

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 | 11/13/2021 | Johnny Lingafelter | Initial document revisions |

**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 wishes to develop a web-based game called “Draw It or Lose It”. The gameplay is loosely based on the rules of “Win, Lose or Draw”, and teams will compete against each other to guess what is being drawn on the screen. The game should allow for one or more teams composed of one or more players.

## [Design Constraints](#_2et92p0)

* Must be developed as a web-based application
  + The language used to develop the game will be HTML/JavaScript or Java (applet)
  + Must have server-side technology in place to host the application
  + Application will need to be tested on all major browsers and operating systems
* Game must allow for one or more teams to compete against each other
* Teams can have one or more players
* Only one instance of the game can exist in memory at any given time.
* Game and team names must be unique

## [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 GameService class is implemented using the Singleton design pattern. This ensures that only one instance of the GameService class is in use at any given time, and the instance is used as a “factory” to create Game, Team, and Player instances for use in the application.

The Entity class is the base class for the Game, Team, and Player classes. This hierarchy of classes demonstrates object-oriented programming. Game, Team, and Player extend the Entity class, therefore inheriting the attributes of the Entity class. In this case, the “id” and “name” attributes. All instances of the Game, Team, and Player classes share these attributes with the Entity class.

Finally, the multiplicities shown indicate that the GameService instance may contain zero to many Game instances, a Game instance may contain zero to many Team instances, and a Team instance may contain zero to many Player instances.

**"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 can be used as a web server without any problems. It can be configured to run an array of different software for this purpose. The main disadvantage I see is the price tag. | Linux is a tried-and-true OS for hosting web content. The Apache server is very robust and fast. The main drawback to using it is that it is complicated, and fixing problems sometimes takes too long. | Windows’s server is perhaps the easiest to use, its GUI interface makes it much less complicated for inexperienced users. My only gripe with it is that it has many security flaws and can be a bit buggy at times. | A mobile device is by no means the right choice for this project. While it is possible for a mobile device act as a web server, the purpose is to host personal content and serve it to peers. A web-based game with high network traffic is not a good fit for a mobile device. |
| **Client Side** | Since this is a web-based application, the client-side considerations will be somewhat similar across all platforms. Not exactly the same, but similar. Moderate cost, time, and expertise would go into testing the web-based application on all major browser on a Mac (Safari, Chrome, Firefox, etc.) | Moderate cost, time, and expertise required for Linux client-side testing. It should be noted that we are not developing a different application for each platform/browser, rather we are developing one application and testing and tweaking it to make sure that it runs on all browsers on all platforms. | Same, moderate cost, time, and expertise to test the major browsers here. Microsoft Edge is specific to this platform, but it provides no greater challenges than other browsers of the same nature. | Moderate to high cost, time, and expertise. While mobile app development can be tricky it is not too different from the average browser. A good understanding of CSS is helpful for mobile layout. |
| **Development Tools** | Again, this is not “Mac-specific” software. The point of a web-based application, as requested by the client, is that it is able to be run by most browsers on all operating systems. We are not deploying this on Mac, we are deploying this to browsers. If developing on a Mac, there are a few IDEs I would use: Notepad++ is great for HTML/CSS/JavaScript, Visual Studio Code is great as well. Or Vim + plugins. I would also use the “LAMP” server set up to testing during the development process. | Eclipse should be used if developing a Java applet. For HTML/CSS/JavaScript there is Vim, KATE, and Gedit. All of these IDEs have syntax highlighting and auto-completion that aid in the development process. Again, I would use a LAMP server here for testing too. | If developing on Windows, IDEs available are Visual Studio, Notepad++, and PhpStorm for HTML/CSS/JavaScript editing. I would use the latest version of Windows Server for testing purposes to test browsers on Windows. Or I would install LAMP on Windows. | Chrome has an excellent tool called Device Simulation that allows you to test, on a desktop, how a web site/app looks on a mobile device. There are also emulators for iOS and Android that allow you to see how the site will look on mobile from a desktop computer. Of course, the best thing to do is always test the site on an actual mobile device to make sure there are no surprises. |

## 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**: I would recommend Windows, without question. It is straightforward and easy to use, has technical support, and is widely used and recognized. Linux is stable and usable for this application, but I would not recommend it because of the difficulty associated with its use.
2. **Operating Systems Architectures**: Windows is written in C, it supports multi-threaded applications and virtual and physical memory addressing. It consists of a user mode and a kernel mode which provides a layer of security between the internal components of the operating system and the user interface. Most operating systems are written for the x86 CPU architecture, so I would recommend an x86-based system, preferably 64-bit.
3. **Storage Management**: I would recommend using cloud storage services such as Microsoft Azure or Amazon Web Services to manage the storage needs required. This gives you unlimited flexibility with regard to the actual amount of space needed, and also gives you peace of mind since you never have to worry about hardware malfunction or losing your data. The data needed can be pulled from the cloud storage at runtime and used to play the game. I would recommend using an SSD (solid state drive) for local storage on the server as it is the fastest option for the price.
4. **Memory Management**: All modern-day operating systems use robust memory management techniques beyond the scope of the Draw It or Lost It game. Windows does not provide anything special in this respect (in my opinion). Windows uses protected memory so there is no risk of data being corrupted by another process, just as it is on Linux or Mac. I would recommend using experienced developers that know how to write clean code when writing the client-side functionality for playing the game in a browser, as memory leaks in JavaScript can be devastating to a program if left unchecked.
5. **Distributed Systems and Networks**: The Draw It or Lose It application can use a database to share player data among different platforms. I would recommend using mySQL for database needs, as it is a free, open-source option that runs on multiple platforms allowing data to be distributed across the player network. Practicing proper database maintenance, The Gaming Room can ensure this data is never lost or corrupted by regularly backing up this database and making it fault tolerant across the network.
6. **Security**: User data will be stored in the database and be sent across the network when appropriate to accommodate gameplay. By ensuring that the database used is configured correctly, one can rest assured that there will not be a security breach at the database level. However, user information from the database will be sent across the network and it is recommended that you use a secure connection with proper SSL certificates to encrypt the data to guard against an attack. Windows has robust functionality for securing web sites using SSL certificates and if configured properly there should be no need to worry about a data breach at the network level.