d



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  2.0  3.0 | 09/27/2020  10/10/2020  10/24/2020 | Christopher Holmes  Christopher Holmes  Christopher Holmes | Creation of the document.  Executive Summary and Constraints created.  Domain Model started.  Added evaluation and development requirements.  Recommendations added. |

## [Executive Summary](#_sbfa50wo7nsh)

We will be making a game in Java based off a 1980s television show called *Win, Lose, or Draw*. On the game show the teams would have to guess what the drawing of their teammates were. For this game, we will render an image to simulate drawing using a stock drawing from a large library. It will take 30 seconds for the image to render completely. At 30 seconds the time will end and if the team did not guess what the image was, the next team will get 15 seconds to make one guess.   
The client requires that there be two or more teams. That each team has multiple players. The game names and team names need to be unique and there needs to be a way to check to see if each team is unique. Only one instance of game can exist in a memory at a time.

## [Design Constraints](#_2et92p0)

We will need to have the option for one or more teams. Each team will need to have the ability to have multiple players. Game and team names need to be unique. There needs to be an ability for the users to check whether a game is running. Only one instance of a game can exist in memory at a given time.

This game will be made in Java and could have difficulties running on operation systems have Java available or Java installed on the system. There will need to be a server to run the game on to let players play together. There will also be a need for storage of the images provided. This can be done both online and on the client device. If the client device is used it will require a lot of storage. If it is through the net it will require a decent internet connection. We will also need to protect the client’s information. If we host the game online, we will need a way to filter traffic to protect from DDOS attacks. We will need to make sure that we use an authenticator for the player. This will only let the player play his user. We will also need to add a way for the player to play together. The easiest solution is through the internet to have the player needed for each team. This requires more work on our part to get the app running. We may also want to have cross platform this can be a constraint because players want to play together if they are on different platforms.

## [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 ProgramDriver contains the main method and we have a way of testing and debugging with the SingletonTester. We will have a super class called Entity that will contain most of are methods. From there we derive the child classes Game, Team, and Player that will inherit the methods inside Entity. The child classes of Game and Team will contain arrays. Game will have an array with the list of team names and Team will have a list of the player names. It also contains methods to add teams and state the team names. This is also available in the Team class but it will add players name to teams. GameService, a separate class that will be used to run the game also has an array of the game names. This is where most of the methods are. It will have getInstance, addGame, GetGame, getGame (again), getGameCount, getNextPlayerId, and getNextTeamId.

In short, we have ProgramDriver that will run the program. SingletonTester is what will use to test and debug the programmer. Entity is the superclass that will give basic methods to Game, Team, and Player. Game, Team, and player will make unique names and make an array list for the necessary information. GameService will be class that have the methods that makes everything work. It will have the have everything to go to next person, start game, go to the next teams.

****

## [Evaluation](#_2o15spng8stw)

For hosting the game server, we have three options Mac, Linux, and Windows.

Mac

Mac is open source and runs on Unix OS. To open a server Mac, it will cost us $500 USD. The benefits of Mac is that it is excellent at hosting other Mac clients, the is support for all the Mac tools, easiest to administer on a smaller scale. The reasons why Mac is not the best for us is because we may have some difficulty making passwords and storing them on the server. Also, Mac is not as scalable as other server OS. If we decide to use packages that are not certified by Mac, then we may not have customer service to help us use the app.

Linux

Linux is also open source and runs on Unix. It is free to use Linux as our operating system for our server. It has lots of free open source apps to run. Linux is a very basic and very stable. It is also very secure with less viruses and malware. It also uses popular languages such as ruby, PHP, and Python. This makes it easy to find someone to work on the server. If we want customer support, we can also use payed options of Linux which will help us out a lot. Red Hat is one we can use. Red hat is a cloud service that will leave us with less responsibilities on our side. The drawbacks are that we cannot run Windows or Mac apps without workarounds, but free software can solve this problem.

