

# **Draw It or Lose It**

# **CS 230 Project Software Design Template**

Version 3.1

## 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 | 9/15/22 | Kayleigh Kinsey | Added Executive Summary, Design Constraints, and Domain Model |
| 2.0 | 9/30/22 | Kayleigh Kinsey | Filled out the Evaluation section |
| 3.0 | 10/13/22 | Kayleigh Kinsey | Filled out the Recommendations section |
| 3.1 | 10/14/22 | Kayleigh Kinsey | Added to the Server and Client side development requirements in the Evaluation section. |

**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)

Draw It or Lose It is a mobile game developed by The Gaming Room. It is currently only available on Android devices, but The Gaming Room is looking to develop a web-based version of the game that is available on multiple platforms.

## [Design Constraints](#_2et92p0)

- The application must be web based.

- Only one instance of a game can exist at a time. The program will need to implement the singleton design pattern to ensure there are no duplicates.

- A single game must have 1 or more teams. The program should make sure the number of teams the user requests is no less than 1.

- A single team must have more than one player. The program should make sure the number of players the user requests is no less than 2 times the number of teams.

- Each game and team must be assigned a unique name. The program will need to be able to check if a name is in use and notify the user.

## [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 Entity class serves as a base for the Game, Team, and Player classes, which all inherit from it. It holds the fields and methods that all of its child classes share so the same code doesn’t have to be copied to each class.

The GameService class manages the current games and ensures no more than one of the same game instance exists at a time. It can hold zero to many instances in a list of games. The Game and Team classes work much the same, except the Game class manages a list of zero to many teams and the Team class manages a list of zero to many players.

The ProgramDriver class uses the SingletonTester class. It also houses the main() method.

The diagram implements abstraction and encapsulation by keeping fields inaccessible to classes that don’t need it and instead using public accessor methods to pass the values in these fields to any classes that request it. The classes do not need to understand what is going on inside the others. They just call the methods they need and focus on their own tasks. The diagram implements inheritance and polymorphism with the Entity class, which contains the fields and methods that its children will all need. This helps to keep the code from being redundant and makes it easier to read.

**"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 is not often used to host web applications. It can be used though, and may be a good option if the application is meant for Mac and iOS products. MacOS products are usually much more expensive than those that run other operating systems. Mac systems are known for being very secure. There is less concern about attacks on the server side of the application if MacOS is the operating platform. Mac is very user friendly and easy to learn, but the user does not have as much freedom to alter their experience as in other operating systems. | There are a lot of available Linux distributions. Most servers run Linux. Linux is flexible and customizable. It is good for hosting multiple platforms. There are no licensing fees because Linux is open source and free. Linux is also a very secure operating system despite its popularity in web hosting and its flexibility. Linux can be difficult to learn, especially for developers that are used to working in Windows or MacOS. | Offers more support for Windows products and optimized for Windows based applications. Licenses are expensive, and it may be necessary to pay for licenses to multiple Windows products. Though Windows is more secure than it once was, it is still the most vulnerable operating system. This is most likely due to the majority of computers running Windows. The Windows operating system is user friendly and easy to learn. | It is possible to host a website on a mobile device, but very rare. A mobile device likely would not be enough to handle a large website or deal with lots of clients. Mobile devices do offer excellent security, especially iOS. This is because the user has much more limited access to files and OS components, and thus it is harder for malware to gain that access. Mobile devices are designed to be as easy to use as possible, and nearly everyone is experienced with some form of mobile OS. The more secure a mobile device is though, the more restricted user access is, which may cause problems in development. |
| **Client Side** | Mac is very easy to learn and use, so it would not take long to get a team used to it if they aren’t already. Mac devices are expensive though. Apple devices use a browser called Safari, so if the application is going to be compatible with MacOS, it must work with Safari. The most widely used browsers, such as Google Chrome and Mozilla Firefox, are available on Mac as well. | Linux has a steep learning curve and may take time for a development team to get used to. Becoming comfortable with Linux requires experience. It is likely the cheapest operating system to use because it is open source. Because there are multiple versions of Linux, they do not all have the same default browser, but in general, most versions of Linux come with Mozilla Firefox and are compatible with all major browsers. | Windows is the most widely used operating system. Most developers are familiar with it and most customers will be using it. Computers that run Windows vary in price, but in general they are cheaper than the alternatives. Windows comes with Microsoft Edge as its default browser, which is available on other operating systems, but not as commonly used. Windows also supports Firefox, Chrome, and all other popular browsers. | To make the application accessible to mobile devices, a version of the site must be created that is optimized for mobile use. This could require reconfiguring most of the site to operate properly on a phone, which will add a considerable amount of time and cost to development. The browsers available on a mobile device depend on the operating system the device uses. For example, iOS comes with Safari and Android devices come with Google Chrome. Other browsers are available to download, so mobile devices generally have access to nearly all the same browsers. The main exception to this is Safari. |
| **Development Tools** | Apple recommends using Swift with Mac devices. This language is primarily used to develop iOS apps, so it is probably not the best choice for a web application. Languages like Java and C++ are better suited for that, but there are significantly less IDEs and development tools available for Mac than other operating systems. The Mac version of Visual Studio is not as versatile as the Windows version. Mac is ideal for developing iOS apps, but not much else. Mac Licensing is not as expensive as Windows. | Linux works best with languages such as Python, C++, and Java, which are suitable for creating a web based application. Linux also has quite a lot of development tools. Its flexibility also offers a lot to the ease of make developing and testing the app more streamlined. It does not have as many IDE options as Windows does. There may be licensing costs for some tools, but Linux itself has none. | C# is very popular for Windows and Android devices. It may be useful for importing an Android game to Windows. Windows is compatible with the largest selection of IDEs and languages of all the operating systems. Visual Studio is optimized for Windows, and on other operating systems it is more limited. Licensing can be expensive depending on how many Windows products the development team would need to use. | The most common language used to develop mobile (iOS) apps is Swift, which is not suited well to web development. It will cost extra time and money to optimize the website for both mobile and pc users, and may require an extra team of people. |

