

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 4**

**Evaluation 6**

[**Recommendations 8**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/20/23 | Michael Moore | Initial Document Creation |
| 2.0 | 06/04/23 | Michael Moore | Server/Client Pattern Considerations and Development Tools |
| 3.0 | 06/24/23 | Michael Moore | Finalizing Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wishes to create a team-based guessing game where the application will steadily render and image over the course of 30 seconds within a one-minute round. The application is to be web based and consist of teams playing through four rounds. The teams will have an assigned image that they will have to guess within the one-minute round. If the team whose turn it is does not guess the correct name of the image, the other teams may submit one guess within an additional 15 second block. The application is to be web-based and playable on any various computer operating system, including mobile devices. As such, the application will be Java based as that allows for the most compatibility and ease of application development for Draw It or Lose It.

## Requirements

The Gaming Room specifically requests these capabilities for Draw It or Lose It:

* Support for each game instance to consist of one or more teams.
* Support for each team to consist of one or more players.
* Support to allow unique game instance and team names within a game instance.
* Support for a user to check whether a team name is already in use when choosing a team name.
* Application designed such that only one instance of each game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

- A web-based application should be made to be compatible with as many web browsers as possible. This is a technical requirement and is being handled via the application being made with Java. Java is a very versatile language that is supported on Windows, Mac, Linux, and mobile operating platforms.

- A web-based application also needs to designed such that it is efficient and very responsive to input. Low latency and loading times via optimization of the application are also necessary to minimize network throughput and will make the game run well for the users playing together.

- The application must be designed to only host one instance of a game in memory such that hardware requirements are lowered and also less data is required to be transferred between the hosting service and all users.

## [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 this UML, we see that the Entity class is a parent class of Game, Team, and Player. We also see that through association through the Game class, the GameService class exists to provide the functions that make up Draw It or Lose It. The overall application through the Entity class, will consist of any number of Game classes, or each instance of a game. Then, each Game will consist of any number of the Team class, therefore each individual team playing in a given game. Finally, each Team class will consist of any number of the Player class, which is then the individual players that make up a team. Through the GameService class, the methods are provided to add Game, Team, and Player class members, therefore to create the game instances and the teams and players for each game. The Entity class is what pulls all the child classes together. The ProgramDriver contains the main() method of which the game will launch, run, and finish through. The SingletonTester class contains the method of which to ensure only one game instance exists in memory.

The UML diagram portrays the four basic principles of object-oriented programming and also fulfills the software requirements of The Gaming Room. Encapsulation is shown as data that is unique to each game, team, and player is private. This data is manipulated via public methods of which add names and Id numbers for each unique class member. This allows hidden data values of class members as it can only be manipulated the public methods. Inheritance is also shown in tandem with encapsulation. The classes are all linked together and provide their own methods of which to manipulate data of other classes. The classes inherit the necessary variables and methods to have their own unique name and Id. Each class that will contain another class contains the method to create each class under it, such as the Game class has the method to add a Team class member to itself. Abstraction is also handled via the public methods of the various classes. A user only needs to be able to add a name for their game, team, or player. In this program, the user would be prompted for such and not be bogged down by the background processes of updating the private variables and the other supporting code creating the unique members of the classes along the way.

The UML diagram shows that each Game class is capable of adding various amounts of the Team class and also of the Player class. This allows for meeting each game to have at least one team and then at least one player per team. Within the method to add a Team, the method would be written such that it would cross-check previously selected team names and notify the user if the desired team name was already in use. This would also apply to Game instances as well, thus meeting the requirement of unique names and the ability to check for the user. The SingletonTester class would ensure only one game instance exists at a time, meeting that requirement.