Windows

Windows is generally the most popular OS. Using it as a server will allow us to access the apps that Windows has and run them. The license for Windows is costly and for each user to access the server configuration we will have to pay for a new license. Windows is the most user-friendly OS on servers. Windows also has a large support with lots of third-party apps. The apps are easy optimized and there is guaranteed lifetime support. There is also the option of system recovery to fix problems if they arise in the server. The problem with Windows is there are many viruses and malware that target the system.

Mobile Devices

This one is not suggested as being your server. There are two main OS for mobile devices, these are iOS and Android. The browsers most popular for these two are Safari and Chrome. Since mobile devices are not wired, there could be disconnection issues if there is too much data needed to be sent and received. To develop on Android, we can use Eclipse which is not much hassle. To develop on iOS, we would need a separate team to make the application if it will not be browser based. Running games on browsers on mobile devices causes longer loading times and more data usage. The best option to make an app. If we used browser as planned then we would need to have an adjustment for mobile devices so they will run correctly and display correctly.

The browsers we choose to host will affect the availability of the game to the clients. There are a few major browsers.

Chrome – The most popular. Available on all platforms. Preinstalled on all Android devices. The most popular all around. It is only second in the tablet group to Safari and the difference is minimal.

Safari – An apple-based browser, comes with apple devices. Whether it is Mac, iPhone, or iPad. The most popular on tablets.

Firefox- Available on are platforms but not as popular because it does not come preinstalled. Firefox is the third most popular browser on desktops.

Microsoft Edge- Forced installation on Windows. It is the default browser of Windows.

Internet Explorer – Becoming extinct because Microsoft is forcing most users to use Edge.

The three most popular browsers for desktops are Chrome, Firefox, and Safari.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac and Linux are both open source and Unix OS. A mac server OS cost $499 USD from the start. The advantages are that it is great at hosting other Mac clients. Support for all the Mac tools and apps. Easiest administration on a small scale. The cons are that to scale there needs to be an intimate knowledge of the system to scale. Some packages may not allow to save passwords. Many packages for Mac are not certified, means no support for the package. | Linux and Mac are both open source and Unix OS. The great thing about Linux has a free server OS which we can run. This is one of the preferred OSs of many developers because that is OS. Linux is very stable. It also has many free tools to use. It is also can be scripted in common programming languages such as Python, PHP, and Ruby. The negative side is that Linux has no customer service. There are companies that offer services through Linux, but these will have to be paid. Another drawback is that normal apps from windows or Mac cannot run on it, but with free tools there is a workaround.  For hosting online, we would need to use Apache or Nginx. Linux does have LDAP available. | Windows is a very beginner friendly software. Running a windows server cost money to license it, and every new user will cost more money to run it. The benefits are that it supports many third-party apps, easy and optional optimized, guaranteed long term support, compatible with Microsoft exclusive programs, and that we can recover from a problem through system recovery. Windows does have LDAP available. The bad thing about it is there is lots of security risk for viruses and malware. | Mobile device devices can run servers, but the options can be very limited by the capacity of the phone. It is not very scalable. The server would also not benefit of running through the phone providers internet, without prior arrangements because of firewalls. If running at home, the phones generally do not have an ethernet cable so it will not benefit from speed of the cables. I do not think there is a dedicated server OS for mobile devices. |
| **Client Side** | The most used browser for Mac would be Safari. It is the browser that comes preinstalled in Safari. Chrome and Firefox are also popular. Chrome is the most popular of all the browsers and should be the focus of our time. Safari is what we would need to focus our time if we were focusing solely on Mac as it is the main browser from Mac. | The browsers that are the most popular for Linux is both Firefox and Google Chrome. These two are the most popular and would need to be our focus. This would be very cost effective because these two browsers are also available on Windows and Mac. | The browsers made by Microsoft are Internet Explorer and Microsoft Edge. Developing for these two would be similar about 8 to 10% of the market. Microsoft Edge is the more popular of the two. Chrome and Firefox are also available and widely used. If we have money to focus on one extra specifically for Windows it should be Microsoft Edge. | Mobile devices are generally Android and iOS. For tablets the main two browsers are Chrome and Safari, Safari has a larger percentage with about 50% of all users. Close behind is with about 40% of all users is Android. While on phones Chrome has about 65% of all users and Safari is second with about 23%. On Android Chrome comes preinstalled and on iOS Safari comes preinstalled. So, it would be cost effective to only use these two as they account for about 90% of all mobile devices. |
| **Development Tools** | While on Mac since our game is Java based, we can use Eclipse. We can also use NetBeans to develop too but Eclipse is known. Xcode is also very good on Mac. It is the best environment to develop for iOS and Mac. CodeRunner offers development on Mac using Java. The cost for joining the developing community is $99 a year. We can use other languages to develop on Mac too. Since it is a browser-based app, we do not need to pay the fee and just develop for the browser. | We can use Eclipse to develop for browsers on Linux. Eclipse is available for both Java and C++ programming. Since we have started development in Java Eclipse is a great option. | For developing on Windows there is a lot of options. Eclipse is a great IDE available on Windows to develop for Windows. It allows for cross platform development. | We do not use tools to develop on mobile devices. The best IDE for deploying on Android is Android Studio. Eclipse was used before Android Studio but if development for cross-platform is Eclipse is the better IDE for development. To release on Android, we will need to have a developer license for $20.  For iOS, Xcode is considered the most popular for developing on iOS. This one uses C++, C#, and C. One IDE that does use Java is CodeRunner. Since we have started development in Java, we should consider this one as an option.  < |

