

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 | 03/30/2022 | Elizabeth Robles | Evaluation of each traditional operating system |

## [Executive Summary](#_sbfa50wo7nsh)

We have recently taken on a new client here at Creative Technology Solutions, The Gaming Room, they are aiming to develop a new game called “Draw It or Lose It”. This game will be web-based and served across several platforms, which at the moment, is only available in the Android app. The game is simple, one or more teams compete to guess what is being drawn, however, rather than actually drawing, a large library of stock drawings will render images as clues to the player. Drawings are rendered at a slow pace, and the team must guess the puzzle before the image is fully rendered, if not, then the remaining teams have the chance to offer a guess to solve it; there four rounds per game. Some other requirements include: each game having the opportunity to have more than one team involved, multiple players assigned to each team, unique team names, and only one instance of the game existing in memory at any given time. We will be streamlining the development of this game using the Java programming language.

## [Design Constraints](#_2et92p0)

As a game that is already available in the app store for Android users, our job here is to figure out a way to integrate the code that already exists for the android app with code we can create to be accessible on any computer, that way the game can be played on any OS. Our challenge here will be making the web-based game accessible to different operating platforms, such as Macs, Windows, and Linux. Different code could be used for each OS, however, my suggesting would be to stick with Java, as it runs on Java Virtual Machine and can be converted into generic bytecodes that are executed on any platform. Another option would be to use several programming languages that could work together. Below are some design constraints that could also delay game development:

* Creating a game that can have multiple teams play together at the same time
* Creating only one instance of the game in memory at any given time
* Programming languages to work for many operating systems

## [Domain Model](#_8h2ehzxfam4o)

From The Gaming Room UML Diagram below, we can see that Entity serves as the base class for Game, Team, and Player classes, all those classes inherit from Entity, the superclass, we know this because all 3 classes have an open arrow pointing to Entity. From the diagram we can see that all those classes I just listed all share name and id attributes from the Entity class. The 0..\* between Game and GameService tells us that GameService is associated with 0 or more objects in Game class, and that every object in Game class is associated with exactly one object in GameService class. In fact, Game, Team, and Player classes are all connected with the same 0..\* symbol. ProgramDriver contains our main which drives the entire program and also uses SingletonTester.

**"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** | **Advantages**:  \*Easy server configuration and accessibility  \*Features are easily created using a MacOS server  \*Good interface  \*Can run a virtualized PC on the cloud  \*ADP files can be opened  **Disadvantages**:  \*More expensive than Windows or Linux  \*Difficult to make work with LDAP  \*Sometimes no support on software or hardware packages | **Advantages**:  \*LDAP works best with Linux systems  \*Easy server configuration and accessibility  \*More cost-effective option compared to Mac (licensing is free)  \*Reliable and efficient  \*Can run a virtualized PC on the cloud  **Disadvantages**:  \*Complex update processing  \*Some versions do not come with support | **Advantages**:  \*Much simpler operating system compared to Linux and Mac. \*User has control over all the hardware (which can be both a benefit and drawback in some situations). \*Beginner friendly \*Support is guaranteed  \*ADP files can be opened  **Disadvantages**:  \*Licensing can be expensive,  \*There can be security issues  \*Difficult to make work with LDAP | **Advantages**:  \*There may be lower costs involved with this option  \*Depending on the device there can also be better specifications  \*Licensing can be affordable  \*iPhone can run a virtualized PC on the cloud  \*ADP files can be opened for iPhones  **Disadvantages**:  \*Security is not as great as the other 3 options.  \*Challenge in compatibility with iOS and Android |
| **Client Side** | Some software development considerations include pricing, which for Mac can be expensive, also there is some expertise that is needed, if not, a lot of time will be spent on this. | Linux is probably the most challenging out of all 4, a lot of expertise is needed, however, costs can be low, and this OS is accessible for free to the public | For Windows, it is much easier to use than both Mac and Linux, price is also manageable so a lot of extra time will not be needed when using Windows. | Expertise is not as needed as the others. Can be more complex and difficult to implement than others. User interaction is a lot more important here since we interact with devices differently than we do on the web. |
| **Development Tools** | Best IDE’s include Visual Studio, Eclipse IDE, and Atom, which allows you to use several programming languages. Programming languages best for Macs are Ruby, JavaScript, Python and PHP. | The best IDE’s include IntelliJ IDEA, Eclipse, and Apache NetBeans. The best programming languages for Linux are Perl, Ruby, Java, HTML, and C++. | The best IDE’s include Visual Studio(an IDE from Microsoft), PyCharm, Atom and Eclipse. The best programming language for Windows is JavaScript, and C-Sharp. | The best IDE’s are Android Studio or QT IDE for androids, and for iOS devices, Xcode or Appcode would be best. The best programming languages are Kotlin and Java for Androids and Swift, C#, or Objective C for iOS devices. |

## Recommendations

1. **Operating Platform**: An appropriate server platform I recommend that will allow The Gaming Room to expand Draw It or Lose It to other computing environments will be to use Linux OS. This operating system is reliable and efficient, it is the most cost effective as licensing is free and so are software fees and can also run on a virtualized PC on the cloud. There are also many game engines to choose from as well, depending on what The Gaming Room is looking for, we can create 3D games, role-playing games, text adventures, and more.
2. **Operating Systems Architectures**: For my clients Draw It or Lose It game application I would recommend a multi-tier architecture, that is because it will be a web-based application. It is also the best option because the process will be handled in the inner layers and not the user interface or data. It can be more challenging to build, but it is the most secure as the player will not be allowed to directly communicate with the database and the game will likely run a lot quicker.
3. **Storage Management:** When it comes to storage management, the best option for the Draw It or Lose It game will be to use a cloud-based infrastructure, that is because, this game application will need a large library of images files. With cloud-based storage, we will be able to add more storage as needed, let’s say, if The Gaming Room would like to add more images to their library, this can easily be done with this infrastructure. It is also the most cost-effective option for our client.
4. **Memory Management:** To help manage memory we can start by limiting the Draw It or Lose It game to only a specific number of players, that way the game can run smoothly on a consistent basis. Image size is also another important factor here, as large photos could potentially slow down the game, we could attempt to avoid this using image caching. We could also use stack memory, which is a temporary storage memory, and when a function is called the memory size needed is allocated, and when the call is over, that memory will be deallocated.
5. **Distributed Systems and Networks**: On a network, there can be various tasks that are spread apart onto several nodes instead of just one, because of this, tasks can be done expeditiously. The Draw It or Lose It game can be complex with many aspects involving players, image library, account info, and more, so having a distributed system that can handle various tasks is important. Having a distributed system can also reduce the risk of failures, if one or two nodes were to fail, the game would be able to continue to run. It can also handle heavy traffic seamlessly, as opposed to when there is only one server. If there happened to be a need for more power, there is the ability to add servers to the system as well.
6. **Security**: There are several network security tools we can use since our client has expressed how important it is to protect user information between various platforms. To protect their data we can use DLP (data loss prevention) techniques to protect users from misusing data. We can also incorporate intrusion detection which will automatically analyze and scan the network so database attacks can be caught immediately. For the player, we can include two-factor authentication to help protect the users account from intruders. Lastly, since we will likely use a cloud-based system for the application, there will be a multilayered security that involves access control systems, threat monitoring, encryption for data in transit and at rest, application security, and a lot more.