**"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 offers the ability to host a website innately to the operating system. It is fairly easy to setup and even includes built-in Java support, which is very helpful for this application. A Mac system is typically more expensive than alternatives, but Apple provides a level of technical support at a higher standard than that of Windows. The upfront cost of Mac and the specialization with Mac might deter use. | A Linux system is one of the more common web-hosting systems. While it requires a lot more knowledge and technical expertise upfront, a Linux system requires much less in terms of hardware. Linux is also an open-source operating system, so there is a lot of data, tools, application templates available for one to perform such a task. The open-source portion of hosting also makes it a cheaper option if not free altogether. | Windows Server also innately supports web-hosting. The Windows web-hosting platform can also be setup to utilize Java fairly easily, making it a useful choice for this application. Windows servers require more maintenance and more well-defined setup to run well in comparison to that of a Linux. However, it will cost to maintain a server via Windows | A mobile device would not be good for hosting the likes of a web-based game. The overall computer power in a mobile device, while still strong, is not nearly as adequate as any computer running a desktop system. However, most mobile devices are Linux based, therefore it is possible to utilize one to do so. |
| **Client Side** | Mac computers require a bit of familiarity to develop and implement software on. The same goes for Windows, but by majority Windows is much more popular. The overall cost to get the game up and running would be higher than other options. Based off of the developer, time to implement this application would not be an issue. Implementing a web-based game could benefit by using no browser specific code or functions. Additionally, the team should include a function checker to determine if a browser attempting to access the game is compatible. | Linux is more of a “barebones” operating system and a lot of the features provided by Mac and Windows have to be done manually by the developer. Linux offers support with almost all general web hosting software, so the added time of configuring the operating system to perform the web-hosting can be justified. Again, Linux is open source, thus free for most of the things you would need to host and run the game. | Windows computers are the most widely used system in the world. Thus, more people are going to have skill developing with this type of system. There is an associated cost to get a system like this performing the web-hosting, but it is not as high as a Mac. The overall implementation time for the program also would be relatively low, as there exists plenty of documentation and information on how to do so. | Development with the various mobile device platforms could be relatively inexpensive. However, getting a mobile device setup to support this application would probably be difficult as a Linux system would be. Mobile devices vary in operating system and could lead to a lot of difficulties and time wasted trying to perform this task. |
| **Development Tools**  Generally, most IDE’s and languages are free. Access to special functions, features, or support could incur additional costs. Multiple teams should not be necessary. The server handles the processes while the website transits the HTTP to allow communication over the browser. | Mac supports Eclipse, Visual Studio, PyCharm, and it’s built in Xcode IDE platforms. The list of supported IDEs is much longer, but these few will allow one to create Mac, Windows, and cross-platform programs with ease in mostly any language. | Linux has Eclipse, Bluefish, Apache NetBeans, IntelliJ IDEA, and loads more in terms of IDEs. Linux is also compatible for developing almost any language as are the other systems. | Windows supports Eclipse, Visual Studio, PyCharm, and a whole host of other IDEs. Windows also has the ability to program in mostly any language with the write IDE. | Android Studio and Xcode are the two main choices for mobile devices, Android and IOS respectively. Mobile Devices have come a long way in terms of functionality for programming and can be used for a large number of languages, but not as many as traditional computer systems. |

## Recommendations

1. **Operating Platform**: I recommend utilizing a Windows based computer for development of Draw It or Lose It. The vast supply of development suites, compatible programming languages, and documentation of which to create and expand a web-based game makes Windows a very easy choice and system of which to do so.
2. **Operating Systems Architectures**: Windows compromises a mix of kernel, driver, and executive services. The kernel allows for code to operate with complete and unrestricted access to system hardware. The driver level runs the communication between hardware and then the executive services control the input/output, power, virtual memory, and a whole host of other optimization processes for the eventual running of an executable file or other application. Windows has a long history of being utilized as a development platform, therefore there are many developers that are comfortable and skilled at working with Windows as a platform for developing and hosting server-based applications. There also exists a plethora of information on not only how to develop this type of application, but how to host it as well.
3. **Storage Management**: The built in Windows Storage Management allows for the user to manage the storage device in question. One can create specified partitions that a game is allowed to operate on, which can be useful in keeping space free for other processes. One can also optimize and monitor status of storage devices on the system via this tool as well.
4. **Memory Management**: Windows allocates each instruction to a specific memory address for the duration of use and processing by the CPU. It will reside there until it is handled and no longer required by the system. This is beneficial as memory space limited, but it can be expanded by utilizing storage memory for handling CPU data. The constant freeing up of RAM memory is much preferred as RAM is significantly faster than storage memory. Windows innately handles this decision-making process. The program is designed with memory savings in mind with the SingletonTester portion of the code to ensure that duplicates of items that take up memory exist via the program’s doings.
5. **Distributed Systems and Networks**: The program would need to be able to store the entire image and load it incrementally throughout a round such that connectivity issues would not complete ruin the game experience. It should also be setup such that if an outage of any player or the game host occurs, the game timer would pause until all players are present or say the other players vote to remove the issued player. The program would send data to update the status of the image and the game, such as time, round count, etc. The web-based game being in Java makes in compatible with all of the operating systems mentioned.
6. **Security**: Windows offers a Firewall and Windows Security suite for the client. Windows Security offers virus protection and protection from other malware and harmful software. The security suite can be setup to often search for said software. The true line of defense is the Firewall. The Firewall controls access of data to and from a computer and other computers on a network. It should be configured such that outside entities can only access the game application via the hosting on the internet. Even then, outside entities should only be able to interact with the data presented to them, and not manipulate the data. That is the whole purpose of the Firewall. The program is also written such that data such as names and user ID are private and not accessible via any user.