## 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**: The operating platform I recommend is a Linux server. There are many types of Linux servers to choose from, so The Gaming Room can select whichever one best meets the needs of the game. Linux is also highly customizable, so the platform can be even further tailored to The Gaming Room’s liking. Linux is open sourced as well, so there are no licensing fees for using it.
2. **Operating Systems Architectures**: I recommend using the Client-Server architectural design pattern, which is the most widely used architecture for web development. Its concept is simple; the entire application is split into a server side and a client side. The client side sends requests to the server side, which interprets and fulfills them. Both sides can share the burden of operation so the application runs more smoothly.
3. **Storage Management**: Very little if anything needs to be stored on the client side of the application. The images, UI, code, and anything else the game needs to function can be stored on the servers. Local storage could be implemented for user data, but that information would likely be more secure in a database on the server. It is better to keep most if not all of the game’s components stored on the servers because it will minimize how much the user will need to download, making it more convenient, and it keeps all the information in one place rather than copying and distributing it among multiple clients that don’t necessarily need their own copies.
4. **Memory Management**: To save space in memory, I recommend keeping significant components for a game instance in the memory of the client hosting the game instead of the server. The server will use its memory to manage ongoing games. The game residing in a client’s memory should be mostly self contained and capable of running on its own with the exception of occasional requests sent to the server for a new image (because keeping them all in client memory would waste a lot of space and slow down the application).
5. **Distributed Systems and Networks**: I recommend developing the architecture of the application with REST, which is an architectural style that includes generally agreed upon standards for a Client-Server design and optimizes its efficiency. In order for the clients to communicate with each other, I recommend setting up a Peer-to-peer network among clients that are currently in the same game. Each of the client computers is like a node connected to all of the others so they can share information. This will make it possible for the players in a game to see the the same visuals and all contribute to the same log of guesses.
6. **Security**: The application should be guarded by password authentication. When a user creates an account, they should be required to select a password that is of decent length and that includes numbers and lowercase and capital letters. The passwords will be very difficult to guess if these requirements are met. The user’s data should be encrypted and stored where only authorized users can access it. Two-factor authentication should also be employed to confirm a user’s identity if they attempt to alter or access their personal information.

**Sources:**

Brown, Korbin. *List of Browsers Available on Linux*. August 22, 2022. linuxconfig.org/list-of-browsers-available linux

Horne, Katie. *Operating Systems and Web Hosting: Read Here Before Making Your Choices*. digital.com/best-web-hosting/operating-systems/

McFadden, Christopher. *What Operating System is the Best Choice for Software Engineers*. interestingengineering.com/innovation/what-operating-system-is-the-best-choice-for-software-engineers

Tony. *Linux-How Does Memory Management Work?. medium.com/geekculture/linux-how-does-memory-management-work-863f86feaf0d*