

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 09/17/21 | Joel Gomez Ruiz | First version of the Software Design Document |
| 2.0 | 10/01/21 | Joel Gomez Ruiz | Updated version of the Software Design Document |
| 3.0 | 10/15/21 | Joel Gomez Ruiz | Final version of the Software Design Document |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is a company interested in developing a web-based game to work on different platforms, expanding its application, Draw It or Lose It, currently only available for Android users.

There will be aspects to consider regarding the development of this game application such as the preferred programming language and its internal structure, which will consist in basically different and unique games, teams and players, all being referenced to one single instance of the Game Service hosting the application.

## [Design Constraints](#_2et92p0)

Technical constraints:

- The programming language to be used should be Java, as it is the main purpose of this project to develop the web-based application. Java can easily be used by any operating system, and it is well known for its use in web development along with JavaScript.

- A copyright-free library of images should be implemented to be used by the program when outputting clues during each game. Or images can be acquired by paying the respective royalties to their authors.

- The game application should have a unique name and ID, so no other instances of the same game can exist, this will prevent the creation of duplicates.

- The game should allow one or more teams per game in order for the game to take place as it has been specified.

- Any team added to the game should have its own name and ID to be properly identified, preventing duplication.

- Each team will have multiple players which will be identified by a unique number and player ID, avoiding the creation of duplicates.

## [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 diagram shown below presents a parent class or superclass named Entity, which contains the basic attributes and methods that child classes should inherit such as a name, an id, getter methods, an inaccessible default constructor to prevent empty instantiations (demonstrating encapsulation), and a public constructor that requires two parameters to build an object.

Classes Game, Team, and Player are the child classes of the Entity class (inheritance principle) and are also interrelated to each other. A Team object can be formed by zero or any number of objects of type Player, and similarly, a Game object can be formed by zero or any number of objects of type Team. Furthermore, another class named GameService (which is not a child of Entity class) is related to the Game class, where multiple objects of type Game can exist in the GameService object.

Each of the child classes also shows polymorphism, as each class overrides the toString method (which is used to return a String when it is needed to be displayed on the console) that they inherited from the superclass Entity.

The GameService class is defined as a class that can only have one instantiation of itself and no other calls could create another one, this being possible by using the singleton pattern, which consists in making its default constructor private to avoid external instantiation of any type and creating a method that would verify no more than one instantiation exists of the class during execution.

The ProgramDriver class can be seen as an example of the use of portability, having all these classes to be used in the main class where the execution of the game will take place.

Finally, the SingletonTester class is created in order to test that the GameService class was implemented correctly and only one instantiation of itself exists.

**"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** | - It can run Apache servers  Advantages:  - It is the most secure because of its hardware and software, being less prone to attacks.  Weaknesses:  - This platform is more costly whenever new updates are released, and it also has a more costly hardware | - It runs Apache servers  Advantages:  -It is free  - Less resources are needed, so it can run on low-end hardware  Weaknesses:  - Because it is open-sourced it lacks hardware and technical support. | - It uses Windows servers and is able to run applications such as ASP, .NET, Microsoft Access or MSSQL databases.  Advantages:  - User friendly interface  Weaknesses:  - Requires high-end hardware, otherwise server would crush  - It requires constant updates, so it is possible for the server to be down sometimes | Although mobile devices may provide portability respect to other operating platforms, applications usually require a lot of resources that mobile devices may not be able to provide, besides the fact of problems of connectivity that are more likely to occur. |
| **Client Side** | Cost: Mac users unfortunately need higher investment due to the high cost of hardware and software  Expertise: Not as much needed as Linux  Time: Consistent with size of the project | Cost: Free open sourced system  Expertise: It may require a lot more expertise than Mac and Windows  Time: It may be time consuming | Cost: Moderately costly  Expertise: User friendly and easy to access and widely used  Time: Consistent with size of the project | Cost: Moderately costly  Expertise: It can be more challenging than desktop development  Time: It may be time consuming |
| **Development Tools** | Programming languages: JavaScript, Java, Python, PHP, C++ and more.  IDE: Visual Studio, Atom, Pycharm, Webstorm, and more.  Only a team would be required as most of these programming languages and IDEs are easy to use and implement. Furthermore, most of the IDEs provided offer are also free. | Programming languages:  Python, C, C++, Perl, Java and more.  IDE: Sublime Text 3, Visual Studio Code, PhpStorm.  Just like Mac, these programming languages and IDEs are free to use and team size may not be affected as these tools are commonly known by developers. | Programming languages: JavaScript,  Python, PHP, Ruby, Java and more.  IDE: Visual Studio, IntelliJ IDEA, Aptana Studio 3.  Similar to Linux and Mac, these programming languages and IDEs are free and open sourced, without a major impact on the number of teams required. | Programming languages:  React Native, Kotlin, Swift, Python.  IDE: Android Studio, Eclipse, XCode  Although programming languages and IDEs are free, XCode may require the payment of a fee.  Overall, development in mobile devices may require extra teams. |