## 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 recommend using a Linux cloud server. This is because it will come with easier expansion and set up. There will be no cost in setting and running a server room and the team to make sure outages do not occur. Red Hat Linux is the most cost effective of the networks. Red Hat Linux Server is easy to deploy, administer, and has an easy to control operating system. Linux also has many free tools to use in the server. We will also be able to collect user data and user it for future marketing or market research. This data can be stored on our cloud server.
2. **Operating Systems Architectures**: I recommend using Java to make the program and design it for Chrome. Since we are using a browser Chrome is the most popular browser by far and if one browser must be chosen it needs to be Chrome. This will allow us to connect to a large percent of the mobile market and the computer market. If a second one must be chosen, then Safari is a close second. This is because it is made by the competitor of chrome. I believe although Firefox is popular, I theorize that the user will have either Safari or Chrome on their mobile device. This will also make them a potential user for our game.
3. **Storage Management**: For storage management I recommend we store data that is not important to the client on the server. For example, we can save team data and player data on the server to free up space. There is no need to save team data on every user device. I also recommend that we not use HD images and use images that look almost the same but are smaller for the storage. This will free up both storage and memory making the app fast and efficient.
4. **Memory Management**: For memory management we will need to make sure that when the pictures of the drawing load to RAM that they do to not overload the system. I recommend that we save the pictures to the storage device. This will allow for less memory processing to retrieve the image. We can also unused data in the virtual memory to help free up memory when the images are loading. This will allow for more efficient memory use and faster loading times. We can use segmentation, swapping, and paging to make sure the memory is super-efficient.
5. **Distributed Systems and Networks**: Internet connection is the best way to connect our clients. Having a Linux server will allow us to connect the platforms together in one server. This client server architecture is the best way to run the network. It is also the best way to distribute the service.
6. **Security**: Using a cloud server gets rid of much of the security issues of running a server. The most important part on our side would be verifying the user is the user for the game account. This can be done by registering the IP address of the user on the first entry and having a saved verified IP addresses that the user has used. Two step verification would be great for a new access to the users’ account. Having a password being the basic security method is a great start to account security. The second step in verification could be email or an SMS message. I would recommend email as it would be the most cost effective.