEs for mobile devices are limited, and in many cases require use of a separate device to write the code in, then transition it to the mobile device. For example, an iOS mobile phone could be coded on MacOS using Swift, then transitioned to the phone. Android devices use Android Studio as the official IDE, while Java is the most commonly used programming language. |

## 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**: Taking into consideration the evaluation of each operating system, I would recommend the Linux Operating System for the Draw It or Lose It application development.
2. **Operating Systems Architectures**: Linux architecture consists of the Hardware, System, System Library, Kernel, and Shell. The Hardware portion consists of the CPU, HDD, and RAM for the operating system. The system portion, and system library, are reserved for special functions required by the operating system. The kernel portion of Linux is the core of the architecture. The kernel is involved in every major action by the Linux OS, and consists of kernel types for various functions. Lastly, the Shell is used as an interface between the user and the kernel. User inputs are taken by the shell and pushed to the kernel for various functions.
3. **Storage Management**: Taking into consideration the possibility of wanting to add more images to Draw It or Lose It later, I believe a cloud-based storage system would be best. In this case, Google Cloud seems like an excellent option. This will offer a cost-efficient option for storage, and allow expansion of the storage as it is deemed necessary. Furthermore, our storage needs would be handled by a third party and much of the burden of maintenance would be covered by the cost of the storage itself.
4. **Memory Management**: While C is a popular coding language for Linux, Java is also widely used for programming applications in Linux. To this point, Java has an automatic memory management tool. This “Garbage Collector” will allow memory management to be handled automatically. It would be our recommendation that Draw It or Lose It primarily be coded in Java.
5. **Distributed Systems and Networks**: The Draw It or Lose It app will be available in the Apple iOS store, Google Play store, and in web browsers for PCs. The RESTful API allows for seamless communication between the different clients accessing the app. REST API has criteria it must conform to, but is largely considered an easy API to use. The RESTful approach is favored in regards to mobile app development.
6. **Security**: Draw It or Lose It will use encryption for user information, and will utilize a multi-factor authentication sign-in process. As personal information may be shared by users, not knowing the repercussions of this, proper analysis will be performed on user input to ensure information such as IP addresses, physical addresses, and other personal information is not being spread. As Google Cloud is the recommended storage management service, much of the security of the server is handled by the provider and would not be a concern.