## Recommendations

1. **Operating Platform**:

After having analyzed the different operating platforms and considered their possible advantages or disadvantages, Linux would be an excellent choice to host the server for this application.

1. **Operating Systems Architectures**:

This operating platform needs much less resources than Windows or Mac, therefore requiring less sophisticated hardware. It has fewer updates which are done very quickly so uptime is not compromised, and its own security features can make the server more stable. Additionally, Linux is a free operating system and also open-sourced, so modifications can be done by anyone as needed.

1. **Storage Management**:

Our client’s project requires the use of a large library of images that would be used for the purpose of the game. These images will need about 1.6 Gb for storage, so it would be advisable to consider the structure needed to allocate the images. For instance, the game consists in showing random pictures that serve as a clue to the puzzle, so quick direct access to specific images should be made, which can be possible through the method of contiguous allocation. This way, any image can be accessed very quickly by searching their index location. Furthermore, it is advisable that the size of the library is predefined to be static and have new updates simply replace old images rather than increasing the size of such list.

1. **Memory Management**:

Linux makes use of virtual memory to manage how processes take place, especially when there is a shortage of physical memory when many programs are run at the same time. Linux manages virtual memory by using the paging technique where memory and files are divided in pages, pages from files are then passed to memory whenever they are needed for a task, and when memory runs out, pages are swapped from the physical memory to the virtual memory as needed. Although the use of virtual memory can reduce speed, it can allow for larger programs than the physical memory to be executed. For this application, the whole image library will not be needed at once immediately as clues are supposed to be passed in a one-by-one basis, therefore another consideration that can be taken is to use the “lazy loading” technique, so only images that are visible for the user are loaded first while other images can be loaded with less priority.

1. **Distributed Systems and Networks**:

The approach taken to make the game available to other platforms besides Android has been to make use of the client-server architecture to use a Wide Area Network, which is how the client also envisioned its application to be web-based. Following this approach, the application is stored in a server, containing all the data and resources needed for it to work. On the other side, the clients (such as users/players and administrators) can make requests (such as starting a game or adding players to a team) to the server through the browser, the server receives such requests and performs the required tasks returning what was asked to the client. Because of this architecture it is also suggested to consider possible factors that could affect the best performance of the application such as connectivity (slower speed connections could affect the communication between client and server), outages (the server should be able to be operative uninterrupted, therefore it should have a contingency plan such as the use of backup generators), and possible system updates which may cause it to go temporarily down.

1. **Security**:

Having several platforms accessing the server of the application it is important to set up some security mechanism to prevent unauthorized access. For instance, it is important to set up a firewall, which would manage the traffic of data between the server and the clients, reducing the possibility of risky connections. Also, a two-step verification password would be much better than a simple one in order to ensure a much reliable connection with the real owner of the account. Furthermore, the program should be designed following the “least privilege” principle, allowing the least possible access to efficiently perform a task, this way the whole system is not compromised in case of an attack and the damage can be contained to a small area. This can be found very clearly in Linux, as permissions are only given to a “superuser”, granting only sufficient and restricted access to common users.